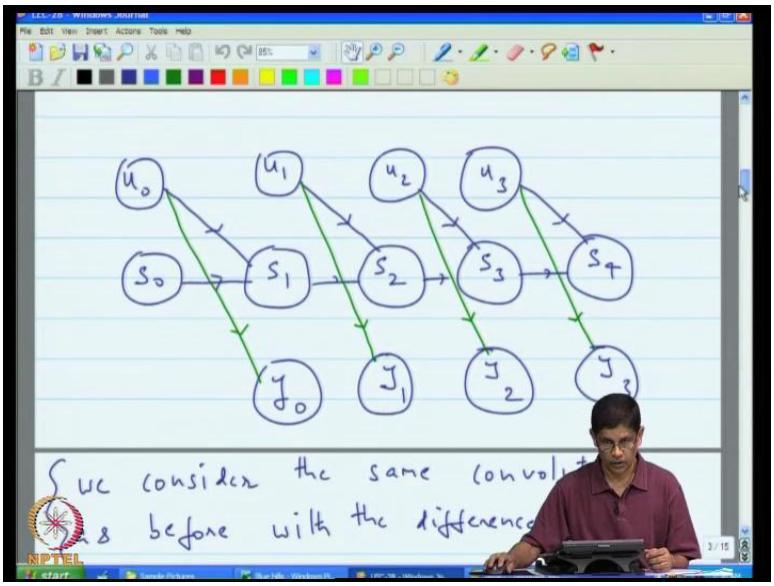
Error Correcting Codes
Dr. P. Vijay Kumar
Electrical Communication Engineering
Indian Institute of Science, Bangalore

Lecture No. # 29

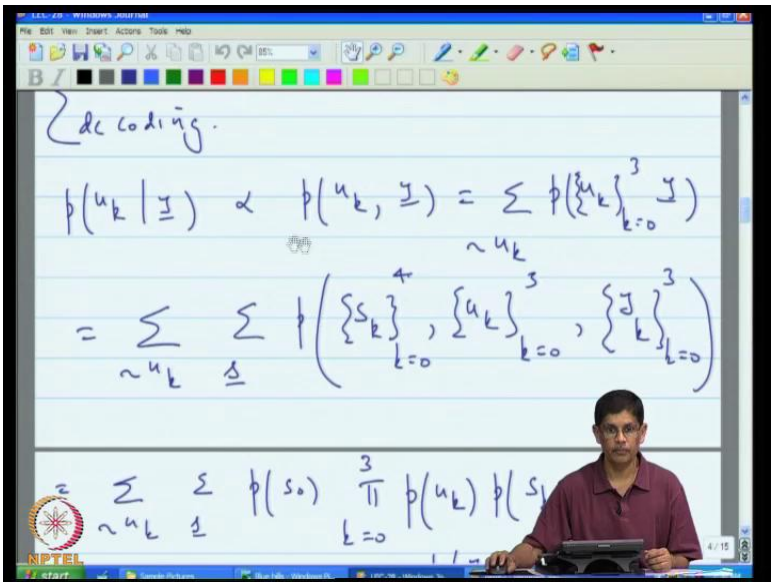
Good afternoon, welcome back, this is lecture twenty nine; and let me quickly recap, what we did last time. I just give you a quick run through for a lecture. So, our last lecture was basically have do go with the maximum likelihood code symbol of decoding of the convolutional code. And it really represented more or less the n of our discussion on the GDL. So, we were looking at the GDL, and we were applying it to the problem of decoding of a convolutional code. And earlier, when we discussed convolutional code decoding, we have talked about the viterbi algorithm. Now, this is the alternative value which is called BCJR algorithm and although, it can derived independently and so on, but I did here where I should have, how you can actually generated using the GDL.

(Refer Slide Time: 01:33)



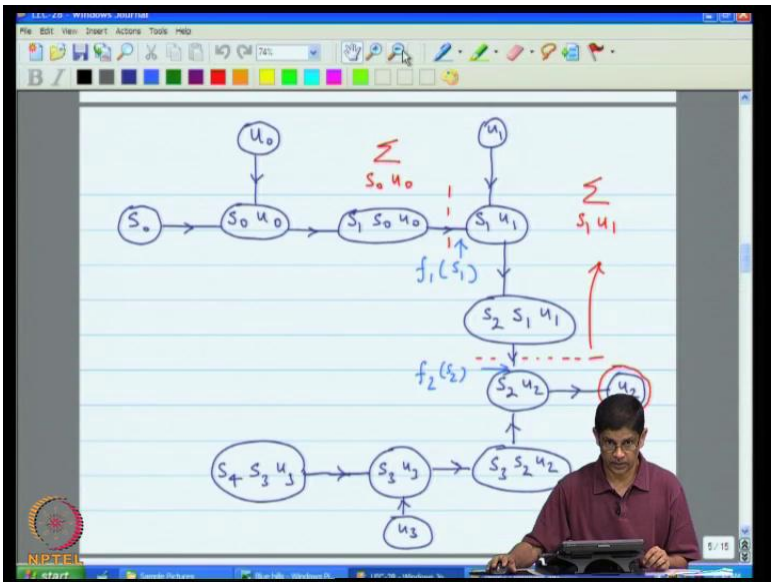
So, the entire which is more or less devoted to showing how the GDL in certile setting, it gives the BCJR algorithm. And so, the starting point was the description of the convolution code in terms of, what I told you were bayesian network.

(Refer Slide Time: 01:40)



And, here is the mpf formulation of the maximum likelihood code simple decoding problem.

(Refer Slide Time: 01:55)

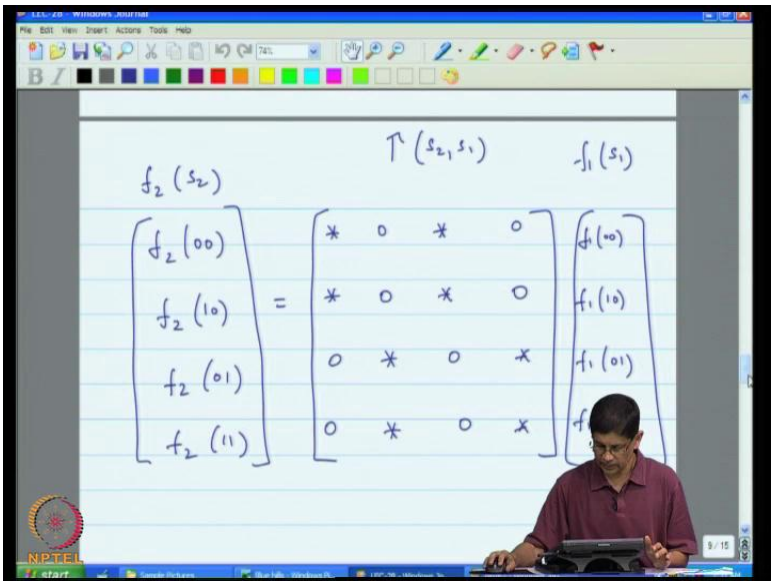


We identify the local domain and local kernals, then we constructed, then we construct the junction tree out of this. And then, once you have the junction tree, then the objective function let us say, in general objective function would be to compute the likelihood of all the message symbols. We took a particular message symbol, u 2, and then we showed how we compute the

objective function? The second step after constructing the junction tree is to actually pass messages. Now, if you treat it like a single vertex problem, then orient by the edges towards the single vertex and passed messages and that what we did.

We have the f 1 of s 1 and f 2 of s 2, our schedule looks like a forward message passing, and then backward message passing sequence both of them are iterative. So in the forward direction, you have f 1 of s 1, and the second iteration becomes of f 2 of s 2 and so on. And we shown what that should iteration look like, and then I gave the interpretation in terms of trellis of the particular example conventional code. I showed you that what it really means is that you are actually doing the matrix multiplication where actually looking at something on the left on going to the right.

(Refer Slide Time: 03:24)



So, it is matrix multiplication of this type, then the backwards algorithm is also very similar; so again here g 4 of s 4 evaluates to g 3 of s 3, because you are going backwards. And the relationship between g 3 s 3 and g 4 s 4 is given by this. Again it can be interpreted as matrix multiplication, and this is what we did. So, both forward and backward are matrix multiplication steps. And then the final step to actually compute the objective function, so let us say, what you want compute this; and this point what you have is you have access f 2 of s 2 from this side and g 3 of s 3, and you want compute the objective function. So, what you do is, say you pass

messages through this imaginaries and multiple by the local cornel and end up with the expression like this. So, end up with of this, and that more or less concludes the algorithm.

Now, it is possible to interoperate even in this calculation on the trellis, but for lack of time I want actually do it. Wherever, what I might do is just add a note here, since you will have access to be the notes, I do not thing remember. So we will just say that this concludes our discussion of the BCJR algorithm, and then there is another command I actually wanted to make in this connection, which is that if you wanted to switch from the BCJR algorithm to the viterbi decoding algorithm. And then, I am going to say it in words and I will summaries it very quickly.

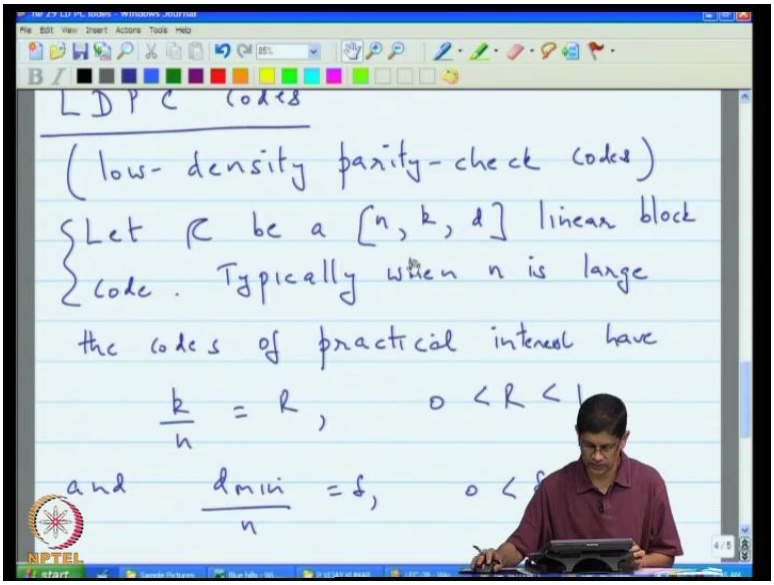
Then, it terms are that, you are you can actually formulate the problem in a very similar fashion; and the only difference been that whereas your operations, you are operating in the some product semi rings. You would be operating there in the max product scenario; so wherever, the summation, you would actually have the max. And apart from that, all the operations are very similar, and you could again have forward and backward message passing.

You might say, in the viterbi algorithm may thus, only message passing in the forward direction a wise in that we have backward direction. The reason, the difference raises is because let me just take to sample lecture from earlier, so here you have the trellis. So when we look at viterbi decoding algorithm, we saw that the algorithm only most from left to right, may be the little bit passing how can have forward and back ward, where the reason is that, in the viterbi algorithm we carry at every recursion, we compute matrix assigned to the node at a certain node level, and moved from node level to the next node level, and we keep going from left to the right. But we also do something else which is we retain the memory of the path leading up to the particular node, which has the logic symmetric. So, these are the survival, so not only keep the survival matrix. We also track, it track of the survival. For example, if this is the survival path, you keep track of it.

In the GDL algorithm, you do no t track of it, what you do is, only keep of the survival matrix. If you go forward the survival matrix, you come back ward to the survival matrix. And then once across the link you look matrix on the left, matrix on the right, find the best connection, and then decode one symbol at a time. I realize this is little bit fast, but essentially it tells you, how you can actually do viterbi decoding as well.

With that, I will start today lecture, the recap is completed discussion of the BCJR algorithm, very quickly recap. And then just quick remark here, that viterbi algorithm can also be recovered using the GDL, when simply operates in the max product semiring in place of the sum product semiring. In the algorithm, the essential difference lies in replacing the summation operation by the max of operation. And for lack of time, I will actually stop here and introduce the new topic. So, the topic for today is LDPC codes.

(Refer Slide Time: 12:28)



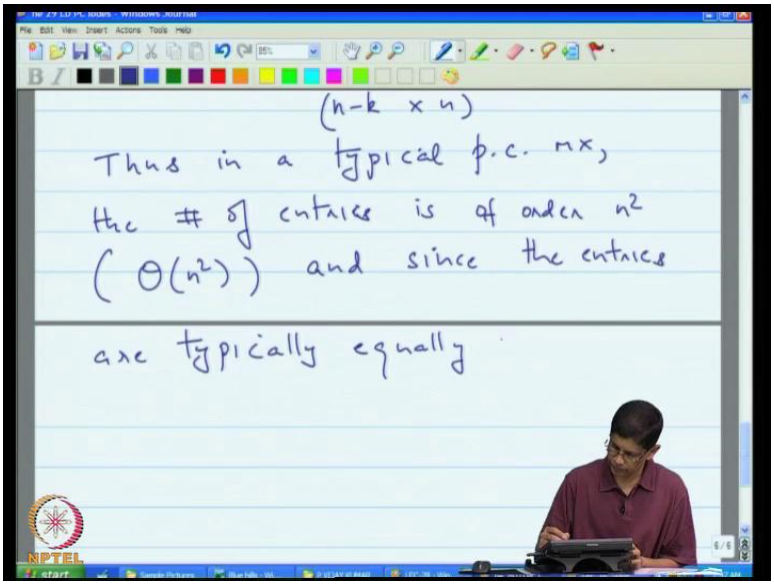
So what is LDPC stands for? So LDPC stands for Low Density Parity Check Code. So, what so that leads to the question what do you mean by low density? And so let me begin I am trying to explain that. So, these are block codes, and every block code has parity check matrix, so supposing you look at the parity check matrix, then as we know typically, if you are talking about 742 code, so let me put that assumption here, rather we do one thing, let me move this little bit down here. And say let c be a seven are more generally and n k d linear block code. As you all by now, this is the black length and this is mention and that is the minimum distance. Now typically n is one, you are really codes, where k some fix fraction of n and similarly d, so typically, I guess I need to create some more place for myself.

Typically, when n is large and I have already explain to you, why large block lengths or interest when you have label communication. When n is large, the codes of practical interest have k by n

equal to R, where R sum integer line between 0 and 1, and d min by n equal to delta, for some delta line between 0 and 1. What you should interrupt this to mean is that term? When n is large, you can think of this is n to n infinite. It is not really infinite, it is not like you might in practice has code is block length is sixteen thousand three two thousand. So, it is large and then you are interested in course where the dimension is 3 and the length is 16000 this more like k by n is some appreciable fraction.

This R some fixed faction between zero and one often, R is half for example or R is one-third or one-fourth. What that means that, as n tends to infinity should escape, this goes minimum distance. You would typically like a minimum distance to also grow fractionally although this fraction could be much smaller. Now, the way defined the parity check matrix this in the context of this notation that is when have a n k d code.

(Refer Slide Time: 16:41)



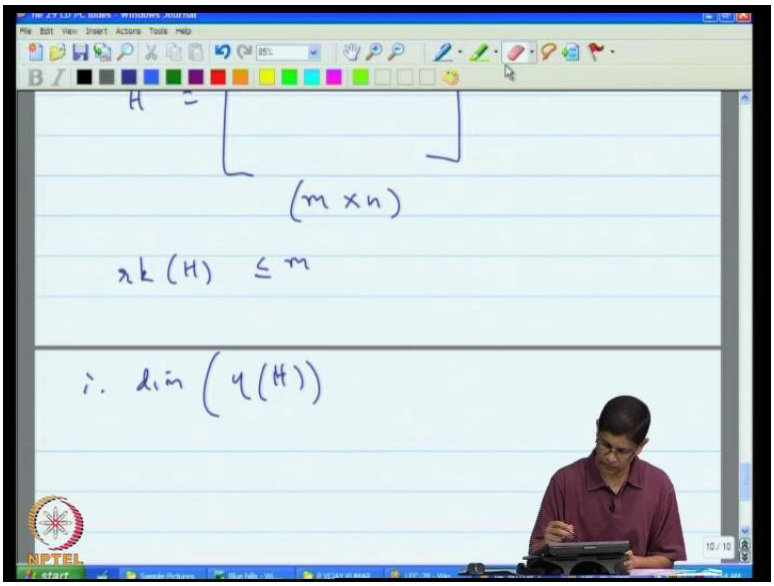
Your parity check matrix would be of size n minus k by n, so this is the matrix of the time. So, what that means is that, the total number of n k is on the order of n square. And in random parity check matrix, then equal number zeros and ones and therefore, the number of one in the check parity matrix should be order of n square as well. Let us we make the order of them. Thus in a typical parity check matrix, the number of entries is of order n square and since entries typically equally likely to be either 0 or 1. The number of 1s would be on the order of n square as well.

This is where LDPC codes, because LDPC codes the number of 1s is not order of n square rather of order n. However in the case of the LDPC codes, the number of ones in H is of order n that is one difference.

Then again, there is the second difference; the second difference is that when we talk about parity check matrix. So, this matrix of size n minus k by n; and when we introduce the parity check matrices, we said that these are really generator matrix of the dual code therefore the rows are linear and independent. However, the LDPC codes, we relax that the assumption, so that means which meaningful to talk about parity check matrix without requiring the rows in linear independent.

A second difference in the case of LDPC codes is that rows of H need not necessarily be linearly independent. However, we still require that rows of H generate the dual code chipper. This can be used to show that, once again this implies that, the null space of H is precisely the original code c. Now, there is the particular class of LDPC codes; there I will now, introduce a d sub v d sub c regular LDPC codes is one. In which, each row of H has d c 1s and each column of H contains d v 1s.

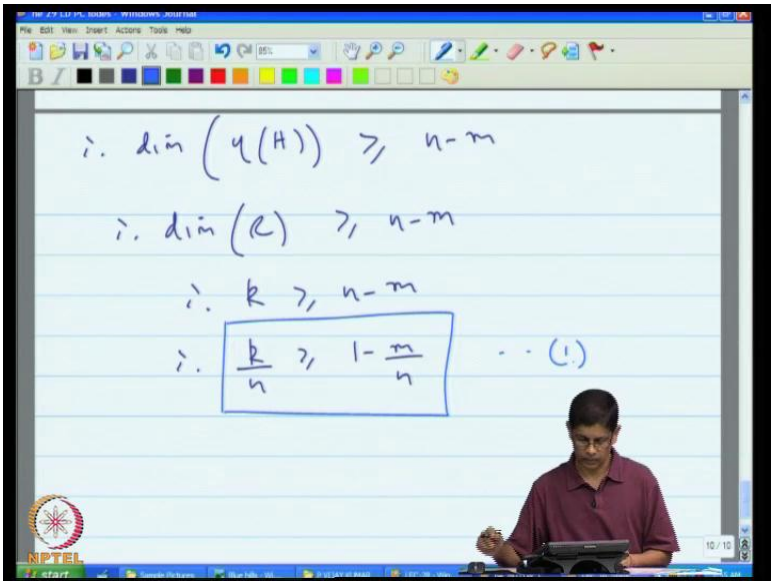
(Refer Slide Time: 24:48)



In this case, H is not minus k by n but it simply let us see that n by n matrix, that is reason as that because see this earlier then the rows are linear independent, then I could say that remember of

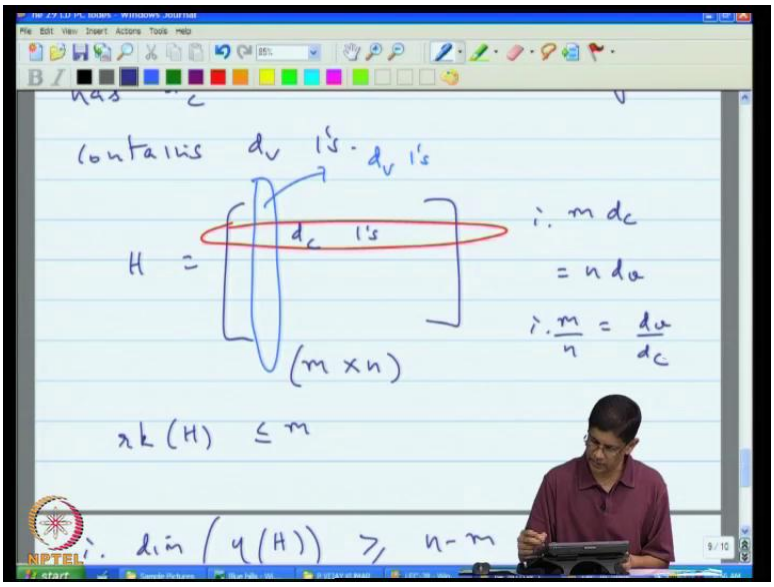
number of had to be n minus k, because the mention of null space k that the mention for code. But since, now relaxed the requirement device being linear independent, what this means, is that the rank of H is less than or equal to m. Therefore, the dimension of the null space of H is greater than or equal to n minus m therefore the dimension of the code is greater than or equal to n minus m.

(Refer Slide Time: 26:20)



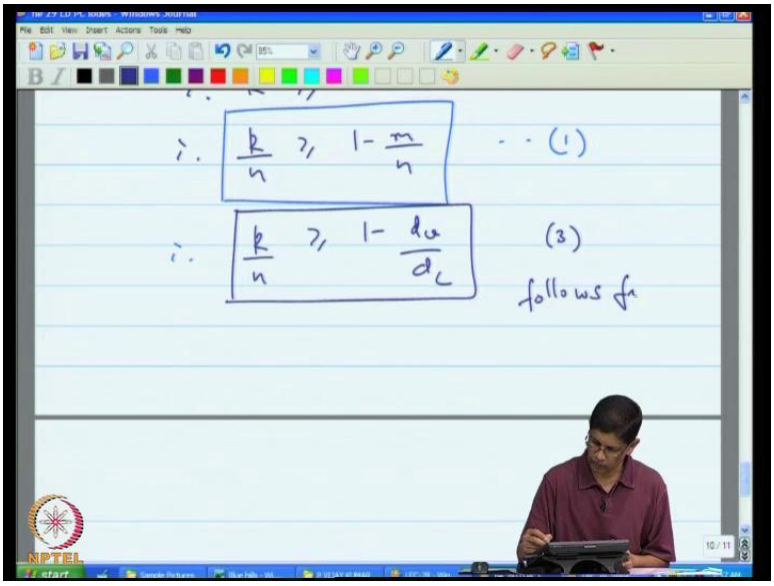
Therefore, k is greater than or equal to n minus m. Now if you talking about rate of the code k by n is greater than one minus m by n. Now, in the case of d v d c code, what will happens is that, every row of this matrix will contain precisely d sub c 1s. On the other hand, if you look at every column will contain d sub v 1s, but now you can to if total number of 1s either by counting row by row or counting column by column.

(Refer Slide Time: 27:32)



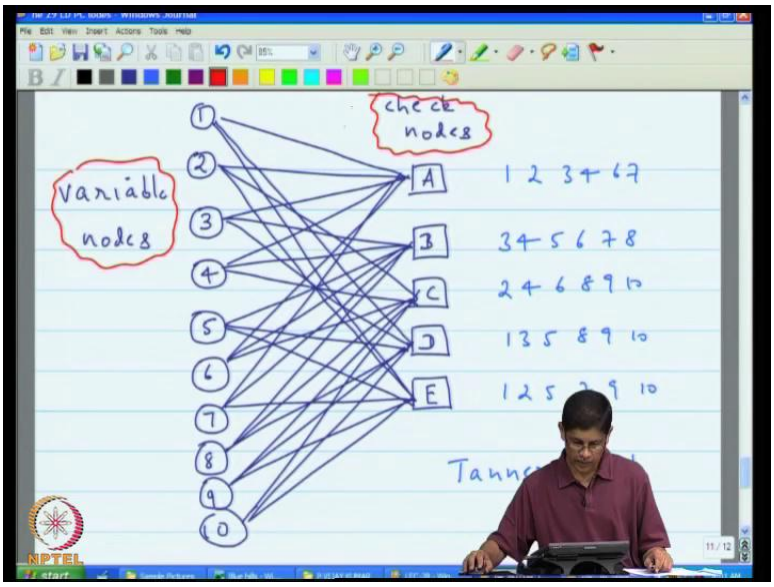
That means total number of 1's in the matrix and on the one hand, m times d c, on the other hand n times d v. Let me just put that down therefore, m times d c equal to n times d v therefore, m by n equal to d v by d c, this is the second equation.

(Refer Slide Time: 27:49)



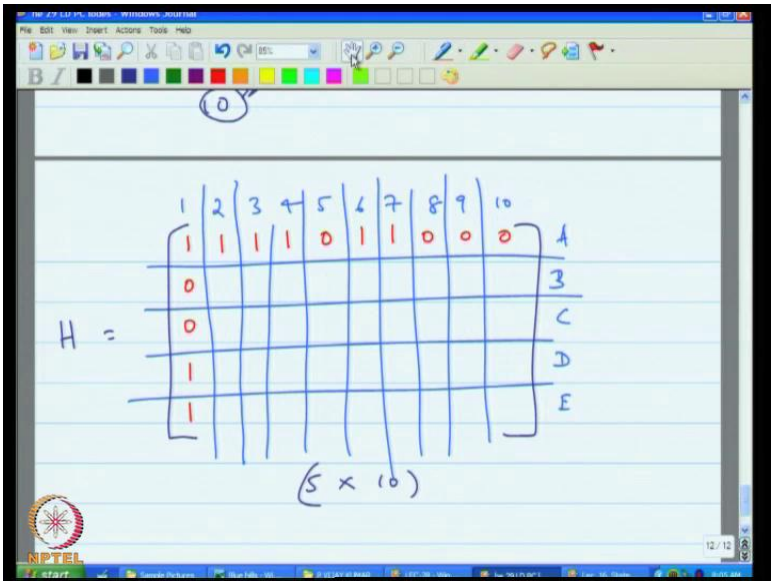
So, going back of here we say that therefore, k by n is greater than or equal to 1 minus d v by d c; this is 3, and it is comes from equation two.

(Refer Slide Time: 28:39)



Various representation of LDPC codes, which takes formula for graph. Let show this through example, I finish drawing this figure take about minutes of more and then I explain this graph here, represent the code and is called Tanner graph and what this graph actually telling you is that, on the left side and these nodes are called variables node and these nodes are called check nodes. And this is really another representation of parity check matrix.

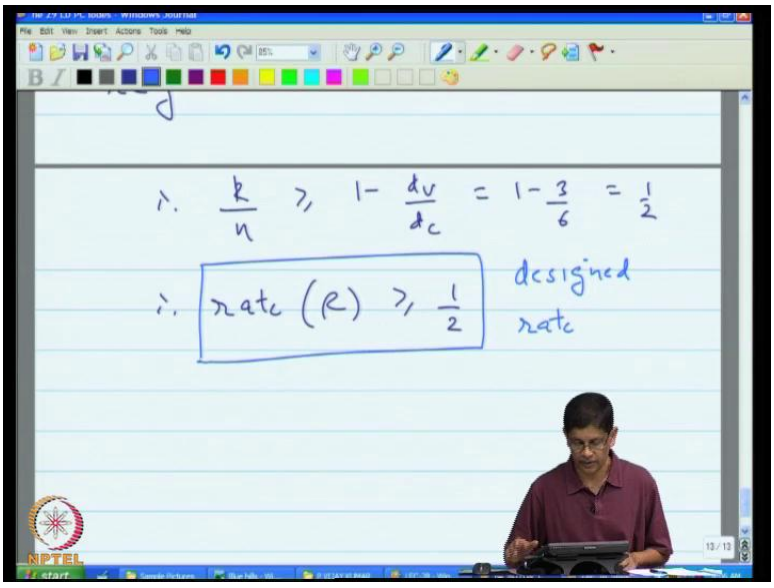
(Refer Slide Time: 33:08)



In the following sense, so you should thing of these as though, I will put this down on the next page. You should of in keep this is picture on your mind, there is the parity check matrix. This matrix that we will be called H is 5 by 10 matrix. The rows of the matrix are let say labeled as A B C D E and there are ten columns, so the column are numbered 1 2 3 4 5 6 7 8 9 10 and what this graph is telling you? The paths are should draw and that is matrix this graph actually telling you is that in row A, remember that the rows are corresponding with the letters and columns are corresponding with this number. So there are 10 columns and five rows, it saying that in row A, there are 1s in column corresponding to 1 2 3 4 5 6 7 and that is indicated by the line joining 1 2 3 4 5 6 and 7. Notice here that as far as every row is concerned every check node is connected to five variable nodes; and on the other hand, if you look at that every variable is connected to exactly three check nodes. So, as a result, let us try to fill in symmetric of the parity check matrix.

So, row A has 1 2 3 4 5 6 7, which means that we would have a 1 in 1 2 3 4 6 7, and then if you look in terms of across rows, where is 1 connected to? 1 is connected to A; it also connected to D and to E, the terms of column there is 1 here, here wherever the remaining symbols are 0. So, in this way you can fill in the matrix; and what is notice that this is the d v d c instances of d v d c regular code, because every row will turn out to have hamming weight six and every column has hamming weight three. This is an instance of d v equal to 3, d c equal to 6 regular codes.

(Refer Slide Time: 36:38)



Therefore, in terms of the terms of right of the code, we can actually say that the rate is greater than 1 minus d v by d c, which is 1 minus 3 by 6, which is equal to 1 half. Therefore the rate of the code of the code is greater than or equal to 1 half; so the reason enough able to be exact about it, because we did not prepare that the row be linearly independent. And so LDPC terminology, this is called the designed rate of the code and typically, people where could designed rate of the code instead of trying to figure out what is exactly is that rate.

This Tanner graph has a particular structure that the nodes on the left are connected to the links only nodes on the right and similarly, nodes on the right are connected through the links or edges only to nodes on the left. There are no links between nodes on the left or nodes on the right. So this is the instance type of the graph this actually known bipartite graph, so make a note of that. And now, we found other representation of the code, a graphical representation that is instead of defining the code in terms it is still a linear code; it is still has generator matrix; it is still has parity check matrix, but instead of describing the matrix in terms of symmetric.

We will have graph and this graph in tells you unlikely defines the code. So it tells you that the parity condition that must be satisfied there are five of them, for example B less down the rule that symbols x 3, x 4, x 5, x 6, x 7, x 8 together must have even parity. That is why I interrupt this. And in terms of graph, the fact that this is the d v d c regular code manifest itself in terms of the degree of the nodes. So every node on the right has degree equal to 6. So, the degree of the node is equal to the number of nodes to edge is connected to edge is links. Each node of right has degree 6, each node on the left has degree 3.

So, make note of that also, degree of each node on the left equal to 3; and the degree on the right equal to 6. And now you might ask, what is the reason for such restriction and I will try to alterative to be the partial answers. One thing you note is that if you look at total numbers of 1s in the parity check matrix, every one corresponds precise to the edge, because it is belongs to certain row and the certain column. So the number of 1s is precisely the number of edges, but the number of edges is equal to number of either the number of rows times degree of the node in the right are equal to number of nodes on the left terms of degree of the node left.

Now, since the degree is the fixed which is a small number of 3 or 6 and typically LDPC codes, when we talk about d v d c regular codes the block length could be very large, but we will still

fix this degree. So, what that means is that the number of 1s in the matrix is automatically some multiple of n, so it is order of n; for this reason, these are called LDPC codes. Let us note that the d v d c constrains implies that the total number of 1s in the parity check matrix is equal to n times d v is equal to m times d c and hence is order of n. Since, d v d c are fixed and independent of n.

If you think of the number of 1s as the fraction of the total number n in the matrix is order on dived by n square, which is very fast so that is called low density parity check codes. That is means terminology, but why did you want low density? While the thing is that, what you actually is going to do? You are going to decode this code by passing messages on this graph. You are going to pass messages from along the edge of the graph, and you are going to do this in the following way, you are going to start with the messages that are passed from left to right, followed by message that are passed from right to left and left to right and so on. Every time you start around of message passing from right to left followed by left to right is called one iteration.

So, this is an initial phase, which you go left to right, but then thereafter, every iterations constitutes round of message passing from right to left and back from left to right. Typically, number of iteration some fix number like 20, may be 50 depending on what level of accuracy required. So the number of operation and every time you pass messages, you do some operation at each of the nodes. So, the total number of operation is proportional to the number of nodes; proportional to the degree of the node and proportional to the number of iteration, but you fix the number iteration and since the degree of the node is fixed.

The total number of operations is linear number of nodes on the left, which linear length of the code. So that means that if you can actually decode this code simply by a sequence of message passing from left to right inverse. Then you can able to decode the code in linear complexity and that is the beauty of LDPC codes. Their greatest strength is the Es with which they can be decoded. And now you might think well, but does comes the prize to some extent, the answer is yes, I know but it depends on what you looking for. These codes in terms of minimum distance may not always be the best possible code that you can get for block length.

However, the maker of performance in terms of that for practical by the performance is pedicted not always, but for most instantaneous. So, this class of code is fast becoming what we people

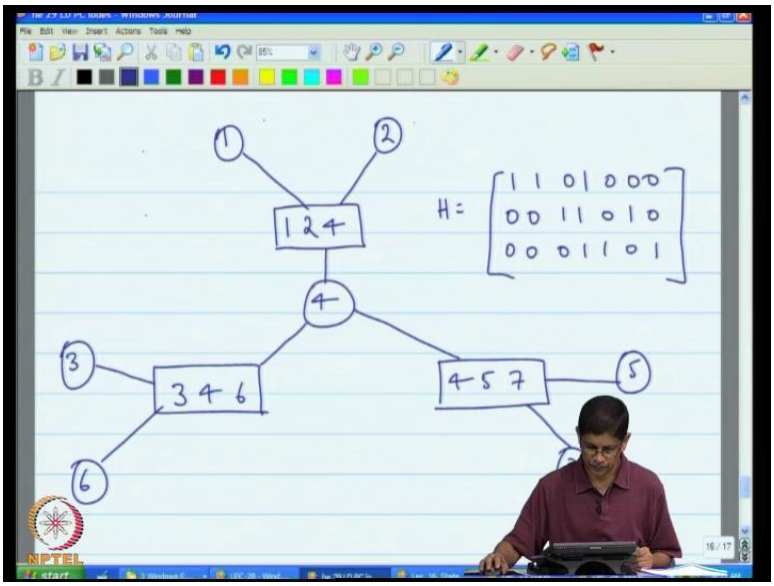
call the ubiquities the error correcting codes; where the codes should be used in all situation so, became very, very popular can be widely in use, because ease of decoding.

Let me just add a command to the effect that the graphical comma message passing nature of the decoding algorithm in the case with the LDPC code implies that the complexity of the of the decoding algorithm is proportional to the number of edges in the Tanner graph of the code. And hence is linear and I think it is with will putting some exclamation mark in the block length of the code. So, it is this fact is a vernal fact which makes this codes of the practical interest, but I might as well, what is the Tanner graph actually realize from? Tanner is called Tanner graph after the Mickel tanner, who is the coding theorist are if you see at the university of California system, who should this could actually, this codes have an interested history. So they were this algebraic and convolutional codes that where in existence uptil the late 80s until an early 90s.

The class of treatable codes are called (( )) codes surface time and the feeling to have super performance, and they were iteratively decoder, and along this time and there are group of people who discover this LPDC codes and sure, and they could also be decoded iteratively. And then people realize that these are the really codes, that were discovered back in sixties, in PhD theses in a beautiful PhD thesis of information encoding theorist Robert Gallagher. They were rediscovered appear in the ninety sixty, but the technology at that time was not ready to exploit the content of the thesis, so the Mickel Tanner was one of those who rediscovered LDPC codes, but he want little bit further and he actually, added some structure, he introduce this graphical interpretation of the decoding algorithm.

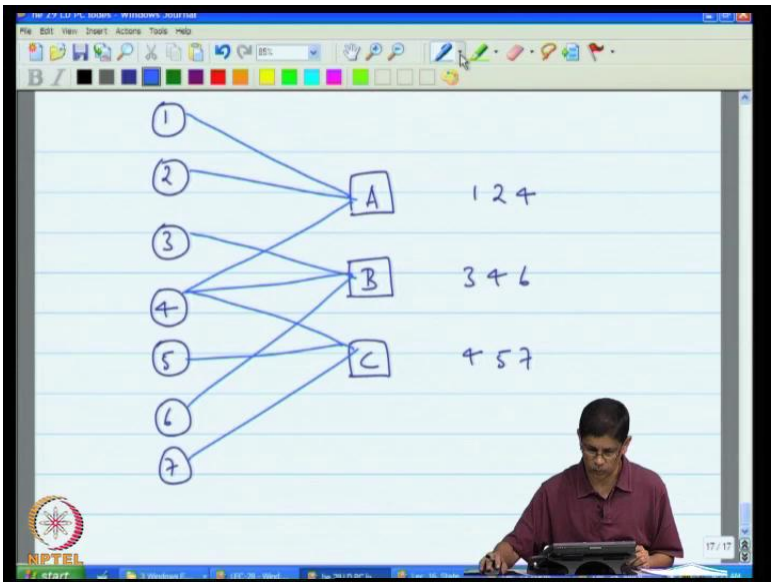
He contributed much theory and that is why these are called the Tanner graph. That now, it must, you must be wondering what exactly this message passing decoding algorithm like; and the nice thing about is that coming from our back ground, we just come off by looking at the GDL. So it turns out that the underpinnings of the algorithm from a certain view point, so difference people explain the decoding algorithm in the different ways, but I am adopting the view point adapt to some others and including the author is GDL. There is that the GDL is in some sense underpinnings of the message passing algorithm and I will make that in a little bit more conflict. If you recall our earlier lecture, we look that term decoding the 74 t code, 742 block code.

(Refer Slide Time: 51:20)



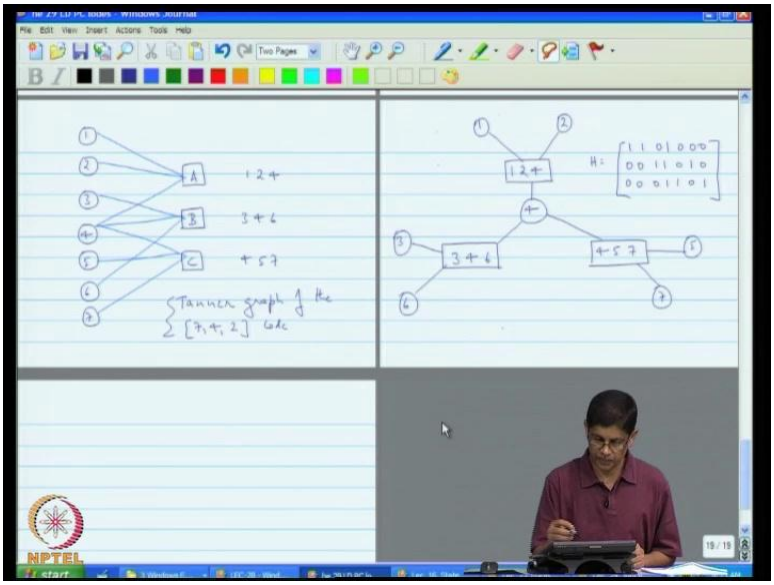
So let us, if you actually pull to up here, we go this looking fast, so let me, here is the junction tree is a junction tree of the 742 code. So what I want to actually tell you is that the same graph can be redrawn in a slide different manner. If you recall this particular code had parity check matrix, which is of this form. This parity check matrix has three rows, and each of this node, parity check is associated node in this graph; and there are seven codes in the symbol and each of them associated in node in the graph. I can redrawn this graph the following way, I can actually drawn in the fashion of Tanner graph.

(Refer Slide Time: 52:33)



I can put down variable nodes on the left. I can put down the check nodes on the right. So this corresponds to1 2 4 3 4 6 and 4 5 7. If I draw edges, like would the case in the Tanner graph so, this is the Tanner graph representation of our earlier 742 code. And the way you can actually thing about it let me show this to you side by side copy this page that might happen.

(Refer Slide Time: 54:15)



Here, we go so we can actually see this two graph side by side and when look at this, you can see that basically, you can integrate this as redrawing because 1 2 4 is A, 3 4 6 is B and 4 5 7 is C, here A B and C and it turns out that, it note to that. In fact, it turns out that there are various possible message passing algorithm that are adapted on the Tanner graph to the LDPC codes, but there is one, which goes by the name belief preparation, which the mimics more or less message, there are passed in the junction tree. In fact, you already have seen the instance of the kind of the message passed in this graph.

Anyway, since we have just about minute left, let me summarize, what we did it today? Look, was introduce new topic which is LDPC code; and I try to explain, why this are of interest? How the mean the low density arrived? How we can estimate the rate of the code, I introduce the graphical called the Tanner graph, and not only the message passed and this graph decode the code and also pointed out and this message and assumption are not completely new because the certain the topic messages passing similar to the message passing we encountered to the junction tree.