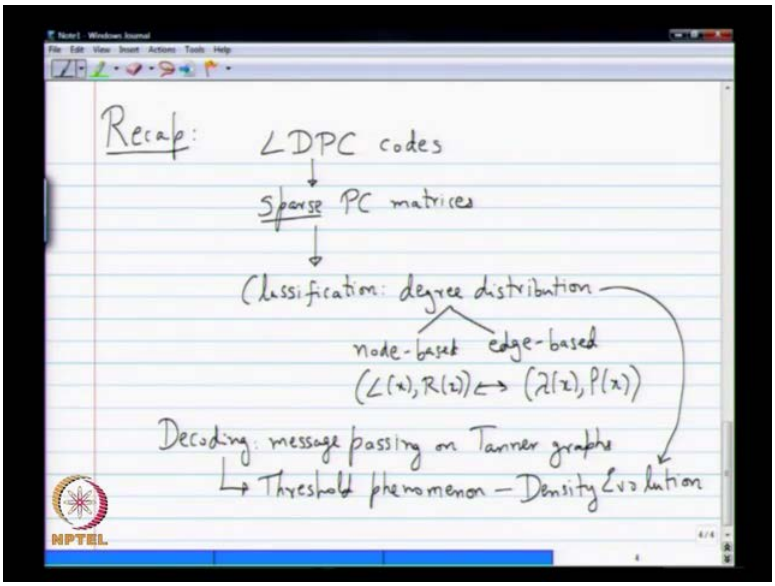
Coding Theory
Prof. Dr. Andrew Thangaraj
Department of Electronics and Communication Engineering
Indian Institute of Technology, Madras

Lecture - 28
Density Evolution for Soft Message Passing Decoding of LDPC Codes

(Refer Slide Time: 00:14)



So, let us begin with the recap, this going to be a rather big and long recap, because we did quite a few new things in previous week. So, the main thing we are looking at right now is LDPC codes. So, we saw that these are defined by sparse parity check metrics. So, as we go long in this recap I am going to ask you a lot of wide questions just to think back and see that we know that reasons for why these things are true.

So, why do why do you think these sparse place a role why this sparse is important.

Student: ((Refer Time: 00:53))

Yeah in a decoder if the parity check matrix is sparse, in your decoder the extrinsic information is very likely to be independent. So, that is the idea that is the reason for sparse and also the quality of extrinsic information will be high when you have sparse parity check. So, those are ideas behind why the parity check matrix has to be sparse. So, after this you classify. So, the classification is based on degree distribution. So, that is an

important method in understanding LDPC codes the degree distribution place an important role here.

So, the there are two different types of degree distribution. One is node based with respect to nodes and the other is edge based degree distribution. They both have some relationship between each other as you can imagine right. So, given the node based distribution which we called as l of x r of x, you can uniquely compute the h based distribution lambda of x row of x. So, this is a good exercise I will urge you to do it. So, given an arbitrary node based distribution l of x r of x how will you convert it into an h based distribution lambda of x row of x.

There is an explicit formula for it is an interesting assignment also. It is not too difficult can be done without too much trouble think about how you are going to do it. So, once again the crucial fact to remember here is the degree distribution along with the block length n, specifies in fact and a non-formal of LDPC codes, does not specify a unique parity check matrix. So, in the way it is really surprising, if you are really if you are educated a lot in classical codes and resole men codes and groups and fields. All these things and suddenly somebody says you know what here is a code that has amazing performance.

How do you construct the parity check matrix, put the once wherever you want just make sure make sure that the number of once you put follows some rule that is it right. So, is a bit countering due to when very surprising that some code that you build like that, without even bothering about areas structures that or the things that you are doing just put it where ever you want it can perform very well. So, that is the idea behind these kind of system. So, it is in a way if you think about it you are going to a large enough block length. So, that any code is likely to be good. So, does not matter that you have to really optimize the structure of the parity check matrix that is not. So, important any code is likely to be good, when you go to a large enough block length that is something that is known to many people known before.

The only problem is in any code you cannot decode you know. So, you put your one's in such a way sparse and following some rules. So, that the decoding is going to work. So, that is the key idea here that is the way you have to think about that, so what the main thing that the degree distribution controls is the density evolution. So, if you think of

decoding so decoders are important. So, decoding is done by, how is the decoding done by message passing on tanner graphs.

So, the tanner graphs is an important notion. You do message passing on tanner graphs and the performance is controlled by a threshold phenomena and also is observed in the decoder. That is determined using density evolution and indirectly or directly. In fact the degree distribution controls this density evolution. So, you get a threshold for the degree distribution all right. So, the way you would go about working with LDPC codes is once you have a particular channels say binary symmetric channel or ((Refer Time: 05:17)) channel. You have some modulation scheme defined, first thing you do is you try and optimize your degree distribution to get the best possible threshold for a given rate.

So, that is a complex optimization it will take a lot of effort and it can be done out I mean it cannot be done online or anything like that. It is done before hand and you decide a degree distribution, like I said there are also databases available for good degree distribution. So, once you have a good degree distribution you have to follow certain construction procedures for constructing a suitable parity check matrix. So, the construction procedure have not spoken much about it. Like I said mostly it is random, but there are also some pseudo random type constructions available.

So, you do that you get a parity check matrix from the on ensamble, once you get a paretic check matrix from the on ensamble you stimulate and check if it is good enough or not for your performance. That is the method or program that you follow to work with LDPC codes. So, it design LDPC codes. So, there are three or four main blocks that you need. One program you need for optimizing the degree distribution, another you need for constructing the parity check matrix and another for doing encoding decoding. So, all these things you should have, if you want to work with LDPC codes. Any questions on this in the general idea why it is working, how it is working etcetera, any comments.

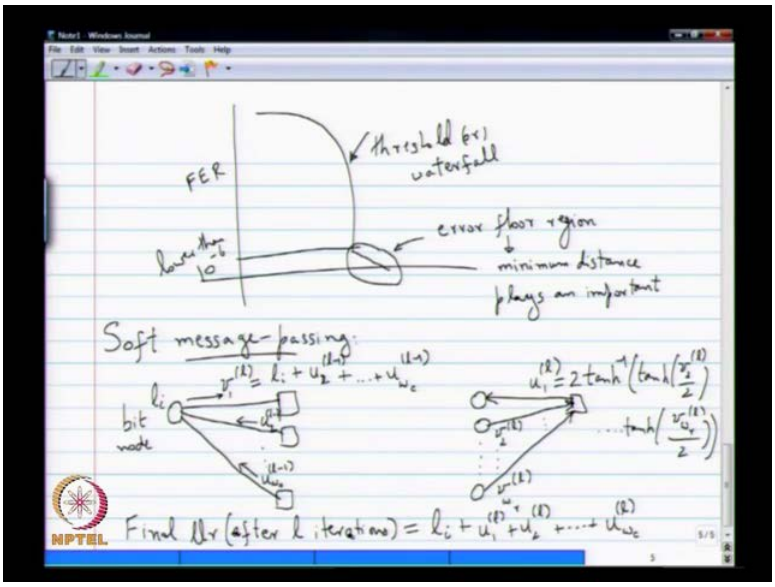
It is fine last week was plus this week currently it was.

Student: We do not really speak about minimal distance.

Yes, we do not really speak about minimal distance any more. The reason is the I mean it is critical I have to tell you that it is critical. It is important, it controls the certain region

of the bit error red curve. So, usually what happens is, when you run these decoders and when you look at say, f error rate verses s n r or some channel parameter.

(Refer Slide Time: 07:17)



It will have the threshold phenomena. Of course, it would not keep on going down for ever. So, at some point it will start flattening out. So, this part is controlled by the minimum distance. So, this is called like the water fall region this is called the threshold region threshold or water fall region. This is pretty much controlled by the random properties of the matrix, I mean it minimum distance does not play a big role here. There are good reasons for that also people have analyzed it. Then you will keep on going down and eventually the minimum distance will start to play a role.

So, this is called the error flow region and minimum distance plays a role here. So, minimum distance is important for this, but usually what happens is for most LDPC codes, if you design a random construction properly when I say properly I am not defining it I am not defining it very clearly, but there are papers about how to do a random construction of LDPC codes, how to make sure you do not get into trouble. If you follow all those rules carefully, it turns out the flow will not hit you in terms offer. This flow will have happen lower than at least 10 power minus 6.

You can do a design, but this flow rule will happen below 10 power minus 6. So, you can do that design quite easily. 10 power minus 6 is something you can stimulate and verify almost channels. You can ensure this. So, if you looking at wireless applications or

applications for 10 power minus 6 frame at a rate is really an overkill. You do not care because you thinks you are happy, but if you want to look storage or other application by 10 power minus 6 is still way too higher for frame error rate.

So, you can imagine today's data centers, have data in millions of terabytes. I do not know its millions of terabytes. Do you know there is lot of big lots of zeros after one's right. So, when you have that in 10 power minus 6 is definitely not acceptable. So, if you want 10 power minus 15, 10 power minus 20, there is no way you can stimulate there you have to have minimum distance for guarantee all right. So, if you have a random construction method of LDPC codes which also gives you a guaranteed good minimum distance you can become very rich.

So, think about that people like Google might buy they run a lot of data symptoms. So, but it is difficult problem people are still working on it. I am also working on it who knows somebody might find it somewhere. So, it is important. So, my answer is it is important depends on the application. If your application only once a constant frame error rate which is low enough say 10 power minus 4, 10 power minus 5. Then you do not have to care too much about it, but you still have pay some attention to the where you construct your matrix.

So, that you do not get a very bad minimum distance minimum distance two and all will really kill you should avoid those things as long as you avoid very low minimum distances, even if it is slightly low not a problem. Once again you might ask me what if I construct a parity check matrix. Find the minimum distance for that for that code that is a difficult problem like I said nobody knows how to find minimum distance given a parity check matrix. Only in your exams they will ask you for a very small problems that you have to find it.

So, small problems, large size problems it is impossible to do it nobody knows how to do that all right. Now, the last thing we were talking about was soft messaging passing decoding. Let me quickly do recap of that and then we will proceed soft messaging I do not know why I wrote that, message passing. So, like all message passing algorithm there are two steps. The first step is let me draw the bit to check step first. In the l th ration I think this is v, u 1, u. I do not know is it I, we did w c minus 1 know.

So, I have a super script may be l minus 1, this is bit node know w c column and row I know check c also for check. So, it is a bit complicated. So, v l 1 and this is. So, if you received I must have used l i for this did I used l i for it or l just l i. So, in the v 1 l is computed as l i plus u 1 l minus 1 may be this is u 2 know u 2 l minus 1 plus to w c right that is not minus 1 q all the way to u l minus 1 w c. That is what we do for that edge and the other edge is you do something similar.

So, you ignore the extrinsic information that came on that edge, but you add up everything else and send it. So, you have to think of the message as the, log likely hood ratio for the i th bit. So, every message that you are passing in soft message passing is a log likely hood ratio for the i th bit. Some estimate of the log likely hood ratio based on what you know at that point. Then when you sent back you have to use the extrinsic rule and make sure that you do not send back information that you all ready have.

So, that results in something called positive feedback you might have read about positive feedback and why it is bad you know. If you keep on feeding back what you knew and suddenly everything will blow up and you will go to either side and after that you cannot you cannot do anything about it. So, it is not good to keep on sending back information its known. So, you have to avoid that. So, this is for the message messages from left to right, if the messages for right to left.

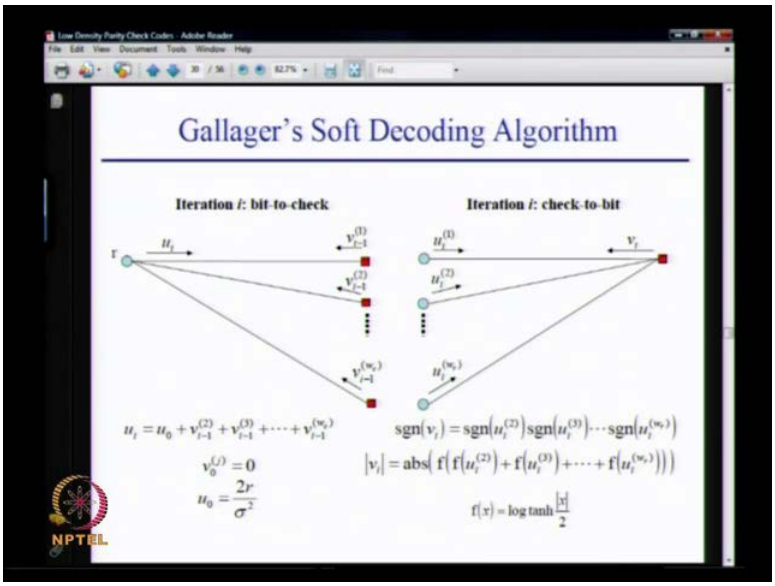
So, it is a little bit more complicated . So, but the formula is quite straight forward in fact at this check node you know that all these bit nodes have to exhaust to zero. So, clearly this bit node has to be equal to the x or f all these other bit nodes. So, that is the rule you use to compute the log likely hood ratio. So, once you know that the log likely hood ratio is what is being send all you have to do is use the computation that you know all ready.

That computation is two times an hyperbolic inverse of the product of an hyperbolic of all the incoming l l r right. So, that is what you send back on each edge what will you send on the second edge in the same formula, except that what will you use here will be v 1 l v 3 l. So, on till v w r l. So, that is the idea. So, you repeat this iteration several times and usually you conclude on the bit node side. So, how do you conclude, if you want to conclude or finish your iteration of the l th step. Then you have to make a decision on some kind of final l l r. So, the final l l r after l iterations this is the l th iteration. After the l iterations the final l l r is written as l i, the first l i plus.

Now, you add everything. So, when you compute the final l l r there is no extrinsic right. So, everything is I mean all the extrinsic information has to be added together. Now, we are not trying to pass back something to anybody else you are trying to make a final decision. So, you simply add everything. So, u n u 1 l plus u 2 l plus. So, on till u w c l. So, this you say is your final l l r after l iterations.

So, how will you decide based on the l l r that is all, if it is greater than 0 you say its bit 1 bit 0 and if it less than 0 you say it is bit 1 that is it ok. So, since we have back to the old laptop I might have a file which I can show you something. So, there are lots of slides here. So, let me see if I find any interesting slide I will stop there and talk about it. So, this is soft decoding I might have like a summary. So, this is the summary that I had.

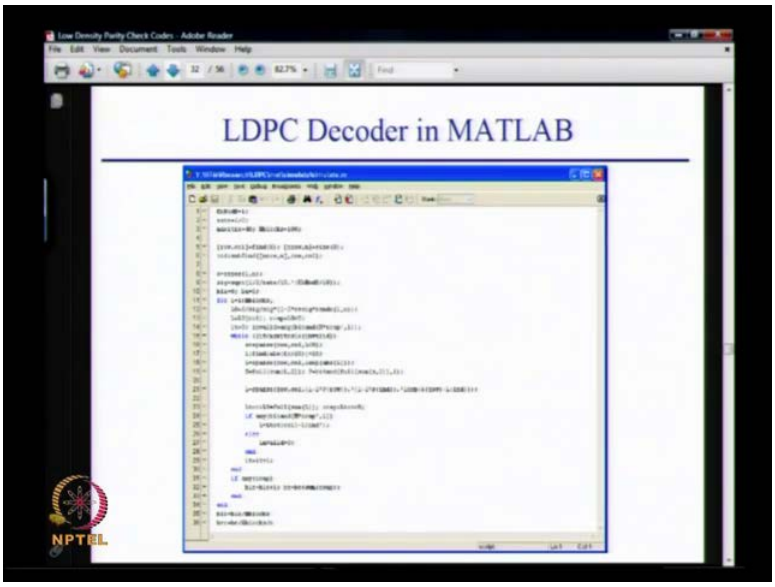
(Refer Slide Time: 17:53)



I wanted to point out one thing in the soft decoder, which I may not have mentioned in detail here. So, the rule that I gave you for check to bit message. So, have I swapped u and v here I think in this notation sorry for that I think this is old presentation, but any way does not matter. So, if you look at the check to bit message here it uses a function f of x which is some log tan hyperbolic mod x by 2. So, it turns out if you use this function then you can split make that operation very efficient that is all. So, you will see a lot of people using this function log tan hyperbolic mod x by 2 and I do not want you to be surprised by it, but it is not very different from doing two time hyperbolic inverse.

You take you divide it by two take tan hyperbolic on both sides and take logarithms you get this formula. So, in VLSI this is particularly nice because all you have to do is have a look up table for this one function f, that is it and everything just involves the same function f. So, that is one thing I pointed out I think I mentioned that briefly in the. So, this is.

(Refer Slide Time: 19:01)



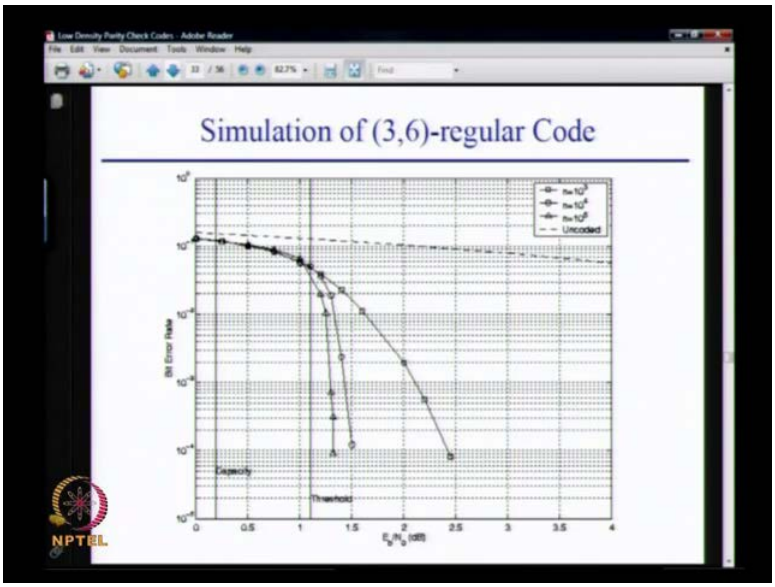
I do not know if you can see the line numbers very clearly on the left hand side. How many line program me is there 36 and you can read the first lines may be. So, e b n or d b n rate and maximum iteration and n blocks. So, hundred or something then there is a parity check matrix h that has been constructed. I do not know if you can read it. So, I am I am trying to read that and then I am doing the decoding. So, this is the entire soft message passing decoder for LPDC codes, for an arbitrary LDPC codes with an arbitrary degree distribution written in mat lab. This is code for that and that is all that is all it takes its 36 lines in mat lab I am sorry.

Student: ((Refer Time: 19:37))

It is I mean. So, only 36 lines was mat lab has some very smart commands to deal with its sparse matrices. If you are very smart about it, you can write a 36 line mat lab programme. So, it will work quite fast. So, I can even in my laptop which is very stubbly low end machine has may not very high end may be has like a two gigahertz processor or something. It is not very fancy, you can run a million blocks of length thousand codes or

something within a few hours. So, it is not it is not very unheard of too complicated stimulations LDPC. So, it is easy to do that small programme. I wanted to show you that because you may have after all the description, you might think it is going to take a lot to implement it. So, it is very easy to implement it in mat lab.

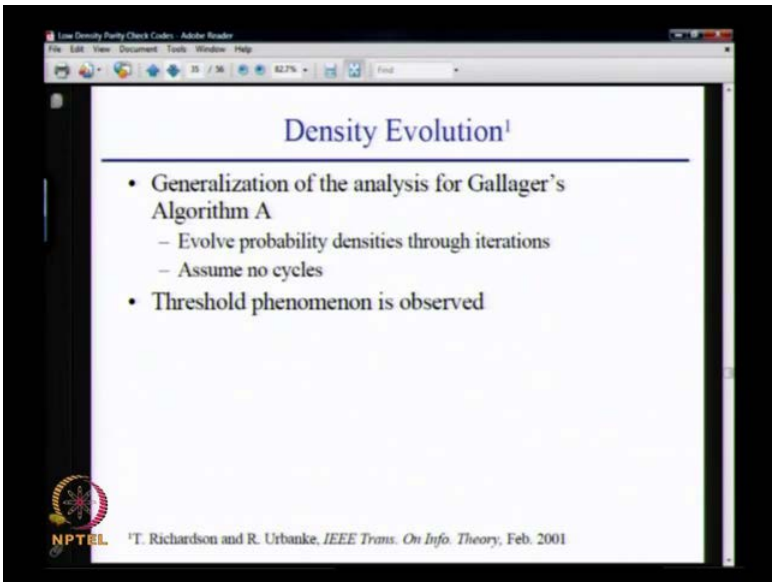
(Refer Slide Time: 20:32)



So, this plot you might have seen I think I have shown you this plot and this is, this plot really captures all that we discussed I mean. So, if you see this the capacity for this is for a rate half codes three comma six regular. So, it has codes and then there is a threshold which will come at around some 1 point 1 I think about 1 point d b d b over and not for rate half. Then you have three curves here one for length one thousand, another for length ten thousand another for length one hundred thousand. You can see how good an estimate the threshold is for the behavior of the decoder. I am going down this is only a bit error right 10 power minus 4, but even if you do frame error rate you will very similar behavior.

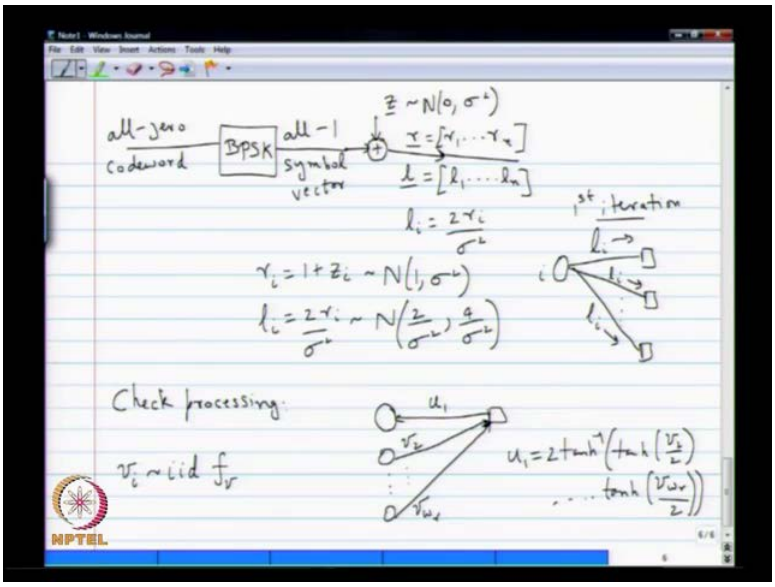
So, this really justifies many of the assumption we made in the density evolution stimulation. In density evolution method right we made a lot of assumptions and at the end of the day this justifies all those assumptions. You might want more intuitively pleasing justifications, but stimulation is ultimately the best certificate for making I mean for the analysis. So, this is that, then let me see if there is any nice slide that I want to show you on density evolution.

(Refer Slide Time: 22:01)



Some way we have we have not spoken about density evolution right. So, we will come back to this when we do density evolution. So, the next point is how do you density evolution for soft message passing decoding. So, previously we did this evolution for allegory decoding in the binary symmetric channel. That was hard message passing right what was passed was just a bit and we could easily track the probability of whether the messages is correct or wrong. So, here we need some more. Let us try to do that the setting is as follows. So, you have a code word ok.

(Refer Slide Time: 22:30)



You have the all 0 code word, we will assume the all 0 code word once again that can be rigorously justified. You can show that the all 0 code word is not a problem then at this goes through BPSK, when the all 0 code word goes through the BPSK modulator what will you get. You get all plus one's right, all one symbol vector to this noise gets added. So, this is normal mean 0 variance sigma square and you get a received vector r.

So, the input to the decoder is not exactly r, it is the input log likely hood ratio right the log likely hood ratio is what. So, if r is r 1 through r n, the input to the decoder is actually the log likely hood ratio vector which is which we have been calling l 1 through l n and what is l i. So, p 2 r i by sigma square. So, so if you transmit the all 1 symbol vector what will be the distribution of r i, r i is what r i is 1 plus z i right and what will be the distribution of this normal with mean 1 and variance sigma square and l i is two times r i by sigma square. You can do a quite arithmetic and check that l i will also be normal with mean 2 by sigma square and variance 4 by sigma square or sigma power 4 by sigma square is correct alright.

So, that is something you can find. So, why did I want to find the distribution of l i because l i is the first message that is passed from bit node to check node right. Remember the first iteration what happens is the first iteration you have a bit i and it is connected to w r check nodes and what is passed in the very first iteration in every h, it is simply l i it is simply l i. So, so l sub i is what you have sending in the first iteration. So, at least we know the distribution for the very first message accurately. Given there are channel is goes in channel, we know that distribution at least for the first iteration for the bit to check node message right.

Now, so the question is how do you track this as the iterations proceed. So, let us look at what happens on the bit node and the check node etc. We will see it can be tracked without too much of a problem. So, so the crucial assumption is all 0 code word which means all the l i have the same distribution. In case if you cannot assume the all 0 code word what will happen, you cannot assume that every edge will be identically distributed because if it is plus 1 then it is going to be plus 2 basic must square. It is minus if it is minus 1 then it is going to be minus right.

So, there is a problem there with not assuming the all 0 code word because you are assuming the all 0code word you have only one distribution to track now what am I

going to do next is to look at the bit node operation and the check node operation. Assume that I know the PDF of the incoming messages and I will further assume that all those messages are identically distributed and independent. Once I assume that let us see how to track the PDF, as it goes through the bit node and the check node it is not too difficult we will see the operation are quite standard.

You can do it very easily not very easily, but at least theoretically it is very easy to describe how to do the tracking of the PDF. Once you track the PDF, that is all you have to do for density evolution right, what is density evolution ultimately, you are tracking the distribution of the message from iteration to iteration. In the first iteration first step I know how to what the distribution is and if I tell you now how to track the distribution through the check node and how to track the distribution through the bit node again, then you can find the distribution for any iteration.

So, that is that part is clear. So, all you need for density evolution is to figure out what happens to the distribution at check node, what happens to the distribution at the bit node. We will again make a standard assumption that at the check node all the information that is coming in is going to be I I D, independent and identically distribution and thus justified because we are using the locally tree idea. So, we know that every neighborhood is locally a tree with high probability 1 over n. So, you would not get any reputations. So, if you do up to only l iterations you are not going to be getting reputations anywhere. So, you will have I I D distributions for the for the for the messages.

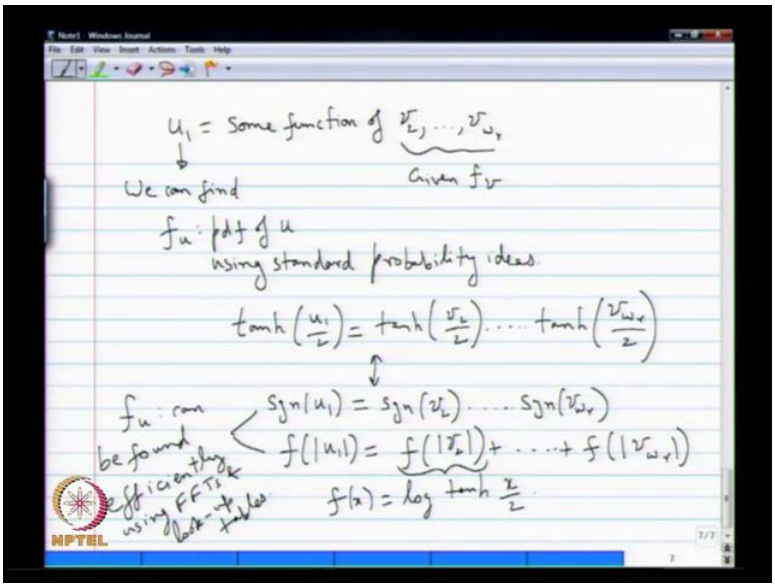
So, the justification is again only the tree neighborhood the local tree local neighborhood that is the only justification that you get same as the allegory. So, let us see what happens at the check node. So, if you are doing check processing you send back you right u 1 and you are receiving v 2 to v w r right. This what happens the check node. So, like I said I am going to assume v i has some distribution which is I I D. Let us say I know the I know the PDF. So, let us say we call it f v I I D, according to some f v maybe I can put an iteration number here, but I do not want to do that now.

So, lets us leave it like this. So, let us see what happens when we do this. So, the question is I know u 1 as a function of v 2 through v w r, what is that function it is some 2 tan hyperbolic inverse of tan hyperbolic v 2 by 2 product of tan hyperbolic v w r by 2

ok. So, ultimately what I have is one function of several independent and identically distributed random variables whose distribution I know. So, technically in terms of probability it is a solved problem. So, you must have learnt this is at some point in your probability course it is a solved problem to find the PDF of u given the PDF of v p 2 to v w. So, it is only a question of implementation detail to figure out how efficiently I can implement that transformation.

So, using that function f of x equals log tan hyperbolic mod x by 2 etcetera. You can implement it very efficiently and let me quickly show that. So, because of which it is not very clean here.

(Refer Slide Time: 30:49)



So, the point I want to make is ultimately u 1 is some function of some deterministic function of v 2 to v w r. So, given f v we can find f u which is the PDF of u using standard probability ideas. How the PDF involves there are several methods. You will see in this particular function. If you use the simplification I was talking about the method will become a little bit more easy what is that simplification. You see this function, this function if you bring the 2 down below and take tan hyperbolic on both sides.

You basically get tan hyperbolic of u 1 by 2 equals tan hyperbolic v 2 by 2 product all the way up to tan hyperbolic v w r by 2. Now, I want to take log on both sides, but the only problem is tan hyperbolic can be negative in which case you cannot keep taking

logarithms. So, what people do is you would separate it into two parts, you look at the sign separately. Once you get it of the sign, you only look at absolute value and find the absolute values. So, you can show that this is the same as sign of u 1 being equal to minus 1, should I write minus 1, let me not write minus 1. I will say product of the signs product of signs of v 2, all the way to v w r.

In absolute value all you have to do is, remember that f of absolute value of u 1. So, instead of equals f of absolute value of v 2 by 2, I think let me just say v 2 plus all the way down to f of 0 e d of v w r, where this f of x basically log tan hyperbolic x by 2. So, of course, mod x by 2. Now, remember v 2 through v w r are I I D according to some distribution if I do f of v 2, I can find the distribution of f of v 2, that is just a single random variable transformation. The standard formula method will work this is an increasing function you can do the standard formula method it will work.

Then what we have you have submission of independent random variables which will become convolution of the PDF. So, all you have to do is implement a convolution here which is very efficiently implementable using fast fury transforms. For instance if you want to do it finite convolution take an f of t it is very easy to do that. Then how do you find the PDF of u again, you have to do f inverse it turns out f inverse is equal to f for this function log tan hyperbolic its own inverse. So, all you have to do is then have some method for doing a transformation by f.

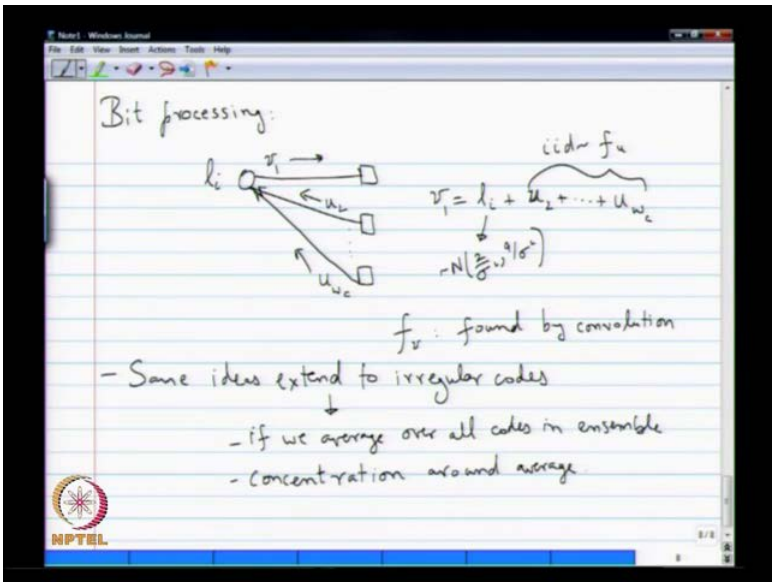
Then have some method for doing a convolution both of which are very standard operations. Once you have those two things you can find the PDF of u 1. So, using these two formula f u can be found efficiently. Using f of t and look up tables and you might wonder why I am worried about efficient implementation, which is quite important that you have an efficient implementation for the density evolution. The reason is when you are trying to optimize the degree distribution, you going to look at thousand of degree distributions and for every degree distribution you have to compute the threshold.

Computing the threshold will involves several runs of density evolution. At least let us say ten runs of density evolution, even might be thousand runs of density evolution to get an accurate threshold. Why do you need so many runs, I mean if you look at the definition of threshold how is threshold defined it is the smallest largest parameter for which the probability of error goes to 0. So, you have to run density evolution several

times still you keep finding that. So, in numerical method for finding that that is also thousand. So, you running this density evolution step thousands of times.

So, several may be even millions of times. So, you have to have a very efficient process for this and this f of t and look up tables keep you know pretty good decent process. So, I know I did not give a very complete step by step method, but I think hopefully you see the idea here. So, the idea here is, from this message that you are sending from check node to bit node is ultimately a function of a deterministic function of the incoming message extrinsic messages. Once you assume the incoming extrinsic messages are I i d according to some distribution and that distribution is known to you. You can compute the distribution of u and the cost of the structure of this tan hyperbolic.

(Refer Slide Time: 36:28)



All that it can even be done reasonably efficiently that is the big idea. Now, let us now move to the other side, if you come to the bit processing, what happens the bit node is even easier have l i and then you have 1 v 1 going this side. Then you have u 2 all the way down to u w c coming in from this side and what is my v 1 is simply a summation of u 2 all the way down to u w c. I know the PDF of this is normal mean 2 by sigma square variance 4 by sigma square. I know that all these guys are I I D according to some f u how will I find distribution of v now simple convolution. So, all you have to do is convolution.

So, f v can be found by for convolution, once again convolution can be discretized form of it can be easily implemented as an f, f t is very efficient. So, at least seems theoretically clear that there is going to be a method density evolution. Density evolution method is very easy to do come up with, you have to track the density assets goes through bit processing and check node processing. So, in fact if you want you can write you can write notation, write everything in notation, but before that I want to move on to irregular codes as well.

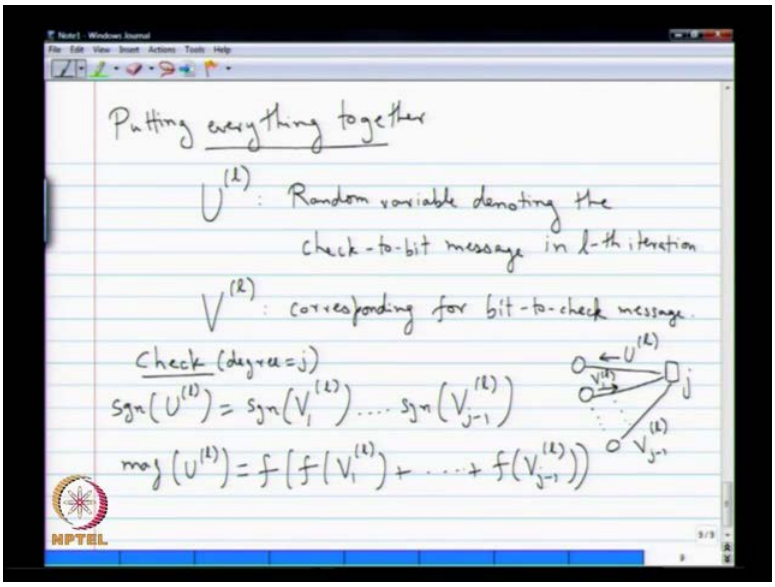
So, far I have been talking about regular codes, but everything that we said. So, far can be extended to irregular codes also same ideas extend to irregular code. What are the additional things you have to do for irregular codes and density evolution. So, the problem in density evolution for irregular codes is, all the neighbor hoods are not identical right, you have so when you assume that all the messages are I I D, but really that assumption will immediately go wrong in irregular codes.

So, how did we get around that assumption you have to average over all the codes in the ensemble sound. So, so this extension happens if you if we average over all codes in on ensemble, but we can do that and still get away with that because why is that we can get away with that, get away with the averaging, why does not it affect anything else. The reason is we have a concentration around the averages, averages result there is also a concentration around average result. Somehow that is not happening in ((Refer Time: 39:57)) no concentration around the average, but in this case there is a strong concentration around the average. So, even if you do analysis with the average you compute the average.

You show that the average is going to 0. It means that any one code also is very likely to obey that average rule. It will also go to sleep when probability will also results. So, very similar ideas can be extended. Once again the tricky issue in irregular codes will be the neighborhoods are not the same depending on the degree the neighborhoods changes. So, you have no business saying assuming I I D for incoming messages, but if you do not assume I I D, there is no way you can do density evolution. So, how do you get around that you get around that by average in over all the codes in the ensemble, which gives you the I I D properties, but then the problem is, then you only get the average block error rate average, average of all the densities.

I mean is it really the real one how do you answer that you answer that by showing a concentration around average proof. So, you say that, if you pick any one candidate codes from the on ensemble its very very likely to be very close to the average. So, once you show all these things all the pieces fall together and you can trust your density evolution. So, one final thing is let us just put everything together now.

(Refer Slide Time: 41:18)



So, let us put everything together and talk about density evolution for the soft message passing algorithm. So, we will let us say capital U l denote a random variable that denotes denoting what denoting U is a check 2 bit messaging right denoting the check to bit message in l th iteration remember. Once again this is average this is, you know this random variable is averaged over several things. It is average over all the channels realization first of all different channel realizations and it is also averaged over all the codes in the ensemble. So, what do we mean by averaged over channel realization well all possible channel outputs right.

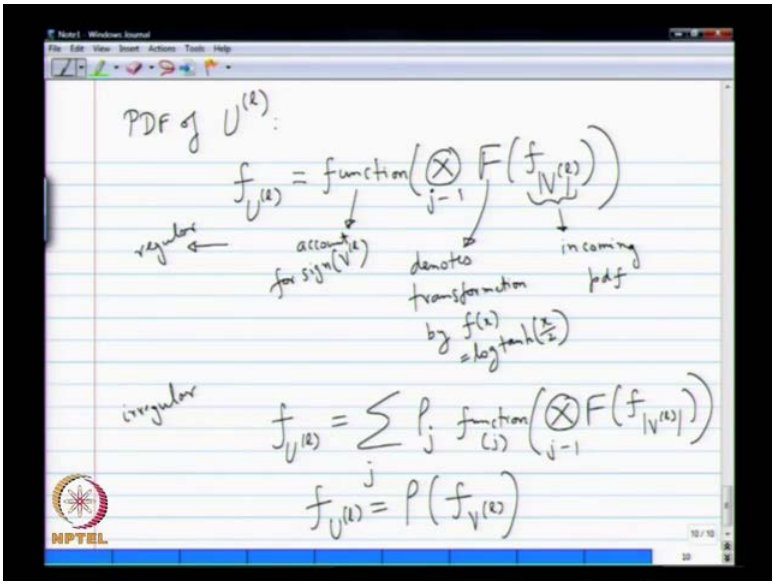
So, that is why we are assuming r i is normal with mean one and variance sigma square. So, that is that is are assumption for r, go back and look at the received vector r. So, clearly the one averaging is definitely happening over the channel realizations, channel noise realization, but the other averaging you have to do for irregular codes is also over all the codes for regular codes you do not have to do that, but for irregular codes you have to do that also. Then v l will be corresponding for bit to check message. It is a

corresponding random variable for bit to check message. Then what you can write is the following kind of rule. So, you can write.

So, I am interested in tracking the PDF of v l right. So, I need so I have the following rule for the check processing at the check node the rule. That I have is sign of u or v, u right u l equals sign of let me write v 2 or let me write v 1, v 1 l all the way down to sign of v. So, let us say check node of degree j. Then it is j minus 1 l what are these v 1 through v j minus l these are all I I D according to the same distribution of v l. Then I have magnitude of u l equals f of v 1 l plus, so on till v j minus 1 l. So, this is the check node of degree j,, remember check node of degree j.

I have a bit nodes j minus 1 bit nodes will contribute to this message, message going in this direction is u l and what comes here is v 1 l all the way down to v j minus 1 l. So, this is the equation that we wrote just now for the check processing, at the degree j check problem. So, what is f of x f of x is log tan hyperbolic mod x by 2.

(Refer Slide Time: 45:53)



So, this is ultimately some transformation. So, what I can do is I can write the PDF of u l can be written as f u l will be can be written as some function. I will say what this function is of a convolution, it denotes a convolution j minus 1 times capital F of the PDF of v l all right. So, this f v l is the incoming PDF capital f denotes transformation by transformation by f f of x f of x, which is log tan hyperbolic x by 2. So, you have the

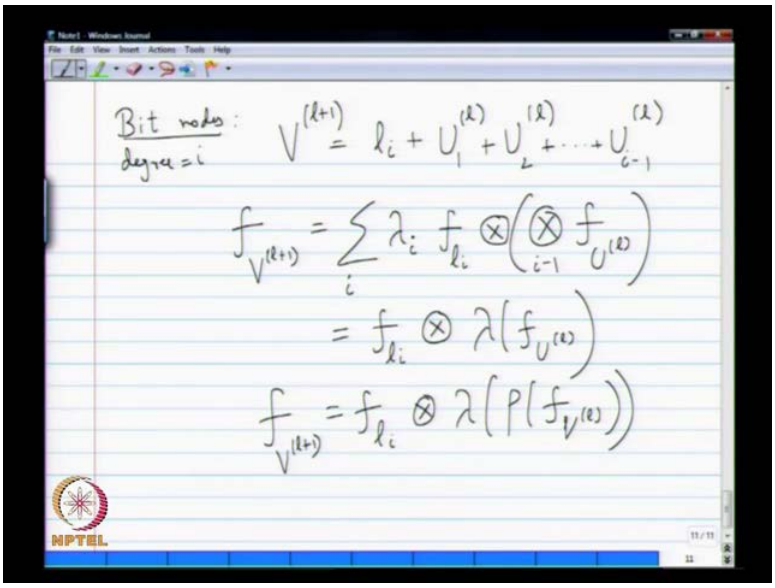
incoming PDF of v l of course, that should be for the absolute of v l. Let me write down absolute value of v l and then you look at the transformation by log 10 hyperbolic x by 2.

You will get some PDF, you convolve it j minus 1 times you will get the new PDF and then I have a function here. This function has to account for the sign also, this has to account for sign of v l. We never took care of sign of v l in this convolution. So, you have to account for the sign of v l that is not too difficult. So, you have to do that if you do that you will get some f u i. So, it is not an explicit thing, but you at least understand the method behind it there is an algorithm.

You can come up with to find the PDF of u l. Now, this is for a degree j check node when you do irregular codes you have to average this over a randomly chosen edge. So, when you do that. So, this is for regular if you have irregular f u l will basically be summation over j row j the same function there are counts for the sign the convolution will be j minus 1 times f of f mod PDF. This function will also involve j and all that right. So, we will have something like that.

So, this complicated function can basically be denoted row. So, this is some standard notation usually people use this, you write this as some row of f v l I could have just written this directly, but any way I thought I will write down the mechanics also, given the PDF of v l. So, in this picture given the PDF of v l, we can find the PDF of u l and that PDF that transformation is denoted row, the same thing you can do for the bit processing also. I am not going to repeat it that is a little bit easier to write down, but at the bit nodes you can also do a very similar thing.

(Refer Slide Time: 49:41)



Let me just quickly write down one common final expression. Here you know that u, it is not u its v, v l plus 1 is basically going to be l i small l i sorry plus u 1 plus u 2 l. So, on till u i minus 1 l. So, this degree is i. So, if you look at it basically the PDF of v l plus 1 is basically going to be summation over i lambda i the PDF of l i, what is the PDF of l i, this will be normal with mean 2 by sigma square and variance 4 by sigma square convolved with, a convolution i minus 1 times of simply. So, i minus 1 times convolve convolving the PDF of u l.

So, you can may be denote this as. So, this you can denote this as f l i convolved with it is convolution is linear right. So, it will commute with the summation. So, you can write it as some lambda times f u l. Now, what is f u l now f u l I know is row of f v l. So, ultimately I have this being f l i convolution lambda row f v l. This is the final density evolution for soft message passing. So, I think the time is up. So, we will stop here for now. We will take small break and continue in about another ten minutes.