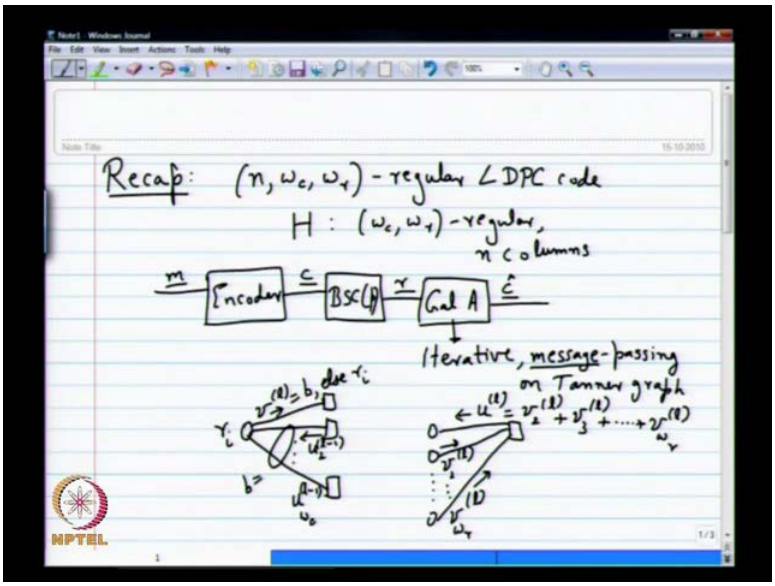
Coding Theory
Prof. Dr. Andrew Thangaraj
Department of Electronics and Communication Engineering
Indian Institute of Technology, Madras

Lecture - 25
Thresholds of LDPC Codes

(Refer Slide Time: 00:24)



So, let us begin the quick recap. So, well looking at the decoding of let us say an n, w c, we are regular LDPC code case. Once again when I say LDPC code, clearly it is not this just not define single code. So, what I mean is I am thinking of a randomly generated. So, the speak parity check matrix H with w c, w r regular, it right and n columns case and I am going to do this in some way. Then after that I am going to do a decoding for the code with priority check matrix H. So, clearly it is not unique but let us say we randomly pick one from this. Then what is the channel I am looking at? I am looking at case w e. We spoke very briefly about this encoder, essentially what I said was you have to do, you have to convert H to some kind of a systematic form. Then do the encoding case, that is what we want to go do and that is not very complex because H is parse.

Then let us say this goes through BSC with transition probability p and you get a received vector r. We look that Gal lager a decoding algorithm to come up with a C cap and this can be this Gal lager. A decoding algorithm can be very nicely described in the using the tanner graph of representation of with the parity check matrix for essentially. It

is iterative and in each iteration there are two steps, iterative message passing algorithm that is the main, that is the main idea.

There are two types of messages that are passed, the message from bit nodes to check nodes and then messages from check nodes to bit nodes. So, there are two parts are two steps inside each iteration. The first step is basically looks something like that. So, particular the bit to check messages were called v o u v, right v v. So, these messages the l basically depend on u 2 l minus 1, all the way down to u w c l minus 1. If all these guys agree and they are equal to b then this is going to be equal to b a, else whatever that was received by the from the channel. So, what will happen in the first iteration is you do not have any of these things.

So, clearly v l will be said to r i itself, yes that is what happens in the first iteration. All the messages will be equal to r i in the first iteration. The other iterations it will depend on the u that is received, what happens on the check node side is little bit simpler to describe what sent back here u l minus 1 is simply v 2 o.

So, u l may be right simply v 2 l plus v 3 l plus so on, till v w r l is. So, these are the incoming messages in the health iteration. So, these are the two steps, the messages are basically estimates of the messages. First of all flow along the edges and the edges connect to the bit node to check node every message is an estimate of that bit node which is connected to that h.

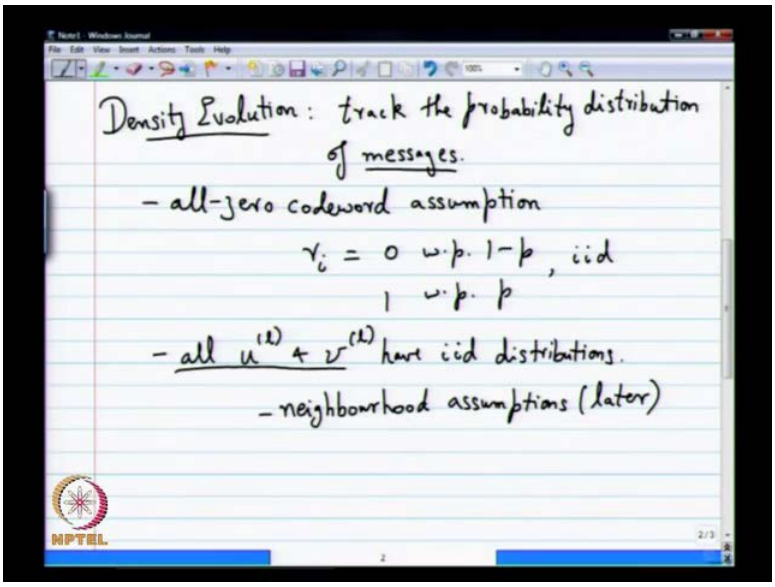
So, that is the main idea and the next idea that is kind of inbuilt here in this iterations ,if you receive some messages from a node when you sent back another message to that same node, you do not use that information. So, all whatever information is received from that node is not used for the instance u 1 l minus 1 is not used in v l.

So, same way here v 1 l is not used in generating u l. So, you do not use that same information, so that so basically messages have to be extrinsic that is the idea messages are estimates but they should be extrinsic estimates for that are being sentenced cannot be intrinsic or it cannot extrinsic in the sense is that it should not be previously know to that note that.

So, this the two rules and the critical thing that is done for first thing. You can do is first of all simulate this and see what happens. I showed you a plot, send and see that it does

well and next thing is to do some analysis and for analysis like I said like a been always saying will do approximate asymptotic kind of analysis.

(Refer Slide Time: 06:17)



The main idea is there is a density evolution. So, what is density evolution? So, it is basically in density evolution, you will track the probability distribution of messages. This is the general definition of the density evolution. Density evolution is some algorithm processes which tracks the probability distribution of messages. So, messages of course in iterative message passing decoder. If you do not have an iterative message passing decoder, we do not have any messages to track that is no problem. So, first of all I have been in the previous picture when I described the algorithm, I simply said, I simply, I gave a very simple description with respect to one bit node. Of course, in the tanner graph you have n bit nodes and n times w c edges.

So, each edges now carrying a message. First question we should ask is which messages probability distribution is being tracked here? Really? I mean if you want to track the probability distribution of every single message you are not going to able to do it, you have to hope or resume that all of them are according to some same distribution. So, you need some simplifying assumption like that so that you can track one density a. Suppose, to some n times w c density which are impossible to do.

So, that is the first assumption that you need and for that you need these all zero code word situation. Once you assume that all zero code word is transmitted, all the r i's have

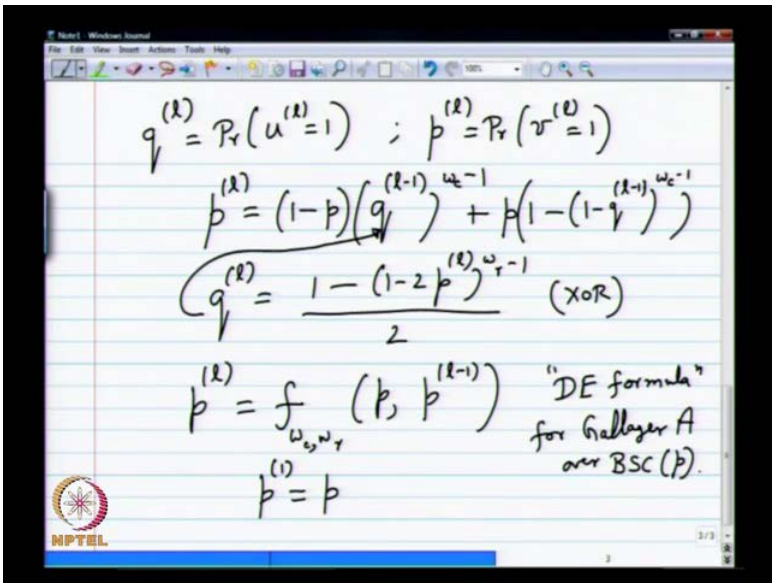
the same distribution, right? Once you make the all zero code word assumption, what is the distribution of r i is equal with 0 with probability 1 minus p and let us one with probability p and it is iid, right? Identically distributed for all i a r 1, r 2 everything as the same distribution.

So, clearly the first message that is going from the bit nodes to check nodes, you can just think of message on any edge. It will have the same distributions, it is enough if you track that. Now, as it terms out if you make some further assumptions, you can assume the following, you need to basically assume in any situation like this. The u 2 l minus 1 through u w c l minus 1 are in fact u 1 l minus 1 through u w c l minus 1 are also all iid k.

So, basically the assumption that you need is this is for the first message, for even subsequent iterations. You need to assume that all u l and v l have iid distributions, what do you mean by all v l and u l, u l represents the message that flows from the check node to the bit node, that is happening on all edges. So, that is what I mean by all. So, if you look at all these u l's you have to kind the resume that they are identical and independently distributed, same thing for v l messages are flowing from left to right. Once again will assume they all are identical and independent.

So, once you make these assumptions then the analysis is much easier but what you need further for these assumptions all zeros, not the only the thing that is needed. In fact, you need some further assumptions for this. So, I will tell you that what you need for that later, so you need some neighborhood assumptions. We will see this later but for now we will assume this and proceed with the analysis and see what we get some neighborhood assumptions are needed. We will see this later in a little while but if you assume that this is true then you can do density evolution relatively in a pain full pain free manner. So, what does the equation there are two equations were remember, right first equation.

(Refer Slide Time: 10:54)



There are two things we track the first thing we track is something that I called is p l which is probability that u l equals 1 u l was q is it I do not know, one of these things. So, this may be this was q l and then the other probability is a p l which is probability is that v l equal to 1. So, these are the two probabilities, we wanted to track and we can p with two recursions and the first recursion is for what I wrote down. I think is for p l this require some thought but did not really derived it fully but assume if enough of you went back and thought about it. You would have found the answer why that is true. So, you can show this recursion is true q l minus 1 raise to the power w c minus 1 plus p times 1 minus 1 minus q l minus 1 raise to the power w c minus 1.

So, when can v l be 1? v l can be 1 is r i was received correctly received a 0 and then all the u l's agreed to be equal to 1 that is q l minus 1 raise to the power w c minus 1 or r i was received in error and all the q l's did not agreed to be 0. So, that is the logic for this formula and you get this, it works out you can the multiple ways of deriving this but eventually this is the idea and then the other recursion for q l in terms of p l is not too bad. That would be something like this 1 minus 2 p l raise to the power w r minus 1 divided by 2.

So, this is the classic x or formula that happens at the check node. So, this is the one, 1 minus the odd number of errors in w r minus 1 Bernoulli random variables. So, that is the idea in this, right? If you have w r minus 1 Bernoulli binary random variables with

probability p l of being equal to one, what is the probability that you will get an odd number of once you do the formula you will get this.

So, if you substitute q l minus 1 into this equation use this here with l minus 1, you will get a formula of this form p l equals some function which depends on w c and w r, right? That is parameterized by w c and w r of p and p l minus 1, you can write down this function. It is not a very complicated function but such that are so many terms and you have to write it out and I am being lazy here and not writing their expression of since it easy to write that expression out and this is the density evolution formula d e formula for gallager a over b s c, right?

So, of course in any iterative formula like this you need a starting point what is that starting point p 0, what is p 0, yeah I think it is good to think of p 1, right? So, we numbering iterations from one onwards. So, what is p 1 probability that v l equals 1 v 1 equals 1, what is v 1. First of all v 1 is simply r I, right? What is the probability that r i is 1 p, it is equal to, so that is the starting point you plug it in and you keep repeating this you will get the probability that v l equals 1 in the l th iteration is that.

So, this is the density evolution formula for Gallager, a decoding algorithm over BSC for the w c w r regular l d p c d codes. So, any ways there are numerous thinks to notice in this formula. So, one of the let us see what is missing. So, what is missing in this formula? What do you think should be there in the performance of a code or a probability of a error analysis of a code n is clearing missing, right?

So, that is something clearly not correct. Of course, this formula cannot be exact, if you have a finite block length n, this n has to show up to somewhere when you cannot just have a probability of a independent of n. You remember in m l probability, n clearly shows up, right? So, obviously n will show up somewhere and n is not showing up here. The reason why n is not showing up here is this formula is valid only when n becomes n is infinite. When is fine any finite then this formula will not be valid. Eventually there will be an l, may be for l equals 1 it will be valid. You know eventually there will be an l for which this formula will not be valid for any finite n. If n is very large you can make it be valid for any larger in case of that. That is definitely true that the crucial thing to understand is this part, right?

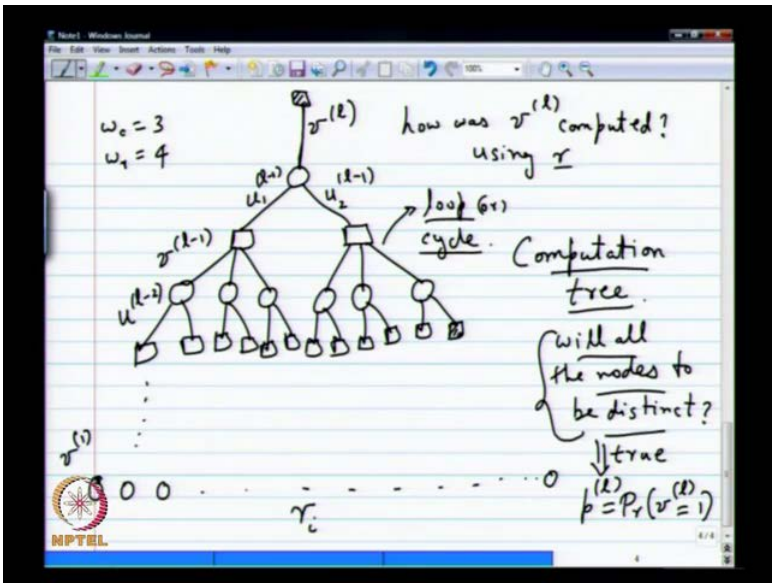
The all zero code word played a very enabling role as in we could just reduce the number of messages whose probability distributions we have to track just becomes 1. You know it is one random variable, you have to track all the messages are i i d in the first iteration itself, because of these all zero code word. If it do not have the all zero code word assumption then you cannot do it, right? Prob if you if you cannot track it very easily.

So, errors will be messages been zero will be another, for some errors message after has to be one and guys you cannot track those things very easily. So, it is just quit outly losed. So, all zero code word assumption can be justified without too much trouble for the BSC and the Gallegary decoding algorithm. You need a certain property called symmetry for that and we do not have too much time in this course to discuss that. Let us just say it is the other problem which is more interesting from a code construction point of view is just neighborhood assumption.

So, this will help you in actually generating a priority check matrix, they are good priority check matrix. So, to speak from the one example so that is why we should spend some time in understanding this. So, what we do mean by saying all the v u l's and v l's should have i i d distributions under what conditions.

On the neighborhoods that is on the graph or on the tanner graph is this assumption valid. That is the first question will ask we will try to answer that next. So, any way we will come back and look at this formula much more closely but I want to first settle this neighborhood assumption once for all and then will proceed with this formula.

(Refer Slide Time: 18:28)



So, let us look at u l. So, you have a particular list to that v l you have a particular bit node then there is an edge that is connected to this I am drawing it kind of in the upward direction is maybe you can just to make the thing little bit more clearer. Let us say this is sum v l, so the question I want to ask is how was this computed may be this is check node d r. How was v l computed, it was obviously computed using u l minus ones, where did I get these u l minus ones from from.

Let us say we just use let us say w c equals 3 and w r equals 4 for this picture, you can do for anything else also but the picture is just become more mercy. So, I will just use it for this small values and I can make a nice a picture. So, how many other will be there? Two other check nodes will be there and you would have got. You say u 1 l minus 1 and u 2 l minus 1 from here and then if I have to answer this question completely, I have to somehow make my way all the way down to the r i's, right? Because r i is what you got from the channel, eventually when you did all the iterations v l well ultimately be some functions of the entire vector r. It cannot be something else, right?

So, when I say ask this question how was v l computed, how was v l computed using r, thats the question I am asking, v l is definitely a function of r. What is that function explicitly? Suppose, you have to specify that how will you go about doing that. So, you turns out you have to kind of unwrap your tanner graph. This is the unwrapping that I am doing.

So, you start with this particular edge and v l was going on that particular edge. It was generated using the two u l's how where these u l's generated from v l minus 2 and where did the v l minus 2 come from a v l minus 1, may be a. So, v l minus 1 where did that come from? Three other bit nodes and we once again here v l minus 1 would have come from three other bit nodes and then what will be the next question I ask this is v l minus 1. How, where the v l minus 1's computed that would have been computed from u l minus 2 that would have come from two check nodes.

This is u l minus 2 then you will have a v l minus 2 and so on, till you get all the way down to just bit nodes which are just v 1 and that is equal to r r. So, v 1 is the last thing you have and that will have a whole bunch of bit nodes is that and that will be just r r 1, r 2. Whatever r i's anybody just write down r i because I don’t know which co efficient will be there.

So, these all are r i's, so such a tree is called a computation tree. You can call as a computation tree, if you want you can call it as neighborhood. So, there are several properties for this tree, you can say a lot of things about how many nodes that there will be at each level each depth. So, to speak for depth one, depth two, extra you can keep on in terms of w r and w c. You can answer that question. What is important is for any edge I pick from the bit node to the check node.

This tree will have the same structure for regular codes strong as the code is regular at any edge. I pick it will have the same structure same structure same kind of un folding will happen the what is what. So, the next thing is to keep in mind is that is fine. The same structure is fine but what about the actual nodes that show up there, show up all the way here well. They all be distinct that is the next question that we have to ask its very important but well all the nodes be distinct. This is this is quite important, why do you first of all want all the nodes to be distinct.

So, there are normally in due to reasons, why all the nodes well have to be distinct but a very statistical reason is that if all the nodes are distinct. Then what happens, my independents assumption is true, right? If all the nodes in this computation tree are distinct, then my formula for v l holds exactly. The density evolution formula for v l holds exactly p l that is probably v l equals 1 will be exactly the probability.

That happens only when all these nodes are distinct because there is no dependence, see all the r i's are distinct a independent here and the way I am combining them. As long as these nodes are distinct you will all come from independent observations and at every step independence is satisfied and the formula for p l is valid and if this is true this implies this is true p l is actually the probability that v l equals 1.

So, those are the two points that I wanted to observe and keep in mind for regular codes. The structure of this unfolding in this computation tree or the neighborhood of the particular edge, the remains the same for all edges and for a particular level l for a particular iteration l. If this entire computation tree has no reputations in any node, no check node repeats, no bit node repeats. If that is the situation then the density evolution formula for p l is exact p l is the probability that v l equals 1 and that is exact.

If you have the all the zero code word and you received r i corresponding to them is that hopefully that is convincing. It is not, it is not a very difficult argument, I did not write down the full proof but you can see how it works. It is all in depend, everything is working out very nicely, is that clear? Everybody is happy?

Now, you have to ask the question how can I construct a h so that this is satisfied, not just for one edge but for every single edge. So, that is the question you should ask from a construction point of view, because if you had to construct for let us say l equals 10. Let us say 10 iterations, you want to do and you want that 10 iterations to be accurate and your density evolution formula to exactly reflect. What happens then what should have happen in every neighborhood computation tree, that it ended for every edge up to depth 10.

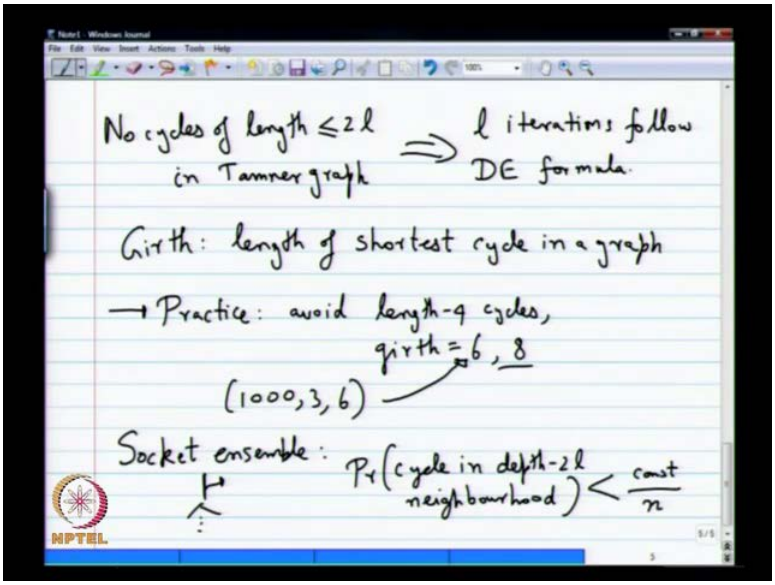
So, depth is like iteration 10, what should you get what should you not get, you should not get any reputations in the nodes. If you constructed the original graph in that fashion then you can be guranteed that the density evolution analysis will work. Otherwise there is no guranteed that it will work.

Now, that this seems like a over crazy requirement for a graph. So, what does it mean in terms of the graph. It turns out you can simplify the requirement, what happens when any nodes repeats in this tree. What does it mean from a graph point of view then anything can make you, suppose this check node repeats here what does it mean. It means like an you have a loop in your graph.

So, if this check node repeats here, it means this guy becomes loop in your graph loop or cycle. A cycle is a much more standard term, right? So, if I want my computation tree for iteration l to be free of any node reputations, what type of cycles should not be there in the graph. So, no cycle of length less than 2 l, it will turn out should not be there no cycle should be there. So, cycles of length less than 2 l should not be in the tanner graph.

If you can gurantee in that in your construction that there are no cycles are loops of length two times l. Why did I say two times l. So, you just look at this graph this graph will have a total depth of 2 l. It is not l right because l minus 1 is a appearing twice. So, everything will appear twice. So, it will have a length of 2 l. So, if you can avoid all cycles of length less than or equal to 2 l. This is the thing no cycles of length less than or equal to 12 must be there in your tanner graph, if that condition is satisfied then you can do l iterations with the density evolution formula accurately reflecting the probability of bit error, remember the probability of message error that is the key idea.

(Refer Slide Time: 28:05)



Let me summarize that write it down, no cycles of length less than or equal to 2 l in tanner graph implies l l iterations follow d e formula. That is the basic idea. In a way you are not doing anything wrong in the decoding. Everything you have to done up to that point l iterations is as it should be an not making any approximations of mistakes is that.

So, the length of a minimum cycle in a graph is called girth. So, girth is a technical term length of length of shortest cycle in a graph, remember in a bi partic graph. You can only

have even length cycles, you cannot have odd length cycles because if it has to come back, right? It has to go there and then come back again in any where for a loop. You will have a even number of length, even length and the if you have in the socket construction girth, two is possible, right? If because you can have multiple edges from between one bit node and one check node and this socket construction girth two is possible but if you avoid multiple edges, clearly the minimum girth possible is 4.

So, if you have girth four then how many iterations can you do? You can do 2, it 1 1 2, you cannot do right and you can do only one. Let us correct and also it is not correct actually. So, I think some 12 minus one missed missing it. If you have girth four, it turns out this an overlap of 2. So, you cannot even do two iterations. So, maybe there is like a 12 minus 1 know.

So, I guess this 12 minus 2 we have to look at right may be let we make sure if I have will see if I have any nice expressions written is I do not think to have it but I think that should be a trick, 12 minus 2 may be because this length this depth is what 12 minus 1, right? So, it has to be a 12 minus 1, so you should not have 12 minus 2, is that correct? Or is it 12 plus 2? So, let me thing about this carefully emulate in a little bit confused here because I am not sure that if for if recall correctly.

If you have girth 4 or if you have a cycle of length 4, you should not be able to do even one iteration. I think because see even that one iteration will cause some problem was you will have a you will have structure like that. I am sorry you can do one iterational just will travel but only when something it will. Oh there will be a problem.

So, I guess 12 was so right well is so I guess you can do one iteration. If you have girth 4 the one iteration is may be not too bad only when you combine by it and in sent it back you will have an violation of your probability density evolution formula, you are right? So, p 1 of course will be correct is just r i and then q 1 will also be correct only if p 2 will be wrong.

So, you are right. So, this is fine this is fine, I think 12 holes, so if you have girth 2, you can do one iteration girth 4, you can do one interation, you cannot do more than that. If you have girth six then you can do two iteration. So, even to do the second iteration when I said can do when you can up to two iterations. The density evolution formula will continue do hold for probability. That is what it means, not that when you can do any

number of iterations, no body stop you to iterations but density evolution formula may not hold that is all.

So, in typical constructions, so this is in practice. In practice what do you do is when you construct avoid length four cycles that is usually for most graphs. That you use girth will be 4 girth will be 6, I am sorry. So, one missing girth will be 6. So, there are some graphs that have girth 8 may be so but this is bit more rare. 6 is very common for most lengths and 6 can be easily ensured, you can write a very small simple program to make sure that there are no length 4 cycles. You can constructed very easily, it works quite easily but beyond girth 6, it is very difficult.

So, if you for a instance say if you want to construct a 1000 comma 3 comma 6 code. Usually you will get girth 6. It is difficult to go beyond girth 6, may be girth 8, you can do may be 10, may be but beyond 10 and all you cannot do much. So, you cannot construct it just takes. So, long may be they don't even existed. I don't know.

So, it is difficult to construct a these kind of things. So, usually people settle for girth 6, it turns out an practice even though the density evolution formula does not hold the decoder works relatively does not become very bad. For instance in the plot that I showed you yesterday, I showed you some plots. The graphs that I constructed have only girth 6. It do not have girth 8 or anything, only girth 6 and still you could see that the bitterer rate was falling quite steeply as I increase the block length. It was falling steeply.

So, what do you make of the density evolution formula them. That is the question you have to ask. So, basically it means that it is a good formula, even if the assumptions do not hold. So, as long as the if the remember threshold was computed using the density evolution formula. So, density evolution formula is good, even if the independent assumption does not hold.

So, you have to take that with the pinch of salt, it is not accurate any more but its approximate but it works is that. So, it is like saying I have a sine wave, you know you can never have a sine wave forever. You say always, say you have signed wave right well is very useful and practice and its great approximation. All your problems work out greatly great with that.

So, use think of the sign waves, so likewise think of density evolution in similar fashion is an approximation but it works very well in practice or may be the delta function. It is the another nice thing to mind yourself, sorry varies if I will come to that, I will come to that. I will come to that one minute.

So, but what you can show but you can show is the following. So, in the socket on sample in the socket on sample you can show some interesting results using counting and some very clever probability arguments. May be even more advance probability arguments. You can show that the probability of a cycle in a depth. Let us say a depth 12 neighborhood, this is less than some constant. That does not depend on n obviously divided by n.

So, you can show this. So, this of course involves some interesting arguments, the various ways of showing it you can since not too bad, the argument is not very bad, just not, just not providing it here. So, it is quite easy in not just socket on sumble event even other on some stumbles, you can show this. So, if you fix one particular edge and then look at its depth 12 neighborhood, what is the probability that you will have an reputation. What is the probability that it will have an cycle is less than some constant divided by n.

So, what is that mean as n becomes very, very large? This probability is going to 10 to 0. So, you won't have a cycle with high probability, all right? So, that is a good thing, so what is not very good thing about this formula, it would have been nice to have something larger than n. Why would you have something larger than n in the denominator, because the number of such edges is what you have n times w c edges and you want all the neighborhoods to have very low probability.

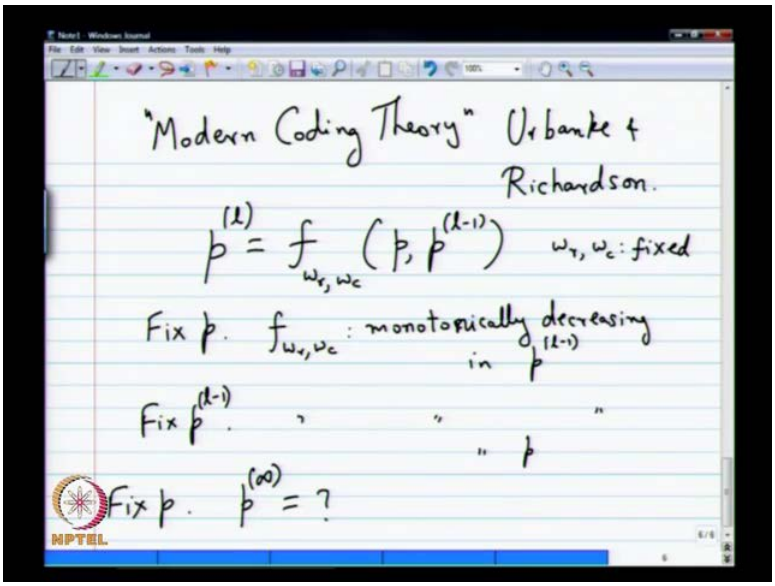
The probability that all the neighborhoods have a cycle should go to 0 but this thing does not quite guarantee that why because when you multiply by n, if you want to use a union bound in multiply by n, you get a constant and that constant many many case will be greater than 1. So, you won't get any information from this. This in equality for all the neighborhoods, in fact you can show that won't happen.

So, you have read more analysis to show more interesting results but this is interesting now, other less. So, every neighborhood with high probability will not have cycles but when you look at all of them together we have pre dimised guaranteed to have cycles.

That is why I said girth 6 on alls are now avoidable in this, in this constructions you will end up getting some girth that is the idea, all right? If you know, if don't if you haven't studied all of these asymptotics and probability what will not make sensitive at this point is every neighborhood has very low probability of not having a cycle but why is that all the above is together have a very good probability of having a cycle.

So, those are things that work and you know when it is going to its not really that low that is what it means. So, only constant by n it is not constant by n squared or n bar 3. So, is is not that low it is not low enough, only if it is very very low you can be sure about all these are the things. This is why I said this formulas valid asymptotical. So, another type of analysis which is again equally interesting is to fix n and let l go to infinitive, right? Then you cannot use the density evolutuion of this. So, you have to you have to do something else and that also is possible. People have done some work in that direction also. There are lots of work in this area like I said is it is a very, its matured very fast. So, people last 15 years work. So, have worked a lot on this and a very good reference for all this is modern coding theory.

(Refer Slide Time: 39:52)



This book modern coding theory by Urbanke and Richardson, it is a tuff book to read, I will warn you if you don't have a lot of works call mathematical maturity its a particularly more difficult book to read.

So, it will go, it will have it has a lot of technical details and you have to have some comfort with reading technical literature. If you can do that then this is a good book to read it. It has all in this information but otherwise the information is in papers. In papers are more difficult to read, that is also may be this is a good thing to read.

So, where are we now? So, we have this density evolution formula where is that here is the density evolution formula and we have a reasonable idea of when this formula has valid. So, its valid when valid when what there are two things that you have to keep in your mind in theory? When is it valid and in practice when is it valid in theory? It is valid when well is not really any where any time this is not for only for one particular bit.

So, to speak I mean it just one one edge, its valid not for all edges together. You can not take it like that, it is not valid in that case even a symptotically its not valid. If we take all edges together I did not practice. It seems to be a very good formula. So, it predicts the behavior for large n very accurately. That's the way to think about this, any question? I think all these must be totally new to you but nobody seems to be disturbed that we have studied enough things like this.

So, it is any questions, any comments cyclic nature talks about the dependence of v l with the same v are some other iteration and not able to get how it in the same v. I just drew the, I wrote down everything as v but it is not v really you know mean each edge is different. So, we have to think about it.

So, if there is reputations some edges will also repeat. So, other than that that is why I put u and u 2 here but then I cannot put v 3, v 4 means just comes mercy. It is difficult to write. Now, all right now let us given given all these conditions under which the density evolution formula is valid. Let us look at that formula more closely and see what we can say about it.

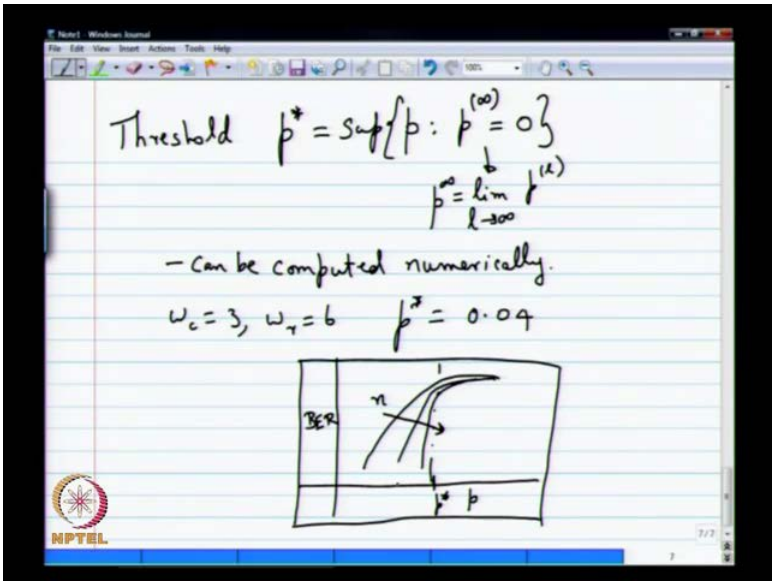
So, that what is we want to go do the formula, we have is of this of this form p l equals f w r w c p p l minus 1. So, this is the form that we have all right question, all right? So, what you do for this is you fix p w r and w c and of course w r and w c always fix. So, let me not say that w r and w c are fixed. So, you fix the w r and w c and then first think you can ask is you fix p and this function simply becomes a function of one variable p l minus 1 and you can do some very basic calculus to analyze that. For instance you

differentiate it, figure out if it positive or negative and you can show that this will be a monotonically. What can you expect as a function p of minus 1.

This function will be a monotonically decreasing, right? So, it will be a monotonically widgets. So, you fix p f w r w c is monotonically decreasing in p l minus 1. These are analytical properties that you can show and the other thing also is to fix p l minus 1 and then again you can show this is montonically decreasing in p, is that all right?

So, what's interesting is what is interesting now is you can ask an asymptotic question, you fix p and ask the question p infinitive, p 1 is equal to p itself, p 2 is going to be something lesser than or equal to p and p 3 will be lesser than or equal to the previous thing extra. So, it is that's going to be definitely true. The only thing that can happen now is p infinitive might either be 0 or some positive value less than ro equal to p or some non-zero value or 0. So, that is the question that you asked. So, what you define is the threshold is the following p star is actually.

(Refer Slide Time: 45:17)



So, you look at all those p such that p infinity is equal to 0. So, I am saying equal to obviously there is no equal to is here this is all limits. The way I should way I should, I write this down actually is limit as l tending to infinity of p l. So, I should write it simply writing in this p infinity equal to 0. So, that limit this is p infinity is clearly is a function only of p right for a fix p you will get a p infinity which might either be 0 or not.

So, I am going to look at the set of all p such that p infinity is equal to 0. For instance, p equal to 0 is clearly an entry in this set right p 0, when p infinitive will clearly be 0. It is nothing but can happen, no error at all. So, you was do iterations you want to go introduce new errors, well early, I will not going to make that situation worse. Let me say that and as is slowly increase p you will still get p infinity going to 0 for a while. It turns out you can show those things, you can show those properties for this iterations and in this set.

So, usually it is this is a set of real numbers and it will be an open set and you can not define what's called the maximum in an open set. So, what is the right word to use supremum, you have to use the supremum. So, I am going to say supremum. If you do not understand supremum, just take it as maximum any way these all these open sets do not exist. What you can compete with is only rational numbers and those things note even rational finite precision numbers, finite precision numbers. Clearly there will be a maximum.

So, think over it maximum of p such that p infinity goes to 0. So, that is called the threshold for this iteration and you can show such a threshold will exist. So, there are a analytical ways of showing that the threshold exist. So, it is not a, it is not just a vaccua's definition. It does exist and you can compute it and you can definitely compute it.

How do you compute this? Just you can as an numerically easily compute it, right? Write a MathLab program, you try a small p like an see what happens to p infinity then you increase p, increase p, increase p by some finite precision not fine infinite precision but in finite precision you can definitely compute it numerically. There is no problem, see when it does this.

So, this is again a very interesting MathLab program to write. It is a very simple program, it is a few lines of MathLab code, you can very easily write it and if you do that for w r w c equal to 3 and w r equal to 6. You will get p star to be around 0.04. So, let me see if all that what I told about density evolution. So, what does it mean p star is 0.04? So, we will identify what about the code word?

So, you are saying roughly correct things which will be and practice but really theoretically it does not mean much, you know. I mean just been the density evolution formulas going to change its behavior at this p star and if for a each bit for each edge the

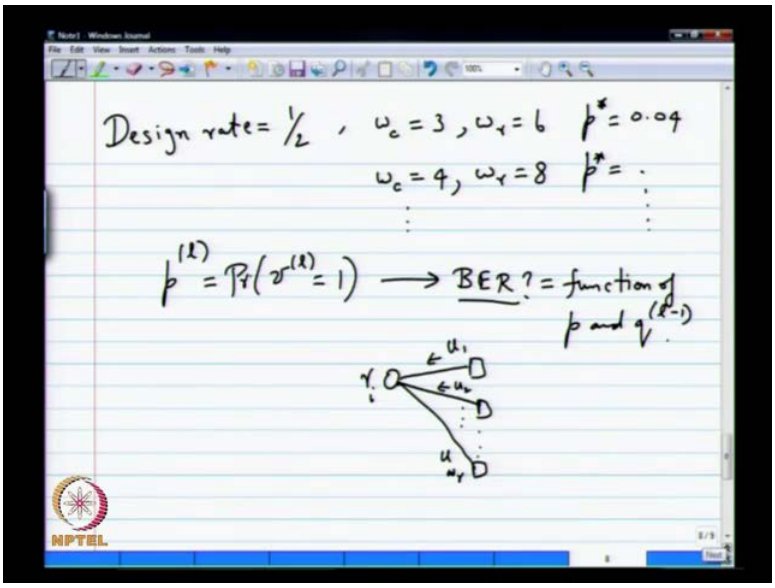
message that is going to go through that edge will be an error with probability tending to 0 for p less than p star and p greater than p zero but as a as a total. When you de-code what should happen?

The density evolution formula does not predict because it is not the valid in those cases but in practice we saw from the plot as n increases the bit error rate is clearly falling very very close to a 0.04. So, what you expect in practice is the following and that is captured very nicely by this p star. Maybe you cannot theoretically prove it in a very sound way but what will happen if you plot b r versus p is the will be the p star and as you keep increasing n what will happen something like this will happen. You are increasing n for the same 3 comma 6. This is the picture that I showed you last time also. This is just the rough version and version without any axis extra but I showed you the actual picture at that time also.

In last class, I showed you the actual picture. This is what we expect to happen which is true but what it means theoretically? You need to really read a lot and really figure out what it means theoretical but it is good analytical tool and it gives you a very nice picture in practice and that is true to make a full complete theoretical statement. You need to go into the cycles and what happens with that probability and so what is this and what is that extra.

So, it is not may able to make a very complete and concise theoretical statement but in practice this is eminently true. So, this what you expect, so one nice things that happens because of this is you do not have to do all these simulations to answer some design questions, okay?

(Refer Slide Time: 51:12)



So, for instance one design question you might ask is if you want my design right to be half, I will show the utility of the threshold. It is very nice to have it around. If you want my design rate to be half I can pick w c equal 3 w r equals 6 or I could pick up w c equal to 4 and w r equals 8 and so on, you can keep doing that, right? So, which one is better? Should I stop somewhere or should I keep doing it? Which one is the best? How do you answer that question? So, one nice way to answer that is you compute p star here. So, this works out, we 0.04 this p star if it is greater than 0.04 then this is a better thing. If it is lesser than 0.04 then you might will ask well settle for 3 comma 6 extra. This is how you used the threshold to answer designed questions rate of decrease rate of decrease.

So, how can you say that may be for a finite number of iterations one could we do not even know it is here. So, may be analyze the same way, it will work out the same way. I think p star is lesser, you will see even for finite number of iterations. If you increase w c and u r more chances of number of migrations being independent is reduce. Well it all depends on n n.

So, for this for the same n, fixed n for the fixed n I guess it all comes down to this kind of plot like this and definitely you compare. So, let us begin the quick recap case well looking at the at the decoding of let us say an n w c w r are regular l d p c code case. Once again when I say l d p c code, clearly it is not this just not define single code.

So, what I mean is I am thinking of a randomly generated, so the speak parity check matrix h with w c w r regular it right and n columns case and I am going to do this in some way. Then after that I am going to do a decoding for the code with priority check matrix h. So, clearly it is not unique but let us say we randomly pick one from this then what is the channel I am looking at?

I am at looking at case we spoke very briefly about this encoder essentially. What I said was you have to do you have to convert h to some kind of a systematic form then do do the encoding case. That is what we want to go do and that is not very complex because h is parse. Then let us say this goes through BSC with transition probability p and you get a received vector r.

We look that Galager a decoding algorithm to come up with a c cap and this can be this Gallager, a decoding algorithm can be very nicely described in the using the tanner graph of representation of with the parity check matrix. For essentially its iterative and in each iteration there are two steps iterative message passing algorithm that is the main, that is the main idea. There are two types of messages that are passed, the message from bit nodes to check nodes and then messages from check nodes to bit nodes.

So, there are two parts are two steps inside each iteration. The first step is basically looks something like that. So, particular the bit to check messages were called v o u v right v v. So, these messages the l basically depend on u 2 l minus 1, all the way down to u w c l minus 1. If all these guys agree and they are equal to b then this is going to be equal to b a, else whatever that was received by the from the channel.

So, what will happen in the first iteration is you do not have any of these things. So, clearly v l will be said to r i itself. Yes, that is what happens in the first iteration. All the messages will be equal to r i in the first iteration. The other iterations it will depend on the u that is received. What happens on the check node side is little bit simpler to describe what sent back here u l minus 1 is simply v 2 o.

So, u l may be right simply v 2 l plus v 3 l plus so on, till v w r l is so these are the incoming messages in the health iteration. So, these are the two steps, the messages are basically estimates of the messages. First of all flow along the edges and the edges connect to the bit node to check node every message is an estimate of that bit node which is connected to that h.

So, that is the main idea and the next idea that is kind of inbuilt here in this iterations. If you receive some messages from a node when you sent back another message to that same node you do not use that information. So, all whatever information is received from that node is not used. For the instance u 1 l minus 1 is not used in v l. So, same way here v 1 l is not used in generating u l. So, you do not use that same information.

So, that that basically messages have to be extrinsic, that is the idea messages are estimates but they should be extrinsic estimates. For that are being scented cannot be intrinsic or it cannot extrinsic. In the sense is that is should not be previously know to that note that. So, this two rules and the critical thing that is done for first thing, you can do is first of all simulate this and see what happens. I showed you a plot send and see that it does well and next thing is to do some analysis and for analysis like a said like a been always saying will do approximate asymptotic kind of analysis and the main idea is there is a density evolution.

So, what is density evolution? So, it is basically in density evolution, you will track the probability distribution of messages. This is the general definition of the density evolution. Density evolution is some algorithm processes which tracks the probability distribution of messages. So, messages of course in iterative message passing decoder. If you do not have an iterative message passing decoder, we do not have any messages to track, that’s no problem.

So, first of all I have been in the previous picture when I described the algorithm I simply said, I simply gave a very simple description with respect to one bit node. Of course, in the tanner graph you have n bit nodes and n times w c edges. So, each edges now carrying a message. First question we should ask is which messages probability distribution is being tracked here? Really, I mean if you want to track the probability distribution of every single message you are not going to able to do it. You have to hope or resume that all of them are according to some same distribution.

So, you need some simplifying assumption like that. So, that you can track one density a suppose to some n times w c density which are impossible to do. So, that is the first assumption that you need and for that you need these all zero code word situation. Once you assume that all zero code word is transmitted all the r i's have the same distribution, right?

Once you make the all zero code word assumption, what is the distribution of r i is equal with 0 with probability 1 minus p and lets 1 with probability p and its i i d right identically distributed. For all i a, r 1 r 2 everything as the same distribution. So, clearly the first message that is going from the bit nodes to check nodes.

You can just think of message on any edge it will have the same distributions. It is enough if you track that now as it terms out. If you make some further assumptions you can assume the following. You need to basically assume in any situation like this. In any situation like this the u 2 l minus 1 through u w c l minus 1 are in fact u 1 l minus 1 through u w c l minus 1 are also all i i d k.

So, basically the assumption that you need is this is for the first message, for even subsequent iterations you need to assume that all u l and v l have i i d distributions. What do you mean by all v l and u l u l represents the message that flows from the check node to the bit node, that's happening on all edges. So, that's what I mean by all, so if you look at all these u l's you have to kind the resume that they are identical and independently distributed. Same thing for v l, messages are flowing from left to right. Once again will assume they all are identical and independent.

So, once you make these assumptions then the analysis is much easier but what you need further for these assumptions all zeros, not the only the thing that is needed. In fact, you need some further assumptions for this. So, I will tell you that what you need for that later. So, you need some neighborhood assumptions. We will see this later but for now we will assume this and proceed with the analysis and see what we get some neighborhood assumptions are needed. We will see this later in a little while but if you assume that this is true then you can do density evolution relatively in a painful pain free manner.

So, what does the equation, there are two equations were remember right first equation was there are two things. We track the first thing, we track is something that I called is p l which is probability that u l equals 1 u l was q, is it I do not know one of these things. So, this may be this was q l and then the other probability is a p l which is probability is that v l equal to 1.

So, these are the two probabilities we wanted to track and we can p with two recursions and the first recursion is for what I wrote down, I think is for p l. This require some

thought but did not really derived it fully but assume if enough of you went back and thought about it. You would have found the found the answer why that is true.

So, you can show this recursion is true q l minus 1 raise to the power w c minus 1 plus p times 1 minus 1 minus q l minus 1 raise to the power w c minus 1. So, when can v l be 1? v l can be 1 is r i was received correctly received a zero and then all the u l's agreed to be equal to 1, that is q l minus 1 raise to the power w c minus 1 or r i was received in error and all the q l's did not agreed to be 0.

So, that is the logic for this formula and you get this it works out, you can the multiple ways of deriving this but eventually this is the idea and then the other recursion for q l in terms of p l is not too bad. That would be something like this one, minus 1 minus 2 p l raise to the power w r minus 1 divided by 2.

So, this is the classic x or formula that happens at the check node. So, this is the one, 1 minus the odd number of errors in w r minus 1 Bernoulli random variables. So, that is the idea in this right if you have w r minus 1 Bernoulli binary random variables with probability p l of being equal to 1. What is the probability that you will get an odd number of once. You do the formula, you will get this.

So, if you substitute q l minus 1 into this equation use this here with l minus 1, you will get a formula of this form p l equals some function which depends on w c and w r, right? That is parameterized by w c and w r of p and p l minus 1. You can write down this function. It is not a very complicated function but such that are so many terms and you have to write it out and I am being lazy here and not writing their expression.

Since it easy to write that expression out and this is the density evolution formula d e formula for Gallager a over BSC, right? So, of course in any iterative formula like this you need a starting point. What is that starting point p 0, what is p 0? I think it is good to think of p 1. So, we numbering iterations from 1 onwards, so what is p 1 probability that v l equals 1, v 1 equals 1. What is v 1, first of all v 1 is simply r i, right? What is the probability that r i is 1 p. It is equal to, so that is the starting point, you plug it in and you keep repeating this. You will get the probability that v l equals 1 in the l th iteration is that.

So, this is the density evolution formula for Gallager, a decoding algorithm over BSC for the w c w r regular l d p c d codes. So, any ways there are numerous things to notice in this formula. So, one of the let us see what is missing. So, what is missing in this formula? What do you think should be there in the performance of a code or a probability of a error analysis of a code n is clearing missing, right?

So, that is something is clearly not correct. Of course, this formula cannot be exact, if you have a finite block length n this n has to show up to somewhere when you cannot just have a probability of a independent of n. You remember in m l probability n clearly shows up, right? So, obviously n will show up somewhere and n is not showing up here.

The reason why n is not showing up here is this formula is valid only when n becomes n is infinite. When is fine any finite then this formula will not be valid. Eventually there will be an l may be for l equals 1. It will be valid, you know eventually there will be an l for which this formula will not be valid.

For any finite n, if n is very very large you can make it be valid for any larger. In case of that definitely true, that the crucial thing to understand is this part, right? The all zero code word played a very enabling role as in we could just reduce the number of messages whose probability distributions we have to track just becomes 1. You know it is one random variable. You have to track all the messages are i i d in the first iteration itself because of these all zero code word.

If it do not have the all zero code word assumption then you cannot do it right prob. If you cannot track it very easily, some errors will be messages been zero will be another. For some errors message after has to be one and guys. You cannot track those things very easily. So, it is just quit outly losed, so all zero code word assumption can be justified without too much trouble for the BSC and the Gallegary decoding algorithm.

You need a certain property called symmetry for that and we do not have too much time in this course to discuss that but let us just say it is the other problem which is more interesting from a code construction point of view is just neighborhood assumption. So, this will help you in actually generating a priority check matrix. They are good priority check matrix.

So, to speak from the on example, so that is why we should spend some time in understanding this. So, what we do mean by saying all the v u l's and v l's should have i i d distributions under what conditions. On the neighborhoods that are on the graph or on the tanner graph is this assumption valid. That is the first question will ask we will try to answer that next.

So, anyway we will come back and look at this formula much more closely but I want to first settle this neighborhood assumption. Once for all and then will proceed with this formula. So, let us look at u l, so you have a particular list to that v l, you have a particular bit node then there is an edge that is connected to this. I am drawing it kind of in the upward direction is maybe you can just to make the thing little bit more clearer.

Let us say this is sum v l, so the question I want to ask is how was this computed may be this is check node d r how was v l computed. It was obviously computed using u l minus ones, where did I get these u l minus ones from let us say we just use let us say w c equals 3 and w r equals 4. For this picture you can do for anything else also but the picture is just become more mercy.

So, I will just use it for this small values and I can make a nice a picture. So, how many other will be there two other check nodes will be there and you would have got you say u 1 l minus 1 and u 2 l minus 1 from here and then if I have to answer this question completely I have to somehow make my way all the way down to the r i's right because r i is what you got from the channel. Eventually when you did all the iterations v l well ultimately be some functions of the entire vector r. It cannot be something else, right?

So, when I say ask this question how was v l computed, how was v l computed using r, that is the question I am asking, v l is definitely a function of r. What is that function explicitly? Suppose you have to specify that how will you go about doing that.

So, you turns out you have to kind of unwrap your tanner graph. This is the unwrapping that I am doing so you start with this particular edge and v l was going on that particular edge. It was generated using the two u l's how where these u l's generated from v l minus 2 right and where did the v l minus 2 come from a v l minus 1, may be a. So, v l minus 1 where did that come from, three other bit nodes.

We once again here v l minus 1 would have come from three other bit nodes and then what will be the next question. I ask this is v l minus 1, how where the v l minus one's computed? That would have been computed from u l minus 2, that would have come from two check nodes.

This is u l minus 2, then you will have a v l minus 2 and so on till you get all the way down to just bit nodes which are just v 1 and that is equal to r r. So, v 1 is the last thing you have. That will have a whole bunch of bit nodes is that and that will be just r r 1, r 2 whatever r i's anybody just write down r i because I do not know which co efficient will be there. So, these all are r i's, so such a tree is called a computation tree. You can call as a computation tree. If you want you can call it as neighborhood. So, there are several properties, for this tree you can say a lot of things about how many nodes that that there will be at each level each depth.

So, to speak for depth one depth two extra you can keep on in terms of w r and w c. You can answer that question. What is important is for any edge I pick from the bit node to the check node. This tree will have the same structure for regular codes, strong as the code is regular at any edge. I pick it will have the same structure, same structure same kind of unfolding will happen.

So, the next thing is to keep in mind is that is fine, the same structure is fine but what about the actual nodes that show up there show up all the way here. Well, they all be distinct that is the next question that we have to ask. It is very important but well all the nodes be distinct. This is quite important, why do you first of all want all the nodes to be distinct.

So, there are normally in due to reasons why all the nodes well have to be distinct but a very statistical reason is that if all the nodes are distinct then what happens. My independents assumption is true, right? If all the nodes in this computation tree are distinct then my formula for v l holds exactly the density evolution formula for v l holds exactly p l. That is probably v l equals 1 will be exactly the probability.

That happens only when all these nodes are distinct because there is no dependence see all the r i's are distinct a independent here and the way I am combining them. As long as these nodes are distinct you will all come from independent observations and at every

step independence is satisfied and the formula for p l is valid and if this is true this implies this is true p l is actually the probability that v l equals 1.

So, those are the two points that I wanted to observe and keep in mind for regular codes. The structure of this unfolding in this computation tree or the neighborhood of the particular edge, the remains the same for all edges, and for a particular level l for a particular iteration l. If this entire computation tree has no reputations in any node, no check node repeats no bit node repeats. If that is the situation then the density evolution formula for p l is exact p l is the probability that v l equals 1 and that is exact.

If you have the all the zero code word and you received r i corresponding to them is that hopefully that is convincing. It is not a very difficult argument. I did not write down the full proof but you can see how it works. It is all in depend everything is working out very nicely, is that clear? Everybody is happy?

Now, you have to ask the question how can I construct a h? So, that so that this is satisfied not just for one edge but for every single edge. So, that is the question you should ask from a construction point of view because if you had to construct for let us say l equals 10. Let us say ten iterations you want to do and you want that ten iterations to be accurate and your density evolution formula to exactly reflect. What happens then what should have happen in every neighborhood computation tree that it ended for every edge up to depth 10.

So, depth is like iteration ten, what should you get what should you not get? You should not get any reputations in the nodes. If you constructed the original graph in that fashion then you can be guranteed that the density evolution analysis will work. Otherwise, there is no guarantee that it will work.

Now, that this seems like a over crazy requirement for a graph. So, what does it mean in terms of the graph. It turns out you can simplify the requirement. What happens when any nodes repeats in this tree what does it mean from a graph point of view. Then anything can make you. Suppose, this check node repeats here, what does it mean? It means like you have a loop in your graph.

So, if this check node repeats here, it means this guy becomes loop in your graph loop or cycle. A cycle is a much more standard term, right? So, if I want my computation tree for

iteration l to be free of any node reputations, what type of cycles should not be there in the graph. So, no cycle of length less than 2 l, it will turn out should not be there, no cycle should be there.

So, cycles of length less than 2 l should not be in the tanner graph. If you can guarantee in that in your construction that there are no cycles are loops of length two times l. Why did I say two times l? So, you just look at this graph. This graph will have a total depth of 2 l, its not l right because l minus 1 is a appearing twice. So, everything will appear twice. So, it will have a length of 2 l. So, if you can avoid all cycles of length less than or equal to 2 l. This is the thing no cycles of length less than or equal to 12 must be there in your tanner graph.

If that condition is satisfied then you can do l iterations with the density evolution formula accurately reflecting the probability. Remember the probability of message error that is the key idea. Let me summarize that write it down. No cycles of length less than or equal to 2 l in tanner graph implies l l iterations follow d e formula. That is the basic idea, in a way you are not doing anything wrong. In the decoding everything you have to done up to that point l iterations is as it should be an not making any approximations of mistakes is that.

So, the length of a minimum cycle in a graph is called girth. So, girth is a technical term length of length of shortest cycle. Shortest cycle in a graph remember in a bi partic graph. You can only have even length cycles, you cannot have odd length cycles because if it has to come back right it has to go there and then come back again in any where for a loop. You will have a even number of length, even length and the if you have in the socket construction girth two is possible, right?

If because you can have multiple edges from between one bit node and one check node and this socket construction girth two is possible but if you avoid multiple edges, clearly the minimum girth possible is 4. So, if you have girth 4 then how many iterations can you do? You can do 2, it 1, 1 yeah 2 you cannot do right and you can do only one. Let us correct and also it is not correct actually.

So, I think some 12 minus 1 missed, missing it if you have girth 4, it turns out this an overlap of 2. So, you cannot even do two iterations. So, maybe there is like a 12 minus 1 know. So, I guess this 12 minus 2 we have to look at right may be, let we make sure if I

have will see if I have any nice expressions written is I do not think to have it but I think that should be a trick 12 minus 2 may be because this length this depth is what 12 minus 1 right starting.

So, it has to be a 12 minus 1, so you should not have 12 minus 2, is that correct? Or is it 12 plus 2? So, let me think about this carefully, emulate in a little bit confused here because I am not sure that if for if recall correctly. If you have girth 4 or if you have a cycle of length 4. You should not be able to do even one iteration I think because see even that one iteration will cause some problem, was you will have a you will have structure like that I am sorry. You can do one iteration, just will travel but only when something it will. There will be a problem.

So, I guess 12 was so right well so I guess you can do one iteration if you have girth 4. The one iteration is may be not too bad only when you combine by it and in sent it back. You will have an violation of your probability density evolution formula, you are right. So, p 1 of course will be correct is just r i and then q 1 will also be correct only if p 2 will be wrong.

So, you are right, so this is fine. This is fine I think 12 holes. So, if you have girth 2, you can do one iteration girth 4 you can do one iteration. You cannot do more than that. If you have girth 6 then you can do two iterations.

So, even to do the second iteration when I said can do when you can up to two iterations, the density evolution formula will continue, do hold for probability of that is what it means, not that when you can do any number of iterations. Nobody stop you to iterations but density evolution formula may not hold that's all. So, in typical constructions, so this is in practice. In practice what do you do is when you construct avoid length four cycles that is usually for most graphs. That you use girth will be 4 girth will be 6, I am sorry. So, one missing girth will be 6, so there are some graphs that have girth 8 may be so but this is bit more rare, 6 is very common for most lengths and 6 can be easily ensured. You can write a very small simple program to make sure that there are no length 4 cycles. You can constructed very easily, it works quite easily but beyond girth 6 it is very difficult.

So, if you for a instance say if you want to construct a 1000 comma 3 comma 6 code, usually you will get girth 6. It is difficult to go beyond girth 6 may be girth 8, you can do

may be 10 may be but beyond 10 and all you cannot do much. So, you cannot construct it just takes so long may be they do not even existed, I do not know.

So, it is difficult to construct a these kind of things. So, usually people settle for girth 6. It turns out an practice even though the density evolution formula does not hold the de coder works relatively does not does not become very bad. For instance in the plot that I showed you yesterday, I showed you some plots. The graphs that I constructed have only girth 6. It do not have girth 8 or anything. Only girth six and still you could see that the bitterer rate was falling quite steeply as I increase the block length, it was falling steeply.

So, what do you make of the density evolution formula them, that is the question you have to ask. So, basically it means that it is a good formula even if the assumptions do not hold. So, as long as the remember threshold was computed using the density evolution formula. So, density evolution formula is good even if the independent assumption does not hold.

So, you have to take that with the pinch of salt. It is not accurate any more but its approximate but it works is that so it is like saying I have a sign wave. You know you can never have a sign wave forever, you say always say you have signed wave right well is very useful and practice and its great approximation. All your problems work out greatly great with that.

So, use think of the sign waves, so likewise think of density evolution in similar fashion is an approximation but it works very well in practice or may be the delta function. It is the another nice thing to mind yourself, sorry varies. If I will come to that one minute. So, but what you can show but you can show is the following.

So, in the socket on sample in the socket on sample you can show some interesting results using counting and some very clever probability arguments may be even more advance probability arguments. You can show that the probability of a cycle in a depth. Let us say a depth 12 neighborhood, this is less than some constant. That does not depend on n, obviously divided by n.

So, you can show this so this of course involves some interesting arguments, the various ways of showing it you can since not too bad the argument is not very bad just not just not providing it here. So, it is quite easy in not just socket on sumble event even other on

some stumbles, you can show this. So, if you fix one particular edge and then look at its depth 12 neighborhood what is the probability that you will have an reputation. What is the probability that it will have an cycle is less than some constant divided by n.

So, what is that mean as n becomes very large this probability is going to 10 to 0. So, you would not have a cycle with high probability all right. So, that is a good thing, so what is not very good thing about this formula, it would have been nice to have something larger than n. Why would you have something larger than n in the denominator because the number of such edges is what you have n times w c edges and you want all the neighborhoods to have very low probability.

The probability that all the neighborhoods have a cycle should go to 0 but this thing does not quite guarantee that. Why because when you multiply by n, if you want to use a union bound in multiply by n, you get a constant and that constant many case will be greater than 1. So, you will not get any information from this. This inequality for all the neighborhoods. In fact you can show that that will not happen. So, you have read more more nuvanst analysis to show more interesting results but this is this is interesting.

Now, other less so every neighborhood with high probability will not have cycles but when you look at all of them together we have pre dimised guaranteed to have cycles. That is why I said girth 6 on alls are now avoidable. In this in this constructions you will end up getting some girth, that is the idea, all right? If you know, if do not if you have not studied all of these asymptotic and probability what will not make sensitive. At this point is every neighborhood has very low probability of not having a cycle but why is that all the above is together have a very good probability of having a cycle.

So, those are things that work n you know when it is going to it is not really that low. That is what it means. So, only constant by n, it is not constant by n squared or n bar 3. So, is not that low it is not low enough only if it is very low you can be sure about all these are the things. This is why I said these formulas valid as asymptotical. So, another type of analysis which is again equally interesting is to fix n and let l go to infinitive, right? Then you cannot use the density evolution of this.

So, you have to you have to do something else and that also is possible. People have done some work in that direction also. There are lots of works in this area like a said is it is a very matured, very fast. So, people last 15 years work, so have worked a lot on this

and a very good reference for all this is modern coding theory, this book modern coding theory by Urbanke and Richardson. It is a tuff book to read, I will warn you, if you do not have a lot of works call mathematical maturity. It is a particularly more difficult book to read.

So, it will go it will have it has a lot of technical details and you have to have some comfort with reading technical literature. If you can do that then this is a good book to read it. It has all in this information but otherwise the information is in papers are more difficult to read. That is also, may be this is a good thing to read. So, where are we now? So, we have this density evolution formula where is that here is the density evolution formula and we have a reasonable idea of when this formula has valid.

So, it is valid when valid when what there are two things that you have to keep in your mind in theory, when is it valid and in practice when is it valid in theory. It is valid when well is not really any where any time, this is not for only for one particular bit. So, to speak I mean it just one edge. It is valid not for all edges together, you cannot take it like that. It is not valid in that case even a asymptotically. It is not valid if we take all edges together I did not practice, it seems to be a very good formula.

So, it predicts the behavior for large n very accurately. That is the way to think about this any question. I think all these must be totally new to you but nobody seems to be disturbed that we have studied enough things like this. So, it is any questions any comments cyclic nature talks about the dependence of v l with the same v are some other iteration and not able to get how it in the same v. I just drew the I wrote down everything as v but it is not v really you know mean each edge is different.

So, we have to think about it so if there is reputations some edges will also repeat. So, other than that is why I put u and u 2 here but then I cannot put v 3, v 4 means just comes mercy. It is difficult to write now, all right? Now, let us given all these conditions under which the density evolution formula is valid. Let us look at that formula more closely and see what we can say about it. So, that what we want to do the formula we have is of this of this form p l equals f w r w c p p l minus 1.

So, this is the form that we have all right question all right, so what you do for this is you fix p w r and w c and of course w r and w c always fix. So, let me not say that w r and w c are fixed. So, you fix the w r and w c and then first think you can ask is you fix p and

this function simply becomes a function of one variable p l minus 1 and you can do some very basic calculus to analyze. That for instance you differentiate it figure out if it positive or negative and you can show that this will be a monotonically what can you expect as a function p of minus 1. This function will be a monotonically decreasing, right?

So, it will be a monotonically widgets, so you fix p f w r w c is monotonically decreasing in p l minus 1. These are analytical properties that you can show and the other thing also is to fix p l minus 1 and then again you can show. This is monotonically decreasing in p is that all right. So, what is interesting is what is interesting now is you can ask an asymptotic question you fix p and ask the question p infinitive p 1 is equal to p itself p 2 is going to be something lesser than or equal to p and p 3 will be lesser than or equal to the previous thing extra. So, it is that is going to be definitely true.

The only thing that can happen now is p infinitive might either be 0 or some positive value less than r o equal to p or some non zero value or 0. So, that is the question that you asked. So, what you define is the threshold is the following p star is actually so you look at all those p such that p infinity is equal to 0. So, I am saying equal to obviously there is no equal to is here this is all limits. The way I should way I should I write this down actually is limit as l tending to infinity of p l.

So, I should write it simply writing in this p infinity equal to 0. So, that limit this is p infinity is clearly is a function only of p right for a fix p. You will get a p infinity which might either be 0 or not. So, I am going to look at the set of all p such that p infinity is equal to 0. For instance p equal to 0 is clearly an entry in this set right p 0, when p infinitive will clearly be 0. It is nothing but can happen no error at all.

So, you was do iterations you want go to introduce new errors well early I will not going to make that situation worse. Let me say that and as is slowly increase p, you will still get p infinity going to 0. For a while it turns out you can show those things, you can show those properties for this iterations and in this set.

So, usually it is this is a set of real numbers and it will be an open set and you cannot define, what is called the maximum in an open set. So, what is the right word to use supermom, you have to use the supremum. So, I am going to say supremum, if you don understand supremum just take it as maximum any way these all these open sets do not

exist. What you can compute with is only rational numbers and those things note even rational finite precision numbers finite precision numbers clearly there will be a maximum.

So, think over it maximum of p such that p infinity goes to 0. So, that is called the threshold for this iteration and you can show such a threshold will exist. So, there are a analytical ways of showing that the threshold exist.

So, it is not just a vaccua's definition, it does exist and you can compute it and you can definitely compute it. How do you compute this, just you can as an numerically easily compute it right write a matlab program. You try a small p like an see what happens to p infinity, then you increase p increase p increase p by some finite precision not fine infinite precision but in finite precision. You can definitely compute it numerically there is no problem see when it does this.

So, this is again a very interesting matlab program to write it is a very simple program. It is a few lines of matlab code, you can very easily write it and if you do that for w r w c equal to 3 and w r equal to 6, you will get p star to be around 0.04. So, let me see if all that what I told about density evolution. So, what does it mean p star is 0.04, so we will identify what about the code word of so you are saying roughly correct things which will be and practice but really theoretically. It does not mean much you know I mean just been the density evolution formulas going to change its behavior at this p star.

If for a each bit for each edge the message that is going to go through that edge will be an error with probability tending to 0 for p less than p star and p greater than p 0 but as a as a total when you de-code what should happen. The density evolution formula does not predict because it is not the valid in those cases but in practice we saw from the plot as n increases the bit error rate is clearly falling very close to a 0.04.

So, what you expect in practice is the following and that is captured very nicely by this p star, may be you cannot theoretically prove it in a very sound way but what will happen if you plot b r versus p is the will be the p star and as you keep increasing n. What will happen? Something like this will happen. You are increasing n for the same 3 comma 6. This is the picture that I showed you last time also. This is just the rough version and version without any axis extra but I showed you the actual picture at that time. Also in last class I showed you the actual picture.

This is what we expect to happen which is true but what it means theoretically? You need to really read a lot and really figure out what it means theoretical but it is good analytical tool and it gives you a very nice picture in practice and that is true. To make a full complete theoretical statement you need to go into the cycles and what happens with that probability. So, what is this and what is that extra. So, it is not may able to make a very complete and concise theoretical statement but in practice, this is eminently true.

So, this what you expect, so one nice things that happens because of this is you do not have to do all these simulations to answer some design questions. So, for instance one design question you might ask is if you want my design right to be half I will show the utility of the threshold. It is very nice to have it around, if you want my design rate to be half I can pick w c equal 3, w r equals 6 or I could pick up w c equal to 4 and w r equals 8 and so on, you can keep doing that, right?

So, which one is better? Should I stop somewhere or should I keep doing it? Which one is the best? How do you answer that question? So, one nice way to answer that is you compute p star here. So, this works out we 0.04, this p star if it is greater than 0.04 then this is a better thing. If it is lesser than 0.04 then you might will ask well settle for 3 comma 6 extra.

This is how you can answer designed questions. This is how can you say that you do not even know it is here. So, may be you have to analyze it but it will work out the same way. I think p star is lesser, you will see even for finite number of iterations it will be somewhere case. If it increase w c and u r l more chances of number of iterations being independent is a reduce, well it all depends on n.

So, for the same n fixed n for the fixed n, so I guess it all comes down to a this kind of a plot. I mean so you what you do is you make a plot like this and definitely you compare. You will get your answer but p star is a good starting point. I mean if its lower than its very likely to be if p star for this is higher then this is likely to be a better code and you will see when you actually, you will simulate it as long as you take care to avoid girth 4, girth 4, girth 4. For some reason is bad as long as you have girth 6 it is density evolution formula. At least for bit error rates of 10 power minus 6, minus 7, minus 8, may be even minus 9.

It is may be if you go below that there is see something but 10 power minus 9 is so low were in many wireless applications. At least nobody looks at such low be as 10 power minus 5 is good enough, right? So, if you want only 10 power minus 5, you go to block length of 1000's or 10000's and then you compare these things. You will see the density evolution formula is really good. When it is not bad even iteration, after iteration you will see it captures the bit error rate very nicely.

So, it is not bad, one thing I should quickly point out. Remember that p l is probability that v l is in error or is in one. It is one, it is not the same as bit error rate, right? It is only message error rate in your message passing decoder. How will you go from this bit to error rate final decision make.

So, you have to use the final decision magic making logic, what was of my final decison making logic. It did not involve v l at all it involved q l because it involved only the use, right? So, you have to look at the bit node, all the messages it receives all the use and the r i's I am going to take a majority decision kind of thing, right?

So, that is what that is my final decision, so it involves q l and then the p itself but of course if p l tends to 0, q l will also tend to 0. So, everything is it works out quite nicely. So, but this is the formula that you use when you cannot just directly say p l is bit error rate bit error rate will be some function of q l. This will be some function of p and q l q l minus may be, right? So, you remember what you we do so you have you have w r messages that are coming from here, right? u 1, u 2, u w r then you have r i itself. What is my decision some kind of majority of these things. So, you so that is what will determine this function, it is a bit more complicated than just p l itself. If you compute bit error rate using this, it is pretty good, I mean for the large n and I say large n n like 10000 and all. It is a very good formula.

So, this utility has a designed to lets a very vital for threshold. This is the major selling point for threshold, you can design what is known as degree distribution. So, what is the degree distribution of a tanner graph, what are the left degrees, what are the right degrees, right so far you are looking at just regular.

So, that is really nothing much to describe but even here threshold is a very useful measure for a particular rate. You look at a particular rate and decide which is the best

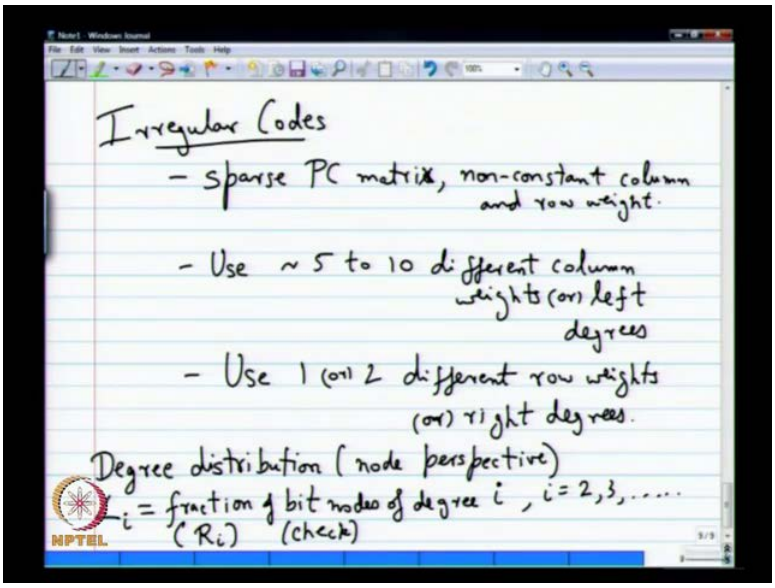
code that you want to do, then after that you have to construct the parity check matrix, for that you have to use girth criteria.

So, we have to try maximize your girth that is bit more complicated then you can write that. Then you run the Gallagery you will get your answer. So, once again all these are good programs to write. If you have the time then you should write the program for computing the threshold. For Gallegary given w r and w c figuring out which one is the best and then write a program for generating a parity check matrix and with these constraints then write a program for be a encoding. Then you write a program for decoding. Then you have the entire simulation ready with you. You can generate these kind of these plots if you have all.

So, this is kind of point like a point where we go to make a bit of deviation. So, in case if you have any questions or comments on is this is a good time to ask any questions is that feel this is good. So, you see the other problem right. So, the other problem I was talking about is a problem of coming up with exam questions.

So, you cannot come up with any exam question in this area, right? What kind of exam question can I ask? The only question I can ask is write down an algorithm for doing something like this and nothing else. I can ask its very difficult to design any good coding questions. Of course, you can ask analytical questions calculus questions, I can ask show that this is a increasing function of that but I am assuming you seen it somewhere. You would know how to show some thing is an increasing function, right? So, this is stuff is like that does what can be asked. We cannot ask a real good coding question, may be may be if I come up with the question then you will have to answer that.

(Refer Slide Time: 58:34)



So, what we want to go to see next is this idea of irregular code. This is where the threshold really helps out here and here again the notion of the threshold is now will become more approximate. Yes you think when you make it more approximate it may not hold. So, rule is a very good tool, so ends up with a really good tool for designing codes with finite bit error rate that is not finitely low bit error rate it 10 power minus 5, 10 power minus 6 and all. If you want to show like asymptotically low bit error rates like orbitally low bit error rate then you cannot do it with threshold but if you want to show just finite low bit error rate like 10 power minus 5 minus 6 and all that you can do by simulation, right?

If you want to show 10 power minus 5 bit error rate in simulate and sure and so for those things this irregular codes are really good. So, in fact most standards use the regular codes. They do not use regular codes, so what are irregular l d p c codes sparse parity check matrix that holds always but then what number of column and row weight is not constant column and row weight. So, by having said that usually you cannot allow all kinds of column weights. You know I mean of course the row wise can be anywhere from 0 to n, right? This block length you cannot allow every single row weight and then you cannot design it. You cannot analyze it, nothing can be done.

So, usually what is done is there will be something like, let us say I think most standards do even lesser but usually let us say 10 or 20 different column bits and then in fact for

row wise. It will either be one or two, that is all that kind of people have kind of settled down to that you know. When you can think of other options also not saying other options are bad but usually in practice people use about use roughly.

Let us say even 5 to maximum 10 different columns rates, remember? This columns weights are also what are also left degrees in your tanograph, if you think of the matrix then its column weight in a tanograph. It is degree of the left nodes, it is called left degrees. Usually use 5 to 10, I am just giving you a figure of role of thumb here. You can also use 20 if you like but it is just a more gets more complicated and for column weight.

You use either 1 or 2 for row weight, I am sorry, 1 or 2 different row weights or right degrees. So, this is just to simplify this vast set of possibilities. If you do not put it down to this then there are so too many possibilities, you cannot make sense of all of them together. So, we must make some simplifying assumptions in restriction the set, even though it is irregular. It is not that irregular minutes moderately irregular that is I have to think about it.

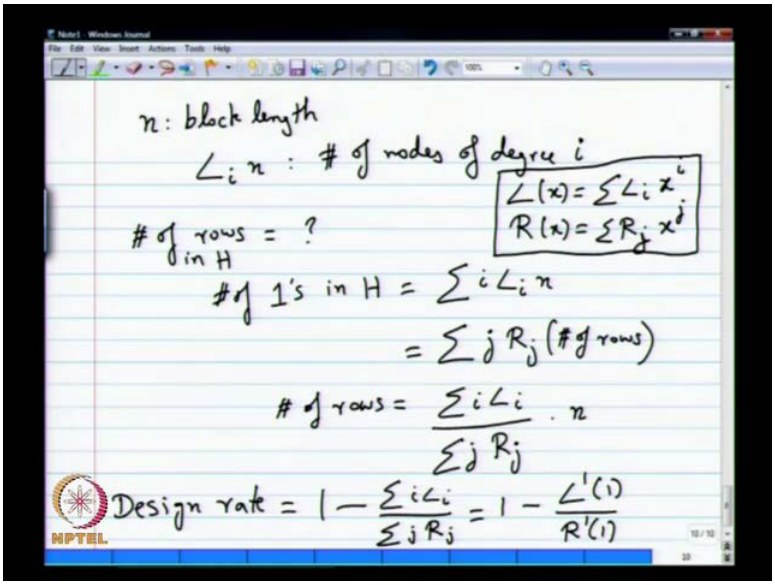
Of course, regular codes there are special case of irregular codes right, yes one column weight, one row weight, it becomes a regular. So, you cannot think of regular codes as being a distinct sets from irregular codes. Regular codes also include the regular codes, alright? Now, we need notation, previously we had just n being the block length and w r being the row weight w c being the column weight that is it that goes enough. Now, obviously that would not be enough. We need more notation and w c 1, w c 2 it is not standard on we have to use standard notation.

So, I am going to introduce all these notation now. So, bit it is not very confusing but still its painful. So, let us just do this so to specify the irregular codes the distribution. So, this kind of thing is called degree distribution. The number of different degrees, number of different row weights or column weights is called degree distributions.

So, when you specify any l b p c codes usually you have to specify the block length n and a degree distribution. So, how do you specify degree distribution is what I am going to talk about now. So, you can do it in two different ways. First thing I will do is what is known as degree distribution from a node prospective because this is what makes most sense.

What is this degree distribution? It is very simple l sub i, we will say is the fraction of bit nodes of degree i and what else is this i, i is going to be let us say usually its starts with 2. You do not want to have degree one bit nodes but can also start with 1. So, of course can start with 1 but usually it starts with 2. There is a corresponding things for check nodes and that is called r i, why am I using l i and r i left and right good. So, r i is for check nodes, so what I am going to do next is going to be slightly, I mean it just simple counting but it always confusing to people when I do this simple counting. So, let me try and do this and as painless way is possible.

(Refer Slide Time: 1:05:01)



So, suppose I say n is my block length, how many nodes do I have of each degree? i l i times n, right? So, I have l i n is the is the number of nodes of degree i. So, clearly this has to be a an integer, so your l i is limited in what is called in a number of decimal places can have or accuracy. What is it called resolution, you cannot have all kinds of l i's route two and all Constance is not possible by l i. You cannot have 1 by route 2, may be for something less.

So, you cannot have such irrational numbers but usually such restrictions are not import, just assume l i is something but then what do you do? You simply approximate this l i n you will flow it or seal it and take something which is approximately the same degree distribution but not the same from n. It can be that way, ultimately it has to be something divided by n.

So, but usually when we specify it there specify l i usually to some ten decimal places or something. So, when you do that you simply flow or seal this l i n in your design. That is the idea is that. So, my question now is it is little bit more difficult how many rows. Suppose n is the block length, what is the number of rows in my paretic check matrix.

So, this requires some little bit more romance counting than before same as last time for except that now you have number of one's is a little bit more difficult thing to compute. Is it more difficult to compute? I do not think so. How do you compute the number of one's? Number of one's is equal to what number of one's in the matrix. It has to be submission i times l i n, right? When you count it from the column prospective and when you count it from the row prospective it should also be submission j times r j number of rows.

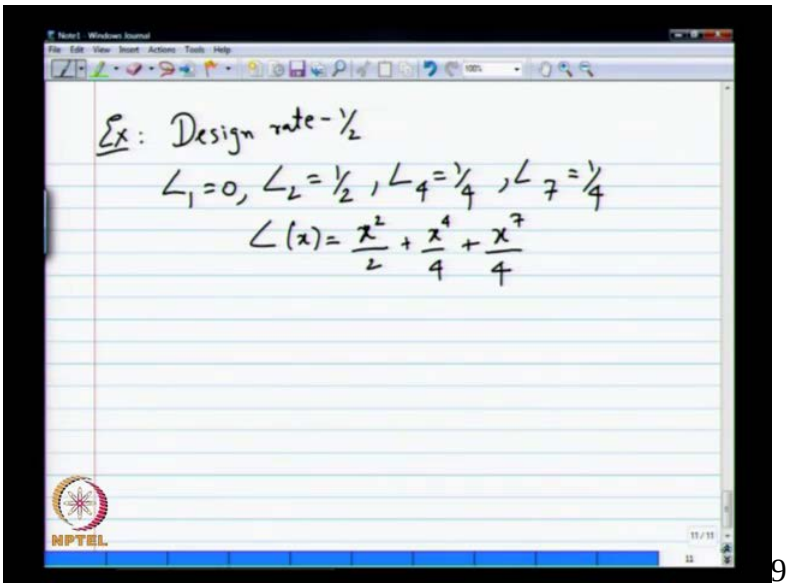
So, what is going to be the number of rows? So, number of rows equals submission i l i divided by submission j r j times n. So, what will this reduced to in the regular case, it was simply w c by w r, clearly even i equals w c l w c is 1, every other l is 0 for the regular case. So, that works out properly, so this is the number of rows. So, of course this also has to be a integer, right?

So, there are further constrains on these l i and r j and all that you cannot have arbitrary choices but once again if it is not an integer what do you do? Usually you adjust this coefficient without too much change to make it an integer. So, that is the idea usually, all right?

Now, what is design rate is 1 minus number of rows by n and that is 1 minus submission i l i divided by submission j r j. So, to write this make this formula look more sophisticated, so what is usually done is you people write it as 1 minus l prime 1 divided by r prime 1, where l of x. Now, is is submission l i x power i and r of x is submission r i x power well r j x power j. If you want i power x what is this why will i power x comes. It is x power i.

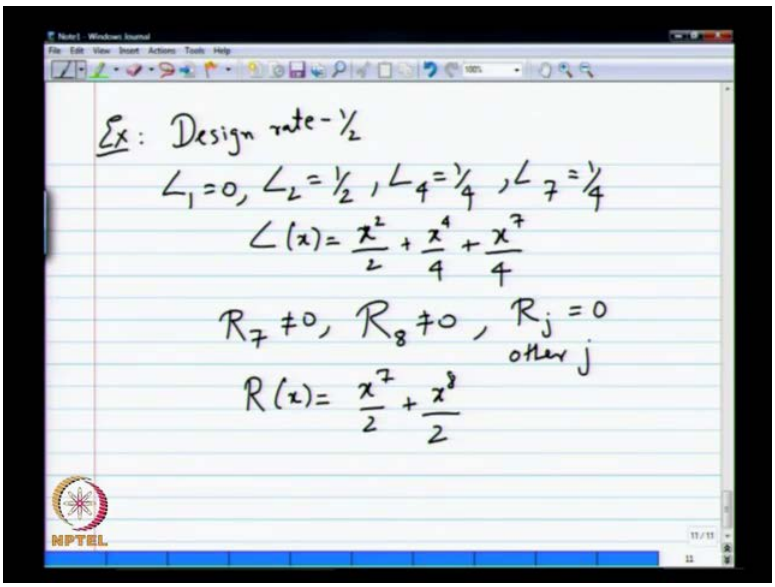
So, what is l prime now what is l prime of x submission i l i x power i minus 1 and then when you put x equals 1 in l prime you get submission i l I, just to make it look a little bit more sophisticated. You do not like the sigma’s, you can do the primes. So, these are called degree distribution polynomials and it is a common way of specifying the degree distribution.

(Refer Slide Time: 1:10:30)



So, let me leave you with one final competition, it is quite simple again which is very simple. Example I am going to say I want a design rate of half and then I am going to say l 1 is 0. Of course, l 2 is half, l 4 is 1 by 4 and l 7 is 1 by 4 again. Of course, the other l i's are 0, so what is l of x x square by 2 plus x power 4 by 4 plus x power 7 by 4, remember? Submission l i has to be equal to 1 because these are all fractions and fractions of bit notes. So, submission l i has to be equal to 1, so I do not know if I mention that somewhere submission li has to be equal to submission rj must be equal to 1. This is also important.

(Refer Slide Time: 1:11:35)



So, this is my l of x and for the right distribution, I am going to say r 7 and r 8 are not 0. Every other r i r j is 0 for other j and the designed rate is half given. All these information let us see if you can spend the next hundred seconds to find out what r 7 and r 8 have to be. Will you first of all will you get a unique solution or will you get multiple solution. You will get a unique solution because there are two equation right r 7 plus r 8 is one and then the other equation for the designed rate. So, when essentially will get unique answer, what do you do tell me what that answer is 1 by 2, that is the correct answer.

So, you can show r 7 equals r 8 r of x equals essentially x power 7 by 2 plus x power 8 by 2. Nobody asked me what this x is, x you accept you know alpha is something that x somehow you accepts just some place hold a variable, does not mean anything just there. So, that I can differentiate nothing more, alright?

So, this is one example, so this is how it will look usually there will be few degrees, not too many of them and fractions will be some nice numbers like this. You can keep them as nice numbers and you get these irregular codes. So, we will stop here and the next thing we will see is how density evaluation is extended for the irregular case, like I said it will be a little bit more complicated approximation but we will do that and then why that is useful for irregular codes for optimizing irregular codes.

Now, you see if I fix designed rate and ask you for the best irregular code, the choices are much more. So, can you get a much better threshold than what you got for the regular case is the question. So, that is why irregular codes are interesting. So, you just increase the choice for a given designed rate for a given designed rate with regular codes. You had only 3, 6, 4, 8, 5, 10 that is all nothing more for irregular codes. Now, you have so many possibilities, if you can do a threshold analysis will the threshold improve or not. That is the main question and when you do irregular codes. So, we will stop here pick up from here next week.