

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

Lecture No. # 30
LDPC Code Terminology

Good afternoon welcome back. Just we was discussing with the excellence staff here that, today would be the 30th lecture. We have got about ten to twelve lectures move to go. So, we are getting to the end. And, just a quick recap of what we did last time. Last time, I started the new topic of LDPC codes. And the discussion was part motivation and part explaining terminology. So, I pointed out why it is that, this class of codes is known as LDPC codes; which stands for low-density parity-check codes. And, I told you that whereas in the typical matrix, the number of ones in the matrix will be of order n squared. In the case of an LDPC codes, it is typically on the order of n .

And the second difference is that LDPC codes are described in terms of their parity check matrix. And while it is customary for the entries in the parity check matrix, the row is to be linearly independent in what we have discussed up to now. When we come to the topic of LDPC codes it turns out that that needs some relaxation. So, we will just say that the null space of this matrix defines the code. But we relax the requirement that there was the linearly independent.

(Refer Slide Time: 02:09)

has d_c 1's and each column j contains d_v 1's.

$H = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$ $(m \times n)$

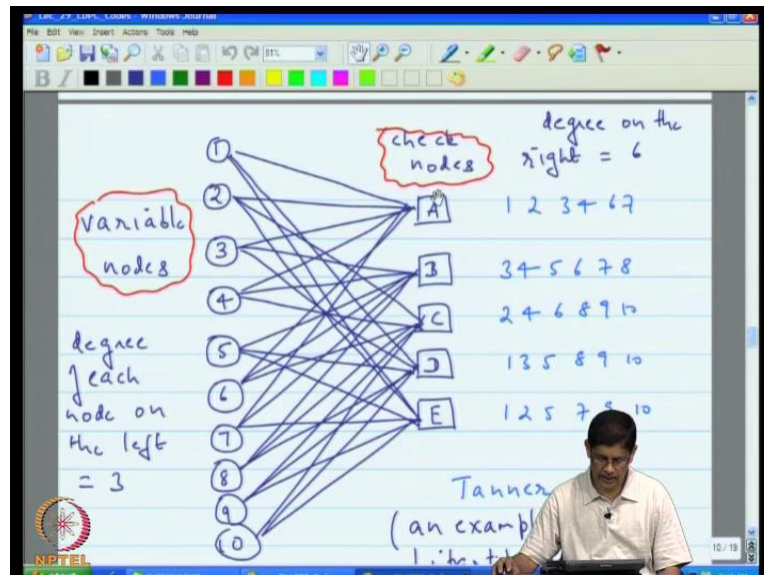
$rk(H) \leq m$

$\therefore m d_c = n d_v$

$\therefore \frac{m}{n} = \frac{d_v}{d_c}$ (2)

Then, we introduced the sub class of LDPC codes known as $d_v d_c$ regular codes. And, here the property of these codes is that, from the point of view of a parity check matrix, every row has an equal number of 1s, which is equal to d_c . Each row stands for a parity check and the column have a constant number of ones, d_v ; and these columns of course, stands for the variable nodes, stands for the code symbols. And then, of course there is a matrix of size m by n , there is the simple relationship which you obtain by just counting in two different ways the number of ones in this matrix. And then, from that you can actually derive a simple inequality, relating to the rate of the code; namely the rate of the code is greater than equal to $1 - d_v/d_c$. And this is called the design rate of the code.

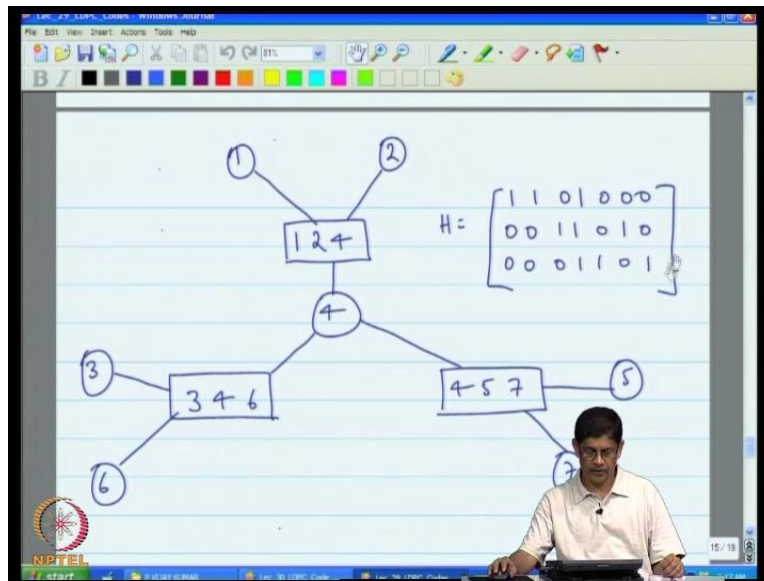
(Refer Slide Time: 03:02)



LDPC codes are typically visualized, and people work with them in a graphical, from a graphical perspective. And what that the graphical perspective does is that it puts one node of the each code of the symbol on the left, one node each parity check on right and makes the appropriate correction. So, for example in this particular code, the code length is ten, there are five parity check symbols. So that, the parity check matrix of the code is of size five by ten and we have a one, for example, we have a one in the A through on the second Column. A through on the second column, if there is a link connecting this, and so that means you can visualize all these edges is as telling you that as identifying the code symbols which meet some to zero and this parity check. They are represented by A. This graph is bipartite, is called a Tanner graph.

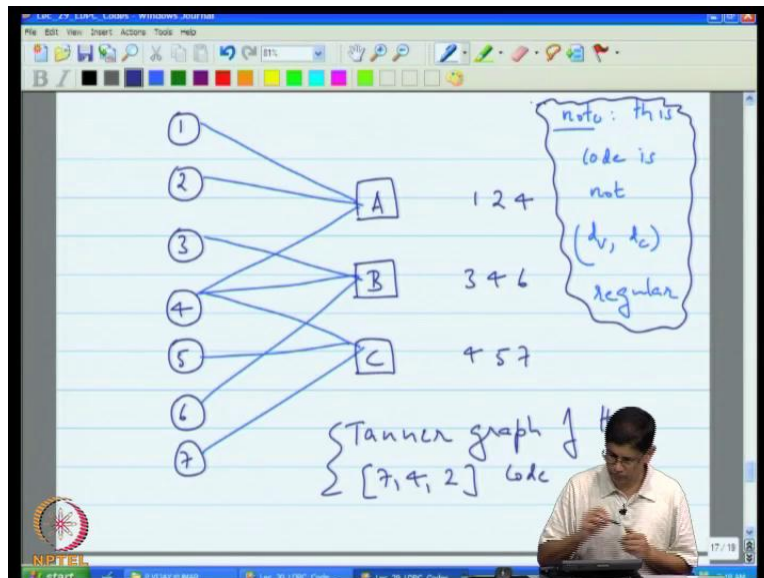
Then, I pointed out that, decoding of these codes is carried out using the sequence of message passing. And, that is very similar to the way in which they pass message across in the junction tree. And that, because... and the d_v, d_c regular condition, then the degree of each node is a fixed constant that gives the decoding complexity, which is typically proportional to the number of edges in this graph is linear in the number of code symbols.

(Refer Slide Time: 04:52)



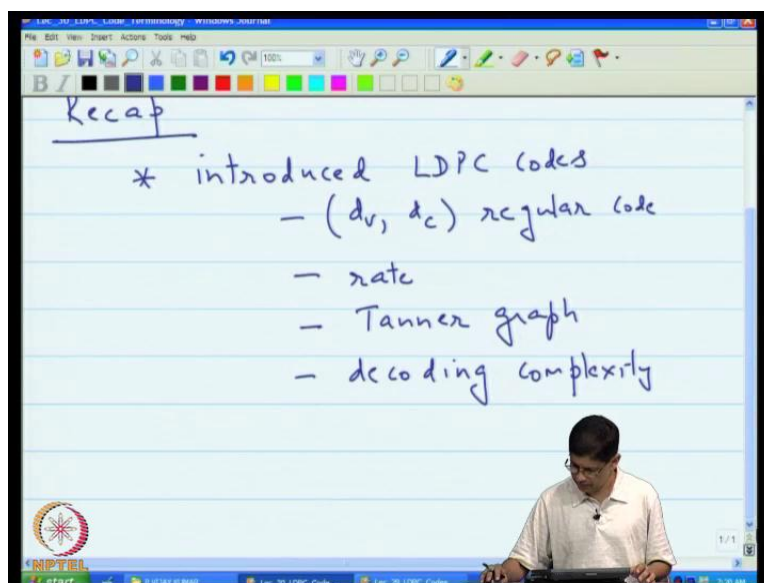
And, just to make the connection with what we have discussed earlier. Here, was a junction tree for the code defined by the parity check matrix.

(Refer Slide Time: 04:59)



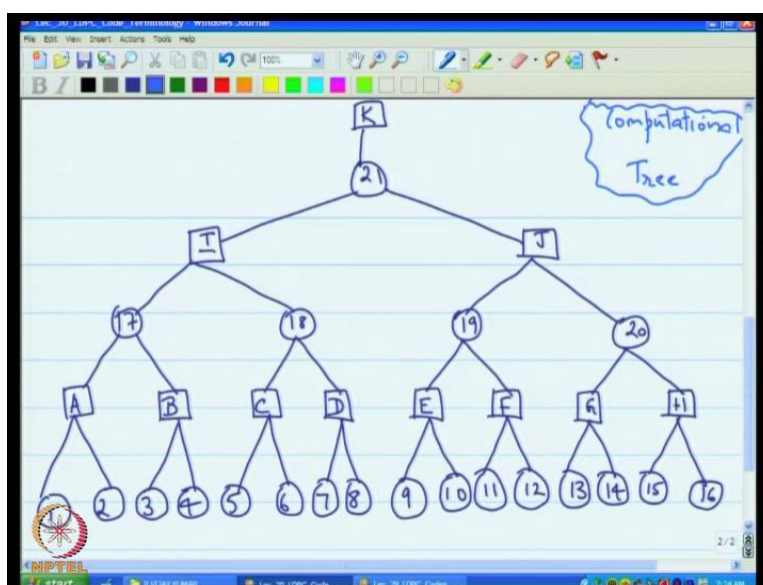
And, however you can rewrite in the form of Tanner graph. This code, however is not an example of d_v, d_c regular code and you can actually see that. However it still does have the representation in the form of the tanner graph. So, maybe it worth, just making a quick note. Note: this code is not a d_v, d_c regular.

(Refer Slide Time: 06:49)



So, that brings us to today's lecture. So, today's lecture; so, this will be lecture thirty. And, I am going to call this very simply LDPC code terminology. Of course, you have already looked at some of the terms that we are going to come up with. But we will continue this because this will allow us to build up a vocabulary which we will need when we discuss performance and method of operations of these codes. So, this is a very quick recap; introduced LDPC codes and we have introduced terminology of d_v , d_c regular codes. We talked about the rate, then we talked about the Tanner graph and of the decoding complexity being linear. So, today we will discuss these codes some more. And, so, perhaps what I will do is I will begin by drawing a graph and we will call upon this graph several times, over the course of the next two or three lectures. It is going to take me a couple of minute to setup this graph. And so let me do that first.

(Refer Slide Time: 08:54)



So, I am going to give this graph a name and we will call it the computational tree. And, I might ask, well, what relevance does this have to us? So, let me bring back our earlier lecture. And, I am going to copy over this particular graph here. Here we go. I am going to remove some of these notations which we do not need at this time. Alright. So, we have this Tanner graph. And, what I mean by this computational tree, which you see here.

What I mean by this here is that, I am going to actually unravel this graph by starting from the particular node. So, I might start from the check node, go back to the variable node, see which all

nodes it leads to and so on. So, basically this is redrawing the graph in a different way. But, apart from that nothing has really changed. So, here what it means is that, I have a variable node which is connected to these three check nodes and each of this is connected to variables nodes and so on. Now, I have numbered this 1 through 21. I want to make it clear that, this computation tree does not directly reflect correspond to this particular code.

See, this is the code whose block length is 10. But the scale is having variables node up to 21. This code has block length at least 21 and typically much larger. This, I do not mean to indicate that, this is, these two codes are the same. But nevertheless, for any given Tanner graph you can always unravel the graph, so to speak and draw it in this form. And now, what I am going to do is going to give you a few definitions involving notions of edge and path and so on. And, those have to do it with this computational tree. So, for example, I will be talking about, the edges that we will be talking about the directed edges. For example, 17 to I, this is an example of the directed edge. And, I to 21 is the second directed edge. And by path, we will mean a sequence of the directed edges.

So, the sequence of directed edges would represent a path. The neighborhood of a node up to a certain depth is basically a collection of edges and nodes that traversed by path whose length is at most d . For example, if you are looking at the neighborhood of the, this node of depth one and that would include these three nodes in these three edges. If you go to depth two and look at the depth two neighborhood of this node, then that would include these two edges, these two nodes, these six nodes and these six edges. So, we will actually make the formal definition of this right now.

(Refer Slide Time: 17:21)

A path in the graph (Tanner graph) is a directed sequence of directed edges
 $\vec{e}_1 \quad \vec{e}_2 \quad \dots \quad \vec{e}_k$ s.t. if
 $\vec{e}_i = (u_i, u'_i)$, then $\vec{e}_{i+1} = (u'_{i+1}, u''_{i+1})$
and $u'_i = u_{i+1}$.

So, I will call this edges and paths; edges, paths and neighborhoods. So, clearly it is clear what we mean by an edge. So here, if this is the variable node and if this were a check node, let me draw that little bit more clearly. So, let say that this is the variable node and let us say that this is a check node and I am typically going to try to represent with the rectangular boxes and variable nodes with circular. So, if I write (v, c) , then this is a directed edge. And that, if I to write (c, v) , that would be a directed edge, but going in the opposite direction.

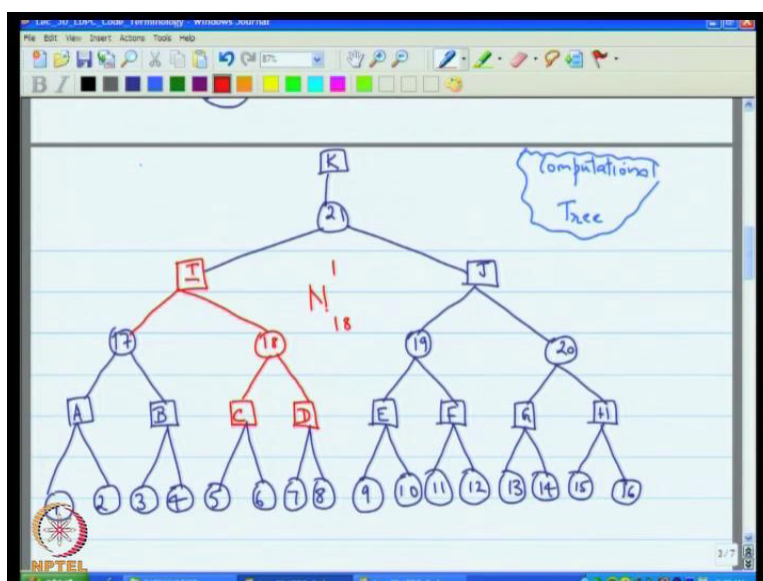
So, now that you understand what the directed edge is and then your path in the graph. And, if you ask the question which graph you mean, well, it could be the Tanner graph except, I am thinking of it is redrawn as the computational tree. But really it is the reference to the Tanner graph. So, you just say the Tanner graph.

A path in the graph is a sequence of directed, is a directed sequence is a directed sequence of the directed edges of the form e_1, e_2, e_k such that, if e_i is equals to u_i, u'_i , then e_{i+1} is equal to u'_i, u''_{i+1} . And, u'_i is equal to u_{i+1} . So, in other words, we are thinking of these two edges as leading has been contiguous and leading from one to the other. So, this is the common vertex for a node. So, from u_i we are going to u_{i+1} , which is this node and then we are going to this node. So, this is what we mean by a path.

And, we say that the length of the path is, for example, here there are k directed edges. So, we will say that the length of the path is k .

So, the length of path is equal to the number of directed edges along the path. Given two nodes in the graph; we will say that the two nodes are at distance d , if they are connected by a path of length d , but not by a path of length less than d . Then, we define $N_u d$. This means the neighborhood of node u to depth d ; which should be taken to mean the induced sub graph consisting of all nodes reached and all edges traversed by paths of length at most d and starting from u . So, let me see if I can illustrate this last definition here. We did that once earlier. Let me just repeat. So, talking about the neighborhood of node eighteen to depth one, as I pointed out just a short while ago. That includes all nodes that are reachable by path of length utmost 1. So, that is 1, 2, 3, 4 and all edges traversed along the path. So, that is 1, 2, and 3. So, that means that, this collection of these forms the neighborhood of the eighteen.

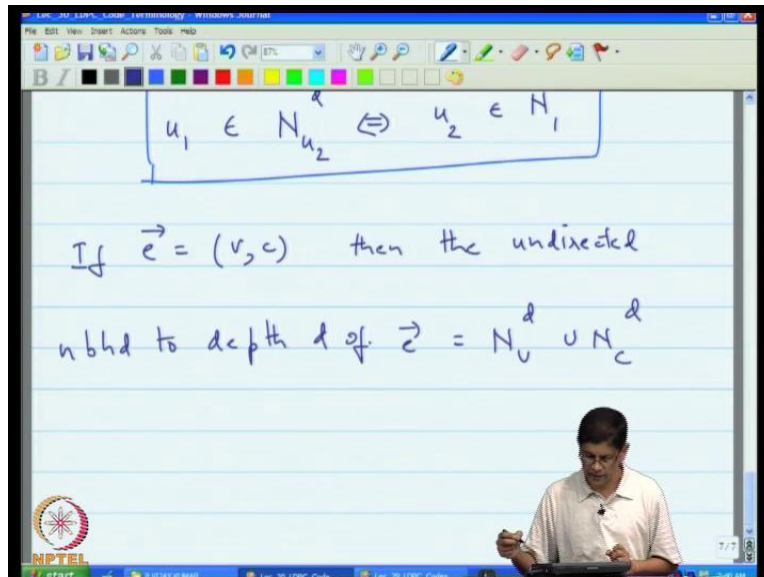
(Refer Slide Time: 24:40)



I think, we can actually use a little bit of technology, if I can do that to indicate that. so, let see if I can put that in red. I am having a slightly harder time to color these edges red, but anyway I think you get the idea. These four nodes are in the neighborhood and also these two edges. Except that I am having hard time in converting these edges to may be I will try in the simple way, so that I do not leave you confused. I need to draw this. So, that is now the neighborhood.

So, this N is the neighborhood of node eighteen to depth one. Then, there is a simple claim at this point.

(Refer Slide Time: 25:45)

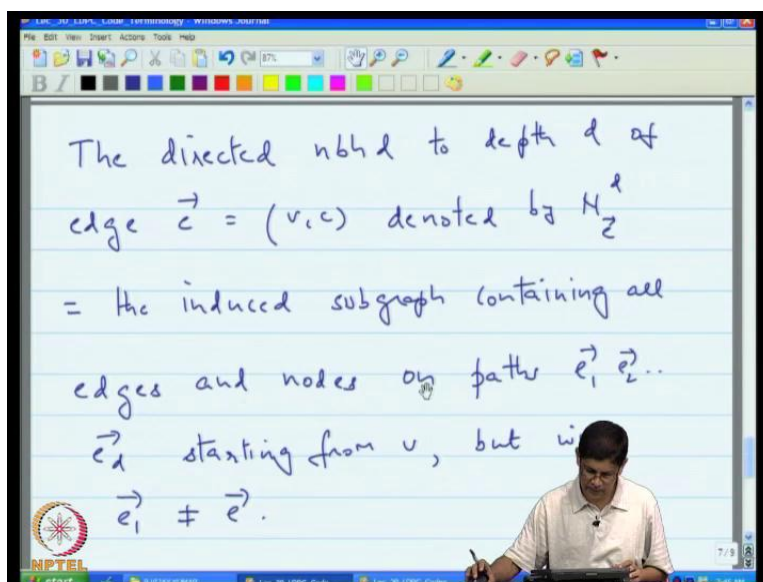


Then, this is simple claim at this point. Note that, u_1 belongs to $N_{u_2}^d$, if and only if for u_2 belongs to $N_{u_1}^d$. That is and that is very simple because what that means, going back to the graph, is that for example, I belongs to the depth one neighborhood of eighteen and it is clear that the reverse is also true. That is eighteen belongs to the depth one neighborhood of I. Now in a similar fashion, if e is an edge v, c , then the undirected neighborhood of e is $N_v^d \cup N_c^d$. Then, the undirected neighborhood to depth d of the directed edge is this. Again, bring to that the graph.

So, for example, they are asking what is the undirected neighborhood of this to depth one. That means that, it is the one depth neighborhood of this node as well as this node. So, in this particular case it would mean; let us see I can get colors right this time. It would mean paths, no, that is going to be hard. So, again the hard way. So, that would in this case mean; so then if you are talking about the edge from 18 to C, then the one depth neighborhood of this edge is the union of the one depth neighborhood of this and one depth neighborhood of this. So, that is why everything that are shown in red is actually included in this. It is also a notion of an undirected neighborhood, of the directed neighborhood. So, this will be your last definition regarding edges.

The directed neighborhood to depth d of edge e equals (v, c) are denoted by N_e^d is the induced sub graph containing all edges and nodes on paths e_1, e_2, \dots, e_d starting from v but with $e_1 \neq e$. So, just a clarification. Earlier, we talked about the undirected neighborhood. So this, the undirected neighborhood is indicated by N_e^d , expect that we do not put that arrow. And, most of this will be clear from context, even if it is confusing now.

(Refer Slide Time: 32:06)



And here, the only difference is that it is similar to the earlier definition, except that the initial path does not, is disallowed from being the original edge. So, for example, in this graph if I have to talk about directed neighborhood of this directed edge; that means I can start from here and I am allowed to walk along path of length d , except that I am forbidden from actually starting on this link. So, that means I can walk. Let us say I am interested in going to depth four, I can go to depth four in any direction except this one. That means by the, that was it is meant by the directed neighborhood. And the difference in terms of notations is that an arrow on the edge.

Here, you can actually prove that now, there is nothing particular important about v and c being like this. We could also start with the c and you would have an definition in which you talked about the directed neighborhood to depth d of edges of type (c, v) and it would be defined in fashion.

(Refer Slide Time: 33:32)

The whiteboard contains the following text:

$$= N_c^d$$

$$e \in N_{c'}^d \Leftrightarrow c' \in N_e^d$$

The directed nbhd to depth d of edge $\vec{e} = (v, c)$ denoted by $N_{\vec{e}}^d$

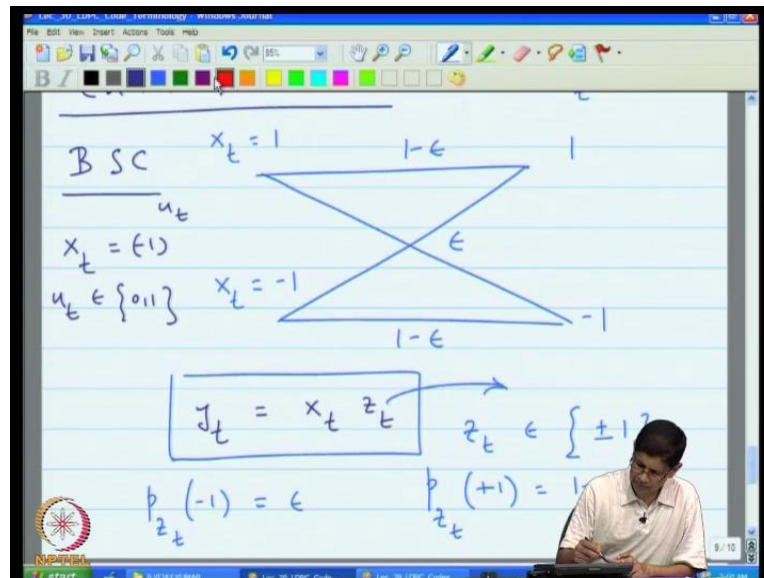
= the induced subgraph containing all

NPTEL

7/9

Another small node denoted here is that; here you can prove this yourself. And, if e belongs to $N_{e'}$ of d , then it must also be true that e' belongs to N_e of d . So, there is a reciprocal relationship, but at least intuitively it is obvious. It takes just a little bit of work to show it, but we want to get into those details. What we have done, let me just take a quick recap. So, it has been a lot of terminology; basically talking about the edges, path and neighborhood I defined what is meant by a directed edge. And then, I talked about the path edges, contiguous edges. let we talk about the length of the path and we talk about distance between two nodes in the Tanner graph, and then we talked about what we mean by the neighborhood of a graph and we talked about neighborhood of a node, neighborhood of an edge, the directed neighborhood as well as the undirected neighborhood. As I said, the reason for going into such a detail in the terminology is because when we talk about decoding we will be in the need of it.

(Refer Slide Time: 35:11)



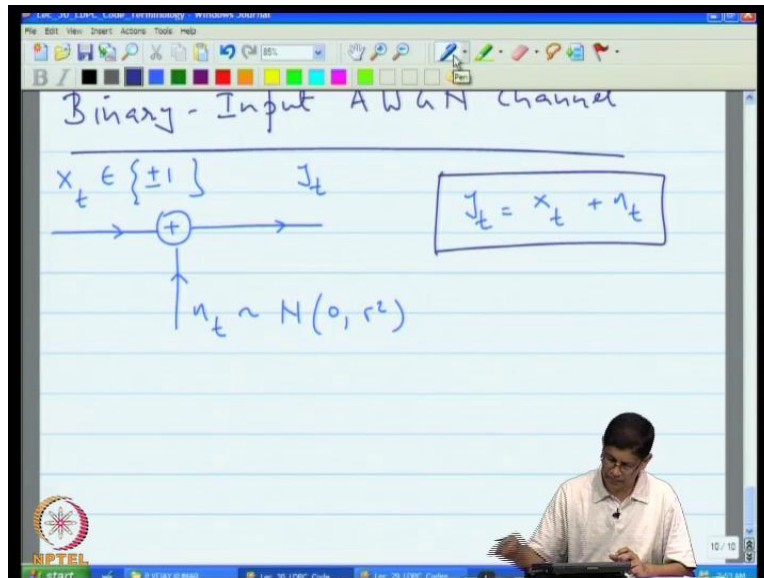
So, next we will talk about channels. So, the two channel models that we have in mind are, first of all the binary symmetric channel. Already, we have seen this before. And, we know the binary selected channel looks like this. There are two inputs and two outputs. Except that, our inputs earlier were 0 and 1. This time our inputs will be x_t equal to 1, x_t equal to minus 1. And similarly, the output will have one and minus 1 and this will represent the output and the cross word probability will be epsilon.

And, now note that, this channel you can actually represent in the following way. We could represents y sub t is in the form of x sub t in to z sub t , where here the z sub t is either plus or minus one. And, the probability of z sub t taken on plus one equals the probability of z t equals one minus epsilon. And from that, it is clear that the probability z t is equal to minus 1 is epsilon.

Now, this terminology is a little different from what we have been used to. Typically, when we encounter the binary symmetric channel, we actually has been additive and we were working in zero one domain. So, in case that puzzles you, what you can actually keep in mind is that, this x_t can be regarded as minus 1 to the u_t and whereas the u_t either 0 or 1. So, this is your small change in a notation for the symbols, but actually it makes things easier. So, it will give you the input and output as plus minus 1 rather than 0 and 1 and the minus one.

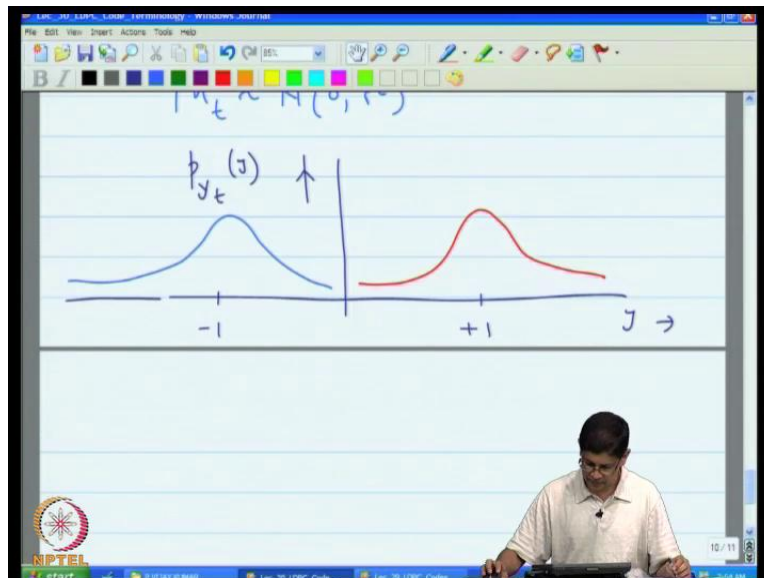
So, one near u_t equal to 0, minus 1 corresponds to u_t equal to 1. And, I just pointed that you can right you represents the channel multiplicatively. That is, you can pretend the output, multiple input symbol or particular symbol, which can taken plus minus 1 value with these probabilities.

(Refer Slide Time: 38:34)



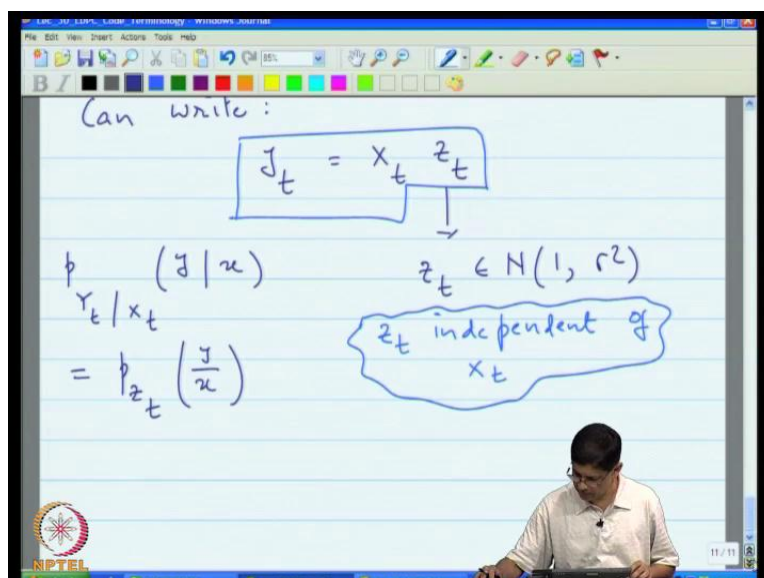
The other channel model we have in mind is the additive White Gaussian noise channel. Or more precisely, the binary-input Additive White Gaussian Noise channels; which, typically we have been representing it like this. They are in input and we are in output. And, let say that the input x_t is again plus minus one, then the noise is added to it is distributed as a Gaussian distribution. So, it is normal with zero mean and variance sigma square. And then, what you have here is y_t . So, the y_t is equal to x_t plus n_t . That is the typical representation.

(Refer Slide Time: 40:15)



And, keeping in mind the nature of the Gaussian density function, I can draw the following picture. So, what I mean to say is that the plus 1 and the minus 1 here are their input. And, what you are looking at here are the density functions of p of y_t of y as y . Now, perhaps I should do the following. Then so, just to avoid confusion, redraw slightly differently. And, I am going to draw this with, in red. So, what I mean by this is that, if the input is the plus one, then y_t has this p d f and input is minus one, then it has this particular p d f.

(Refer Slide Time: 42:00)



Now, here again the interesting thing is that this is the customary representation, where again you can give the multiplicative representation because you can write, can write, can write y_t as x_t times z_t ; where z_t is distributed, normally distributed with mean 1 and variance σ^2 . And, you assume that, $x_t z_t$ is the independent of x_t , it is independent of x_t . Alright. So what, by does that makes sense, well, if, you just think about it. Let say the x_t one, then $y_t z_t$. So, it has this density function because this mean is 1 and the variance is σ^2 . Let us say x_t is equal to minus one, then y_t is minus z_t and this density function will look like this.

But, that is exactly what we had got even under the additive representation. So, actually go to this multiplicative representation. Now, notice one thing that in this particular case, supposing I ask you to compute $p(y_t | x_t)$ of y_t given x_t . Now, that is easy to see that, that it is precisely $p(z_t | x_t)$ and evaluated at y_t / x_t .

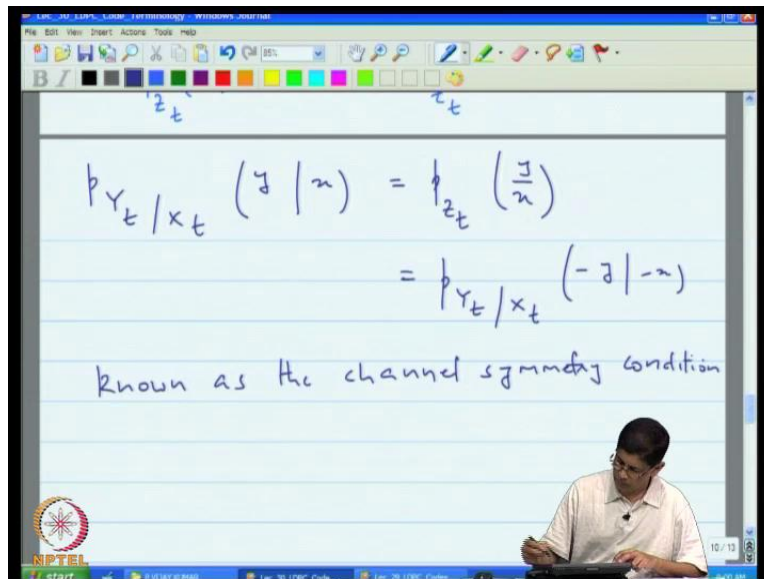
(Refer Slide Time: 44:04)

$$\begin{aligned}
 p(y_t | x_t) &= p(z_t | x_t) \\
 &= p(z_t) \quad \text{since } z_t \text{ is independent of } x_t \\
 &= p(z_t) \quad \text{where } z_t \in N(1, \sigma^2) \\
 &= p(y_t / x_t) \\
 &= p(-y_t / x_t) = p(y_t / x_t)
 \end{aligned}$$

And, it is also easy to see that this is also equal to $p(y_t | x_t)$ of minus y_t minus x_t . The reason being; because this is again of minus y_t by minus x_t , which is $p(z_t | x_t)$ of y_t by x_t . So now, seems to change actually, to do this calculation like this. What you actually saying is that, the probability of the certain density function of y_t corresponding to certain x_t . If you reverse the x_t , you have the same density function for negative value of y_t and you can see that here. Supposing x_t is minus one, let us see and you wanted the density function is minus half would be here. And,

that is same as the density function of y equal to plus half, when x is plus one. If you reverse both quantities, then the value of the density function remains the same. This is the fact that we will use shortly. And, this is known as channel symmetric condition. And you want to point out that the same channel condition, the symmetric condition also holds in the case of the binary symmetric channel.

(Refer Slide Time: 46:07)



$$p_{Y_t/X_t}(y|x) = p_{z_t}\left(\frac{y}{x}\right)$$

$$= p_{Y_t/X_t}(-y|-x)$$

known as the channel symmetry condition

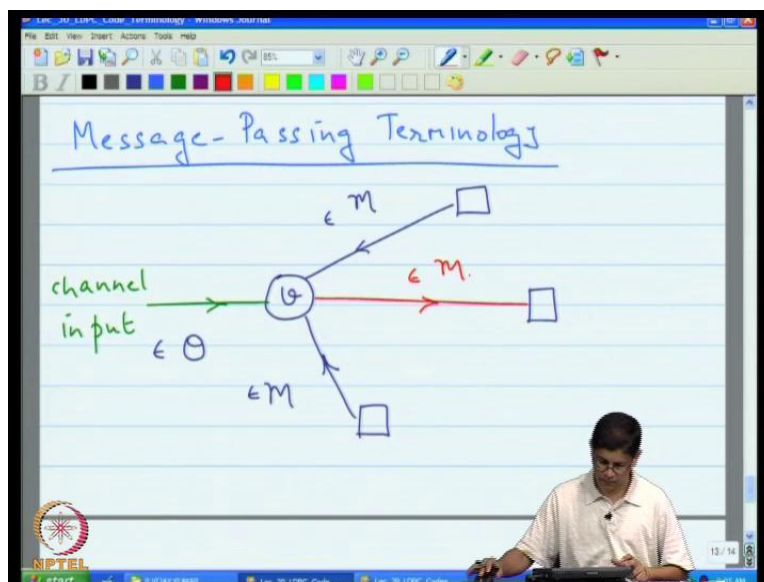
So, let us check that. p of y t given x t of y given x is again p of z t given y by x , which is equal to p of y t given x t of minus y given minus x , for exactly the same reasons as seen before. So, the comment here is that, so this is known as the channel symmetry condition as applied to the binary symmetric channel.

Now, what we want to do is to introduce some terminology associated with message passing. So, let me label that message passing terminology. So, let us go back to the figure here. So, this is the Tanner graph. And as I mentioned some time back, the way we are actually going to decode this code is by actually passing messages back and forth along this edges. We actually start. Let us say from the variable nodes passing along these edge to the check node and back and forth. So, we will be doing this iteratively. And, the way the iterations are numbered, there is an initial or zero iteration in which message flow from the variable nodes to check nodes. And then, you start with iteration one, which is a flow of information from the check nodes to variable nodes

followed by a flow of information from the variable nodes to check nodes. So, that would be the first iteration. Then you would have the second and third and so on.

So, messages are going back and forth. These are the variables nodes, these are the check nodes. Now, initially if you think about it, the variables nodes are the ones that will actually initiate message passing, because they are the only one should have any information at hand and the information that they get is actually the information that they get from the channel input. So, that channel input will then, they will use to actually determine or compute the first set of messages that they passed out. Now, all of the messages that have been passed out are real numbers. So, all of them are actually real numbers or the subset of the real numbers. And now, what do you like to do is, we would like to just introduce some notations relating to the alphabets of these messages.

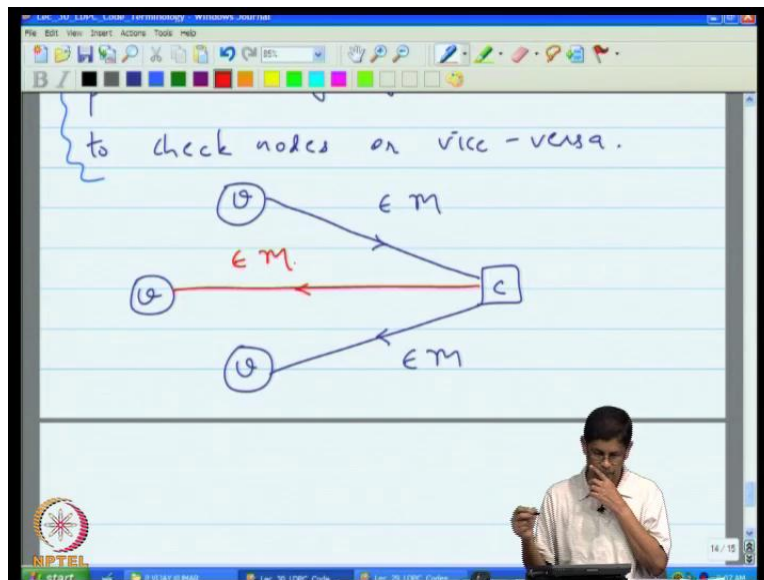
(Refer Slide Time: 49:45)



So, let we draw a graph picture as I introduce this terminology, so, here is your variable node and here is the input from ... just use solid lines, I am using different color any way. I mean just use a solid line here. This is the input that is received from the channel. So, we will make a note of that. And then, on the other hand, it also receives input from the check nodes. And, what it does is that it sends back a message to the parity nodes. So over here, some where over here this is the parity node, this is the parity node and the message that is going back is also going back to a

parity node. Now, all that you want to do is you want to identify the alphabet, which is used to pass these messages or the alphabet corresponding to these messages. So, this alphabet, which is the channel output alphabet will be called \mathcal{o} . So, the message alphabet will come from \mathcal{o} . This message alphabet will come from script \mathcal{n} and so will be this.

(Refer Slide Time: 52:08)



So, then \mathcal{o} is the output alphabet of the channel and \mathcal{m} is the common alphabet employed to pass messages from the variable node to check nodes or vice versa. So, here we will this diagram variable node, but you have a similar picture at the check node. So, the check node we would have and it have a variable node, which are communicating with the check node. And, here again the messages are from the alphabet \mathcal{m} . So, we will note that both \mathcal{o} and \mathcal{m} , may be assumed write it in other words both \mathcal{o} and \mathcal{m} are subsets of the set \mathcal{R} of all real numbers.

And, so basically you just summarize as you just have a couple of minutes left. What we did was we continued developing terminology of LDPC codes that us to, it will make it easier for us to discuss the decoding algorithms. So, I introduce the notion of an edge, a path and of a neighborhood of the length of the path. And then, we talked about channel models. And, I introduced the binary symmetric channel from multiplicative; through a multiplicative representation and that is the same for the additive White Gaussian Noise channel. And, these are the two principle channel modules that we will be looking at. And I introduced the notion of the

channel symmetry. And then, we were just getting in to the alphabet that is used actually to pass the messages. And so, let me just close the lecture here by writing down the equation, which I will continue to develop in the next lecture may be this is the good place to stop. So, we will introduce the notation next time. Thank you.