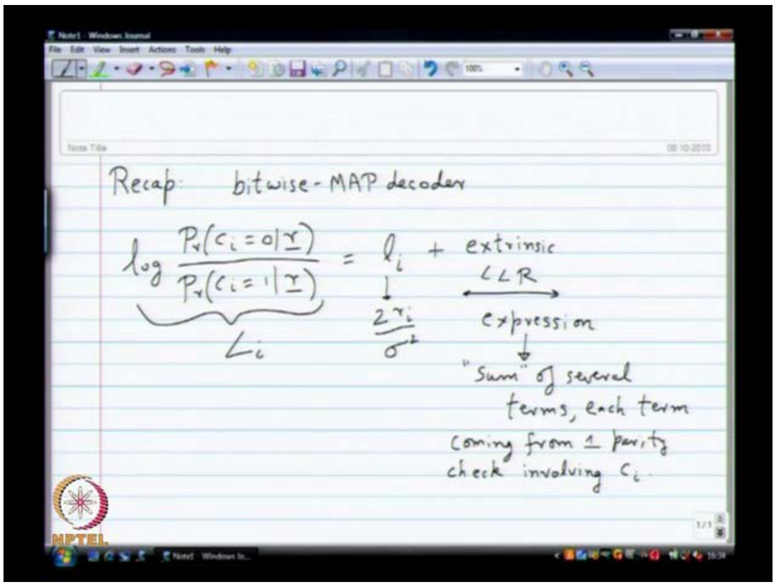
Coding Theory

Prof. Dr. Andrew Thangaraj
Department of electronics and Communication Engineering
Indian Institute of Technology, Madras

Lecture - 22
Union Bound, Introduction to LDPC Codes

(Refer Slide Time: 00:16)



Recap: bitwise-MAP decoder

$$\underbrace{\log \frac{P_r(c_i=0|\underline{r})}{P_r(c_i=1|\underline{r})}}_{L_i} = \underset{\downarrow}{l_i} + \text{extrinsic } \xleftrightarrow{LLR} \text{expression}$$

$l_i \downarrow \frac{2r_i}{\sigma^2}$

expression $\downarrow$ "Sum" of several terms, each term coming from 1 parity check involving c_i.

So, let us once again begin with a quick recap. So, almost the entire lecture last lecture we were look at the bit wise M A P decoder order. So, let me just quickly recap about what I spoke about this decoder. So, the essential idea is to compute what is known as the log likely hood ratio given the entire vector. Guess, I think I might have called it vector log likely hood ratio or given the entire vector. So, basically want to compute probability that a particular transmitter bit is zero given the entire received vector r divided by probability, when the breadth is one given the entire vector r.

So, log of this system the log likely hood ratio that you want to compute. So, which we denoted capital l i, for instance certain sort it can be write as some complicated expression, which will essentially factor as l i plus some extrinsic L L R. This l i is simply two times r i by sigma square, you can think of it as the scalar or the channel L L R for the i th bit. So, this extrinsic L L R of course, is the right the entire point of decoding or soft decoding is to compute the extrinsic L L R. Intrinsic L L R is just the channel L L R is something that is given to you. There is one complicated expression there is an expression for it.

Then we also saw how that expression can be written as a sum. So, loosely a sum, because its approximately a sum of several terms and each term comes from one parity check. So, each term coming from one parity check parity check involving the i th bit, so involving C i. So, of course, this is not true in general, there are mean you can add only when they are all independent. Each parity check will of course, give you some information about the i th bit, but they are all not independent. So, you have no business really adding. Then you have to account for how they are dependent, but a good approximation like I said it seems to be in practice, that you can add them without worrying about any of these things.

So, that is the that is the basic description I gave for the bit wise M A P decoder, none of this part was very rigorous. Those mostly heuristic we were looking at it from a from simplifying expression point of view. We saw mostly examples or three examples and some cases where it worked some cases, where it did not work. Of course, there is lots of theory behind these things. Then we do not have too much time to go into all those theory, but of course, there is practice is what is most important.

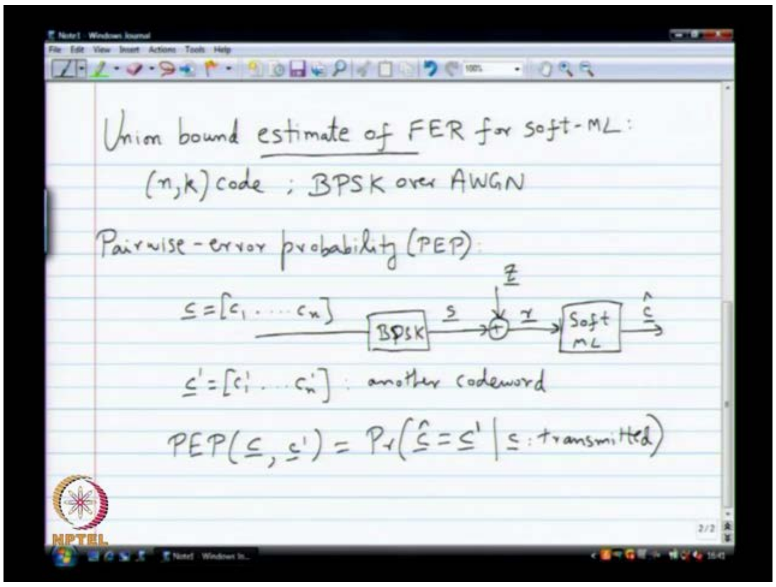
We saw how we can do simulations to verify, whether you are getting any kind of coding. We saw some picture and some definitions of coding again for classical codes. We saw that it was there is a large gap from capacity. So, one thing I want to alert you to, which I did not may be paid too much attention last time, was the difference between bit error rate and block error rate. So, when you do not do coding you have to pretty much decode on bit at a time.

So, your bit error rate block error rate and all coincides, because your block is simply one bit. It does not make any sense to do anything else, but when you do coding you will get a bit error rate. You also get a block error rate and both of these will not be the same. They will not be equal in fact there are bonds relating the two, but usually the bit error rate will roughly be block error rate divided by something. So, you have to think of means there is some division by the number of bits per block going on there. Depending on how many bit errors you make for a given block error the two things will now will be related in a trifle fashion. You will get different kinds of curves you have to pay attention to that.

So, normally as a metric it will be mostly interested in bit error rate or block error rate. What do you think is a good metric in practical bit error rate block error sometimes it depends on the application, but usually block error rate is the real goal standard. I mean you have to worry about entire frame being wrong, but then the other argument for that is may be your message is not very heavily compressed. May your message is some big b m p file that case a few errors here. There does not really sacrifice on the quality of the final experience, but never the less it's important to know block error rate. In most cases particularly if you have a big communication system, which has multiple layers I mean these days communication systems are built with layers.

So, the link layer is something, which will have all this coding. All that above the link layer there will be some other layer, which will only expect a certain block error rate. So, depending on the block error rate the efficiency of the higher layers will be disturbed. So, block error rate becomes quite crucial in practice. So, we have to next move to a low density parity check codes, but before that I want quickly point out one thing, which we missed, when we looked at the soft M L decoder.

(Refer Slide Time: 06:19)



Union bound estimate of FER for soft-ML:

(n,k) code ; BPSK over AWGN

Pairwise-error probability (PEP):

$\underline{c} = [c_1 \ldots c_n]$

$$\begin{tikzcd} {} \arrow[r] & \boxed{\text{BPSK}} \arrow[r, "\underline{s}"] & \oplus \arrow[r, "\underline{r}"] & \boxed{\text{Soft ML}} \arrow[r, "\hat{\underline{c}}"] & {} \\ & & \underline{z} \arrow[u] & & \end{tikzcd}$$

$\underline{c}' = [c_1' \ldots c_n']$: another codeword

$PEP(\underline{c}, \underline{c}') = P_r(\hat{\underline{c}} = \underline{c}' \mid \underline{c} : \text{transmitted})$

So, we going to go back to the what is known as the union bound estimate, estimate of what? Estimate of block error probability of say F E R for soft M L decoder. Such king of long title that the its essentially, what we are going to do next. So, I was making this statement that M L and M A P d coders are quite complicated. You cannot do accurate analysis accurate analysis becomes difficult for the repetition code. It was easy for even other codes there are too many correlations you are computing in soft M L. They are all dependent in a crazy way you could do it, it’s very difficult to define those areas precisely. So, this is actually a recurring theme in many areas not necessarily coding and decoding.

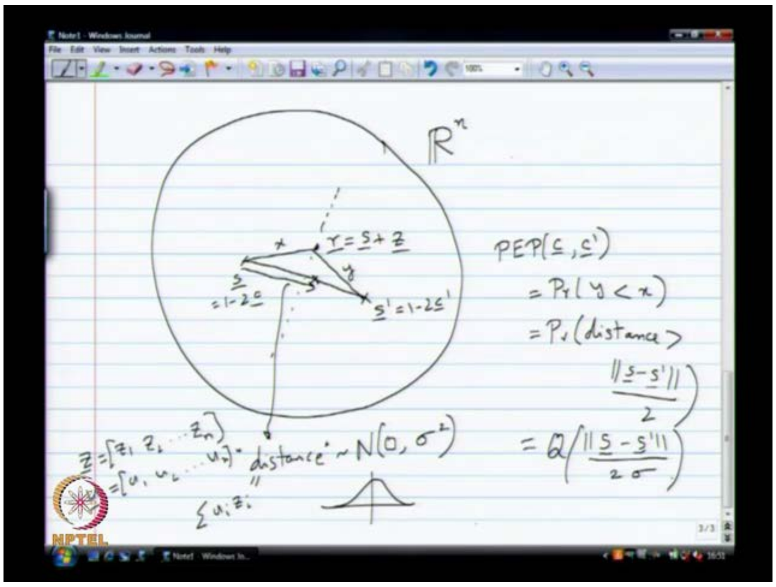
In many communications whenever you do soft decoding accurate estimation of block error probability or frame error rate and all is very hard. So, what most people resort to is some kind of union bond estimate. At the heart of the union bond estimate is something know as pair wise error probability. So, it turn out in most cases you can always compute pair wise error probability very easily. It is a very easy thing to compute there are only. So, it’s like there are only two decisions, when you say pair wise error probability you have only two hypothesis. Either, it is this or that those kind of regions are very easy to compute error probabilities. When you have multiple code words and you can go wrong in multiple ways. Then those regions will interact in strange ways and you cannot compute exact. So, let me begin by this description of what is pair wise error probability we will see what I mean.

So, once again we are going to think of an n k code n k code where I am going to use B P S K over A W G same as before, nothing is changed. So, suppose it transmit the setting for pair wise error probability is as follows pair wise error probability or abbreviated as P E P. If you want this is the settings of pair wise error probability you transmit a particular code word, for the liner code case we fix it to be the all zero code word. Let’s say we transmit some particular code word this is the code word that is been transmitted it gets gets converted into some symbol vector noise gets added to it. You get a received vector r and the soft M L decoder is working on it and it produces some C cap.

So, suppose let us say C prime is another code word the pair wise error probability between C and C prime is basically the probability that C hat equals C prime. So, given that C was transmitted pair wise error probability between is probability that C cap equals C prime given that C was transmitted.

So, is that you are staring at it like I have made a mistake what is the what is the question what is the error you say. So, this is I mean probability that the soft M L decoder decodes to some other code word C prime C prime, C hat equals C prime given C is transferred. So, here this probability it turns out its quite easy to evaluate. I will show you how to evaluate this is very it is a very simple expression, it is just one q function. So, it may not seem like it at the outside, but it is a very simple one q function. Again, the geometric u will help you here. So, we have this big space of let us say this is r n and you have s, which was transmitted.

(Refer Slide Time: 10:59)


Note1 - Windows Journal
File Edit View Insert Actions Tools Help
$\mathbb{R}^n$
$\underline{r} = \underline{s} + \underline{z}$
x
y
$\underline{s}$
$= 1 - 2\underline{c}$
$\underline{s}' = 1 - 2\underline{c}'$
$\underline{z} = [z_1\ z_2 \cdots z_n]$
$= [u_1\ u_2 \cdots u_n]\cdot$
distance $\sim N(0, \sigma^2)$
$\sum u_i z_i$
3/3

$PEP(\underline{c}, \underline{c}')$

$= P_r(y < x)$

$= P_r\left(\text{distance} > \frac{\|\underline{s} - \underline{s}'\|}{2}\right)$

$= Q\left(\frac{\|\underline{s} - \underline{s}'\|}{2\sigma}\right)$

You have let us say another vector S prime, which is what 1 minus 2 C prime S is 1 minus 2 C you transmitted s, what happens in this R n z gets added to s. You get R some r you get here and say you get an R here this is S plus z. The question I am asking essentially is when will R be closer to S prime than it is to s, this essentially I am asking the question. When is this distance let us say this is y this is x when is y greater than s.

We essentially say must probability at y is smaller than x. So ,what you can do for this is to look at this line, which joins S and S prime and project. What project, so you basically you have to look at the projection of R on this line. So, it is enough if you view that. So, in fact the region will basically be the bisector of this line it is all n dimension. So, I cannot just say bisector means there will be some n minus one dimensional object, which separates these two things huge thing in the middle. You have to only ask the question, whether R is on this side or that side. So, all you have to do is to look at the vector r minus S and project it on to this.

This line joining S and S prime and see if it has crossed half the distance. So, you can also show that the projected point will basically be S plus z prime. The z prime will also have the same well. It is just, so let me re do let me re do this carefully. So, let me not say this, so this distance this distance on this line is just one value. It is not a vector any more, it is just one value this distance you can show is normal with what mean being 0. Variance is the same sigma square that you had for z.

So, you would have a variance for sigma square z. Let me make sure I have that I think that is correct. So, this distance would have, so basically what you have to do is when you project r minus S on to this line. The projection of S will just basically bring it down to S its almost like S has become the origin. So, then only thing you are doing is you are projecting z on to this when you project z on to something. You can think of unit vector there, it is just when you multiply it with the unit vector. You will get one random variable with the same variance as the as sigma square. So, you can show this it's not too difficult. So, it will be n by sigma square essentially this probability, now becomes the same as probability that this distance. So, to speak this distance from S is greater than S minus S prime by 2.

So, that is all that we have to worry about. So, there is normal distribution 0 sigma square, when will that be greater than, did I make a mistake somewhere? Made a mistake, that is S n S prime. That is the distance S minus S prime no repeat the question again, no there is no condition that the projection has to lie between S and S prime. It can be outside of S prime of that thing also the thing of, so basically it is almost A B P S case happening over this huge dimension between this S and S prime. So, it is as good as that, so you have S you have only one line to S prime. Any error that you take go in several other dimensions ultimately gets projected on to that one line.

So, it is almost like B P S K between S and S prime. It is only the distance S and S prime matters, you haves the same variance for that projection of any n dimensional normal operator. Also, you can show that that is not very hard see the projection will also be Gaussian. It is a linear combination of several Gaussian things, it has a unite norms. So, you will get something similar to that any way. This is the actual that is why I put norm between S and S prime this will be like and distance square root of between C and C prime. I mean S and S prime, but you have a distance in the mean as zero. It can be negative also, see its normal distributed with mean zero and sigma square that normal distributed value can take negative value Gaussian.

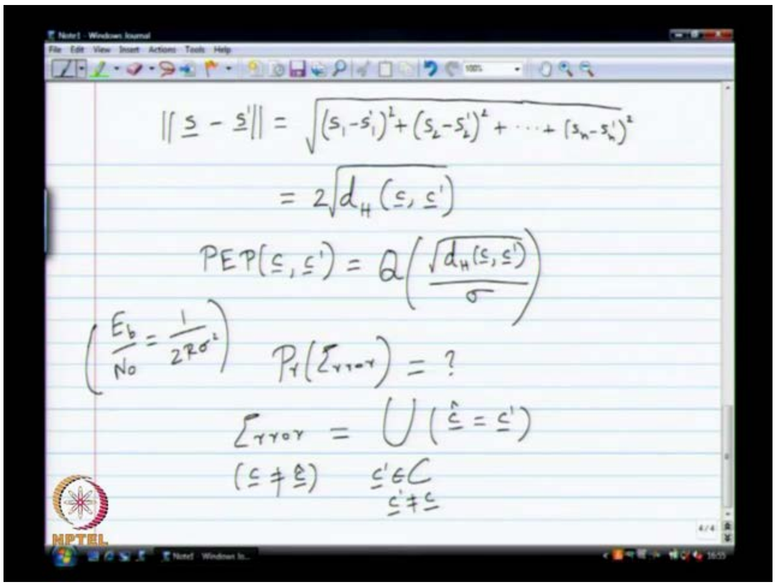
So, this system I am saying is basically distributed as like this. So, it can be both positive and negative. If it goes negative, then you decide S it is like B P S K you transfer plus 1 and minus 1 if you get plus 2 you decide plus 1. So, this distance when I define I will define from this in the in this direction. If it goes negative it will be minus you extend the line. You have to extend the line on both sides I did not expect this to be so controversial. So, we are taking distance with sign, it’s with sign. Maybe, I should not put both the arrows. So, maybe I will put just this arrow in this direction. So, if it goes to the average direction I will make it negative. So, it’s clean believe me it is a very clean idea its correct it is not wrong.

So, very simple thing see this is why essential par wise probability is easy in many cases. So, you have one point you have another point in the straight line joining these two. You are just dealing with one dimensional stuff nothing happens beyond that the projection can take you in this side. It takes you to that side definitely not going to make an error be correct, can also take you to other side of S prime. Then you will definitely make an error that is depend on the distribution is that, but only the projection matters. That is the thing you have to be convinced of.

So, when you do projections, so z is going to be, so let us say z is some. So, this question is being asked. So, basically if you do z, z will be z 1 z 2 z n. Each of these things is normal with mean 0 and variance sigma square. When I project on to anything I will be taking a dot product with some unit vector, its norms one, let us say the unit vector i picked is u 1 u 2 u n.

So, the distance will essentially be summation u i z I it has mean zero. Its variance is summation u i square that is one that all is that it will also be normal, because it is a linear combination of a joint legation random variable random vector. So, we got an up to this point and this is normally distributed. So, this simply becomes a q function q of norm S minus write this carefully q of norm S minus, some way some I think I should time has come to quite. Do not think that conditioning is on in spite of this assurances definitely shown its of, so let us say S minus S prime by 2 sigma that as simple as that. Now, what is S minus S prime?

(Refer Slide Time: 20:34)



$$\|\underline{s} - \underline{s}'\| = \sqrt{(s_1-s_1')^2+(s_2-s_2')^2+\cdots+(s_n-s_n')^2}$$

$$= 2\sqrt{d_H(\underline{c}, \underline{c}')}$$

$$PEP(\underline{c}, \underline{c}') = Q\left(\frac{\sqrt{d_H(\underline{c}, \underline{c}')}}{\sigma}\right)$$

$$\left(\frac{E_b}{N_0} = \frac{1}{2R\sigma^2}\right) \quad P_r(\mathcal{E}rror) = ?$$

$$\underset{(\underline{c} \neq \hat{\underline{c}})}{\mathcal{E}rror} = \bigcup_{\substack{\underline{c}' \in C \\ \underline{c}' \neq \underline{c}}} (\hat{\underline{c}} = \underline{c}')$$

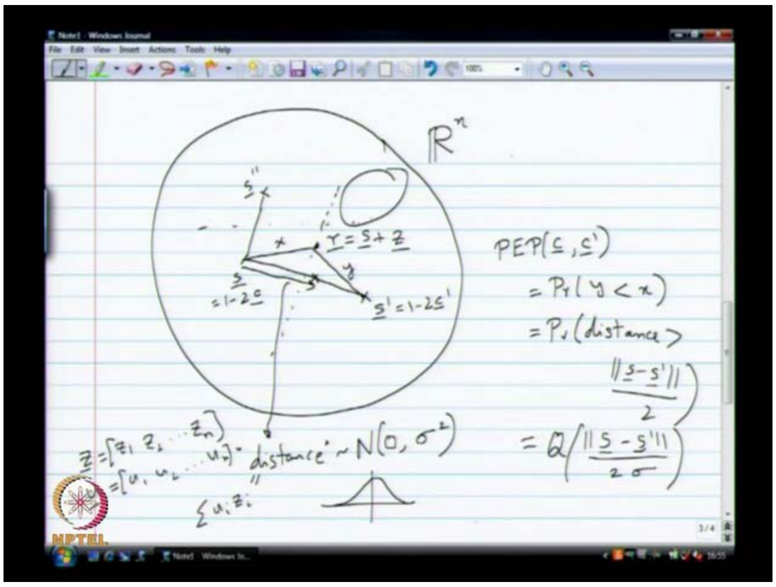
Let us look at S minus S prime bit more closely S minus S prime norm is square root of, what it is going to be summation a do a long square write it down. S 1 minus S 2 S prime 1 square plus S 2 minus S 2 prime square plus so on till S n minus S n prime square. So, it turns out if you, now know the hamming distance between C and C prime. You can use that to write this precisely. So, after all you are doing B P S K. So, you know S S 1 and S prime for instance for plus ones and minus ones. If the two code words did not differ in that first position they will be zero. That difference will be zero if it differs it will be two.

So, it basically be two times the hamming distance between C and C prime is that every time it differs you get a 4 inside the square root. So, you take every time it differs you get a 4 incentives values I thought this might be the root for that, is that fine? So, this is what you get for the S minus S prime. So, essentially par wise error probability between C and C prime simply becomes remember this is B P S K with minus 1 plus 1. If the minus 1 plus 1 is not there will be some nasty constants floating around, there does not change the final answer there. So, when you see the final answer ones I expressed it in terms of E b over n not final answer becomes exactly this.

So, that is the idea q root of d H C, C prime by sigma is that. So, remember 1 by sigma, if I want to write it in terms of E b over n not E b over n not for our case is what is 1 by 2 r sigma square r, remember is k over n. So, you will get n over k there. So, if you have to write it in terms of that you will get some square root etcetera and inside. So, you can express it in terms of that but the next interesting thing is suppose i want to write probability of error using par wise error probability. Wow, people have some readymade answer, let me let me go through this lit it bit more slowly. You can use something known as union bound.

So, it turns out turns out the idea you use there is the error. That event that you have what is your error event C not equal to C cap. You can write it as union of all the C prime and C that is not equal to the transmitted code word C C hat equals to C prime. Remember, its union and it is not a disjoint union. It will not be a disjoint union, C prime C hat equal to C prime. We saw it was just on one dimension, if you go to an another dimension, if you go back to this picture.

(Refer Slide Time: 24:12)



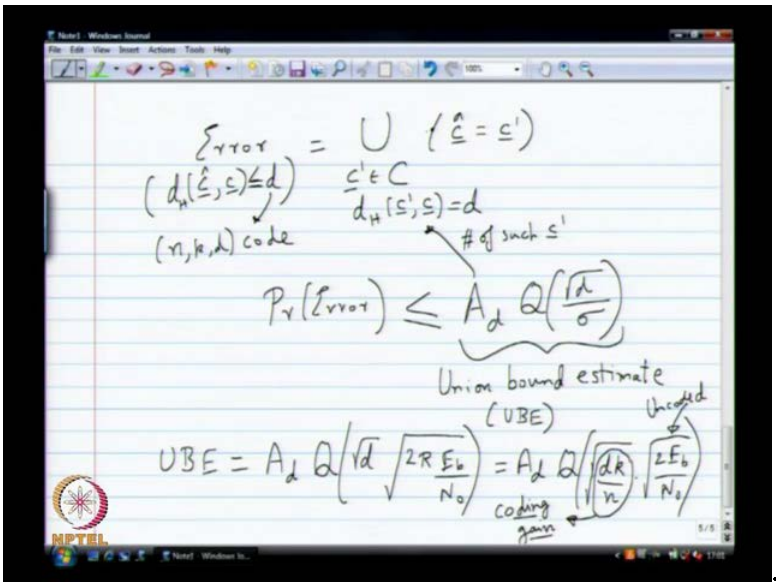
$\mathbb{R}^n$
$\underline{s}''$
x
$\underline{r} = \underline{s} + \underline{z}$
y
$\underline{s} = 1 - 2\underline{c}$
$\underline{s}' = 1 - 2\underline{c}'$
$\underline{z} = [z_1\ z_2 \dots z_n]$
$= [u_1\ u_2 \dots u_n] \cdot$
distance $\sim N(0, \sigma^2)$
$\sum u_i z_i$

$$PEP(\underline{c}, \underline{c}') = P_r(y < x) = P_r\left(\text{distance} > \frac{\|\underline{s} - \underline{s}'\|}{2}\right) = Q\left(\frac{\|\underline{s} - \underline{s}'\|}{2\sigma}\right)$$

If you go along with some other direction to some other let say S w prime. If you draw the bi sector there all these regions will overlap. It’s a proper union its not a disjoint or anything. So, you cannot simplify it in any way you like. So, you can only say that the probability of error here will be less than or equal to 2 par k minus 1. Like this there was 2 par k minus 1 here. The par wise error probability, but if you do this 1 par k minus 1 all that you would not get anything reasonable. So, it turns out the union bound estimate is works very well to restrict yourself only to closest code words. You can do that it gives you a reasonably a good estimate in many cases.

So, what do you mean by closest code words? The code has a sudden minimum distance, some minimum distance d. Let us say the code has the minimum distance C. You only have to look at those events, which are very likely to have happened, which means you only have to look at those code words, which are at distance d from my transmitter code word. I will take a union only over that then do a union bound with that, so it not clearly not the completely error event, only something like a closest code word error event. So, you define some other error event, which is may be some I will simply call it error again, but remember this is not the same error as before.

(Refer Slide Time: 25:44)



$$\mathcal{E}rror = \bigcup_{\substack{\underline{c}' \in C \\ d_H(\underline{c}', \underline{c}) = d}} (\hat{\underline{c}} = \underline{c}')$$

$(d_H(\hat{\underline{c}}, \underline{c}) \leq d)$

(n, k, d) code

# of such $\underline{c}'$

$$P_r(\mathcal{E}rror) \leq \underbrace{A_d\, Q\left(\frac{\sqrt{d}}{\sigma}\right)}_{\text{Union bound estimate (UBE)}}$$

$$UBE = A_d\, Q\left(\sqrt{d}\sqrt{2R\frac{E_b}{N_0}}\right) = A_d\, Q\left(\sqrt{\boxed{\frac{dk}{n}}}\cdot\sqrt{\frac{2E_b}{N_0}}\right)$$

coding gain

Uncoded

.

This one is C hat distance between C hat and C is equal to d, where d is the minimum distance of the code. So, this error event is simply union over what C prime and C. Such that the distance between C prime and C is exactly equal to d. Then you simply take the same event C hat equal to C prime. So, may be here I will put less than or equal to d just to say these are the code words within d i. Only have to look at those code words, which are exactly at distance d, because I know my codes minimum distance is d nothing else, in between 0 and d. You have to make go d distance all that. So, here once again when you compute this, probability of error you cannot do exact computation.

You have to once again do the union bound, but then you get a much better factor multiplying. It is not 2 power k minus 1 all the code words are not going to be a distance d away. May be I will denote that by A d, A d is the number of such C prime you can show that this a d will be independent of C for a linear code. Any C you take you will have the same number of code words at a particular distance away from. So, it is very easy to show that for a linear code this will be true.

So, I do not have to qualify it with C it is just A d it is enough times the par wise error probability at a distance d. What is the par wise error probability at a distance d square root of d divided by sigma? So, this guy is usually called the union bound estimate. It is a pretty good estimate for the soft M L decode. So, that gives you a nice comparison of various parameters C. So, the union bound estimate has abbreviate as U B E. So, let us do this U B E little bit more standard notation see this basically A d times q of square root of d times one over sigma, which in novas square root of two times n by k. We get that k by n is it. Let me just keep it in terms of R 2 R E b over n naught am I right?

So, this is square root of 2 R E b over n naught. Lets write this as A d times q of root of d times r what is d times r d k by n times square root of 2 E b over n naught, why did I split it like this? D k is the only thing that we show of for the un coded case. If you do not do any coding what you will get is simply square root of 2 E b over n naught. So, if you do coding that square root the what inside the square root to E b over n not gets multiplied by D k by n. So, roughly the coding gain according to the union bound estimate is simply D k by n from here.

You can read of the coding gain is D k by n yes or no? Think about it. So, this guide simply becomes the coding gain according to the union bound estimate very rough estimate, but the nevertheless it is a good thing to have what it particularly brings out in my opinion. At least is it brings out the ultimate dependence on union on minimum distance you can go to B P S k A W G.

You can do the best possible decoder i, I was commenting that actually in this soft M L decoder does not have an error correcting capability. In the way we design define for the B S E, but that does not mean minimum distance is not important minimum distance is ultimately definitely important. Even in the union bound estimate it plays a direct role in determining the coding gain. So, that plays a interesting I am sorry can it be greater than. I think bounce no C d is less than or equal to n minus k plus 1. So, I think 1 minus r.

So, I do not know it can be greater than, so we look at it what is crucial? Here, once again is usually if you fix usually k by n will be fixed. When you go to large, when you go to larger and larger block length, you want to go 1 minus A by n. So, it always something 1 minus A. So, its lets than one square root of that will be. So, any way some discussion of D k by n of what it can be. So, it's not so important, it is important. Let me let me point out something else, this may also be interesting. So, you cannot ignore A d for instance

So, if A d is really large then it can play a role it can play a role. The way you have to read this expressions is to think of keeping K by n fixed, then going to larger and larger n, so that makes sense. In practice you know I mean you have a fixed rate the rate is may be controlled by several other things. You cannot change that and then you want to go to larger and larger block lengths min. Basically, used train used better and better codes you know when you go large block lengths codes become better extracts use better and better codes. Then how what kind of coding gain will you get can be a question that you can ask.

So, for instance as you go to larger and larger n you are A d will also keep increasing as a function of n. So, if A d is really large its increasing very large very quickly as function of n. Then also you will be in trouble that is that is also nicely brought out by this union bound matters is D, D controls lot of the gain, but also A d matters. If you have too many code words nearby somewhat you make an error.

Somewhere, it's good not to have too many code words of the minimum distances. Even that is that plays a fairly critical role here. Any way this is the union bound estimate and this is pretty much I can recall pretty much the only analysis. You can do for soft M L decode, so very difficult to do any other kind of analysis.

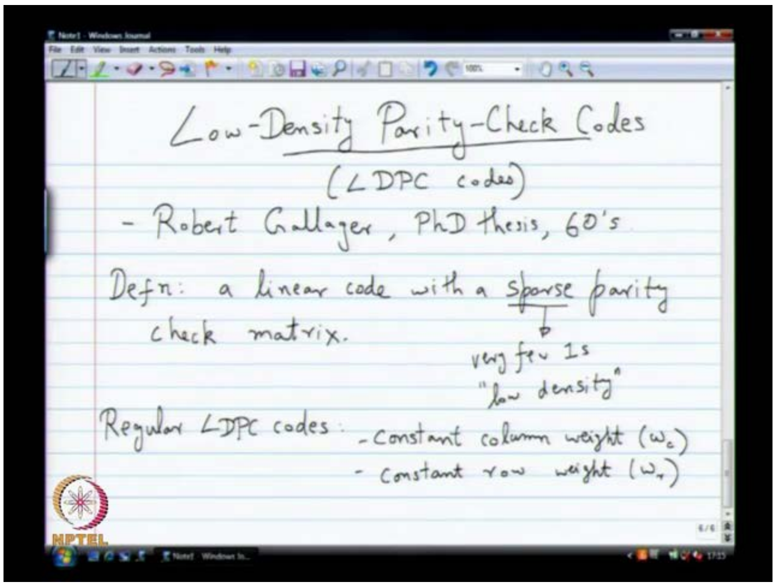
So, this is an important breaking point in the course we are stopping at a critical point. We are doing a major change of course beginning in the next from the next few minutes on wards. So, this is the stopping point for stop soft decoding coder modulation and soft decoding. How do you combine codes with a simple modulation scheme like B P S K? Of course, if you have a more complicated modulation scheme like 16 Q A M things become a little bit more complicated, but ultimately what matters is how you can compute log likely hood ratio.

That is the only thing that matters you might have a 16 Q A M constellation, but it does not matter. Each constellation point corresponds to 4 bits. Even that you receive a particular value how do you compute log likely hood ratios for each of those bits is long is, you can compute log likely hood ratios. You can go into your soft decoder and decode. So, technically at least there is no significant challenge in going from B P S K the 16 Q A M. As far the decoding is concerned, find the L L R and then you decode that all no big deal no confusion. That is why B P S K is kind a enough for a first course to understand B P S K, you can understand other thing.

So, the main themes was themes, where there are optimal soft decoders. In the most cases they become very tough to implement the trick is to understand them, understands their expressions and see how they can be simplified. This turns out bit wise M A P decoder as a particularly nice way of simplifying in to something nice based on parity checks through one parity checks, after the other seem like. So, that kind of decoding is like a mention its used a lot in this low density parity check codes. That is what we are going to start.

Now, if you have any question in this, now it's a good time to bring it up with any commons on anything. We saw on the soft decoding part, because this part is something that I can actually ask questions in the exams. So, once you see once you go to L D P C codes, it is really hard to ask questions on those things. So, this might be a very good point for the n semester exams. So, this make a note of this, so we are we are going to move to low density parity check codes.

(Refer Slide Time: 35:51)



Low-Density Parity-Check Codes

(LDPC codes)

- Robert Gallager, PhD thesis, 60's

Defn: a linear code with a $\underset{\substack{\downarrow \\ \text{very few 1s} \\ \text{"low density"}}}{\underline{\text{sparse}}}$ parity check matrix.

Regular LDPC codes:
- Constant column weight (w_c)
- Constant row weight (w_r)

So, the abbreviation is L D P C some people who are very grammatically correct and like good grammar will put hyphens here. So, if you are like me, then you will put hyphens in some places and you would not put hyphens in other places its I think anyone anything. So, these codes were actually invented by Robert Gallagher as popular you known in his P H D thesis guess when sixties. Let me say early sixties we do not know the exact year it was born in his P H D thesis and early sixties. Because, of various technological limitations and implementations etcetera these codes were not people did not look at it for a long time. Now, of course, they are get making their vain to many standards wireless.

Otherwise, when f d standards pretty much these days at least there is one proposal for L D P C codes. Many of the standards getting to some they may not get into others, but is a very popular codes today. So, the definition is so simple that I could have made it in second or third class will be had. We do not have to wait for so long to define low density parity check codes. So, it’s very easy definition is basically a linear code with a what is known as a sparse parity check matrix, whatever mean by sparse very few once.

So, very few once, so this motion of sparse is also basically low density. So, the density of once you can density of once if you like very regress sly. You can say number of once divided by number of rows times number of columns, that this has to be low. So, as an smaller than one much smaller than one. So, you can define all these things regress sly, but I guess the idea is basically sparse.

So, this is the main idea and this is linear code. So, while the definition is not restrictive all is not does not say what is the code. I mean if you compare this with the definition of red solemn code. I wrote down a very precise parity check matrix you might say what is alpha. We do not know what alpha is still, but whatever alpha is, its a very precise well defined quantity in some finite field, there is a very precise parity check matrix. So, here there is a definition, which is totally I mean totally open anything can be low density parity check code.

Looks like I am very specifies the code, how do you know of its a low density parity check code or not? It’s tough to find out how do you know it does a sparse parity check matrix or not? Remember, sparse parity check matrix means see each row of the parity check matrix is the code word of the dual, when I say sparse, I want very few once.

So, if I want a low a low density parity check matrix I have to find low weight codes of a code. May be the dual code in this case, but you have to find the low weight codes of the code, like I said long time back finding low weight code words. This is the hard problem it is not very easy given a parity check matrix. So, you cannot even be very sure if a given code is low density parity check code or not that kind of definition it is. So, the only thing you can do is opposite you construct a parity check matrix, which is already low density.

So, you put very few once into your parity check matrix that is what we usually follow. That is what implied by this, but another thing you will see as you go along as you will not really worry about minimum distance or trying to prove minimum distance will be concerned about it. The construction we will take care of it and in different way, but it will not be the explicit worry the most. If you remember the re solvent construction we were very worried about error correcting capability. How do you correct a given number of errors how do you construct a parity check matrix `were a given number of errors can be corrected here.

That will not be the point we will just construct a will try an decode come with a decoder. We will de code, where we will evaluate the decoder will basically be by simulations. There is some analysis, which we will also touch up on, but mostly min mostly it's done by simulations that. So, it will work in many ways this will be very different from the way I treated re red solemn.

You see the closest connection that can be made in terms of motivation. So, even though this was born completely in a different context, when you introduce it into a course you want to kind of make it fit it with something else. You want to say all this happens. So, because of that this is great the best way to tight up with that is may be the way you split it up in terms of parity check matrices. You remember I made an argument saying low weight low weight parity checks are good for you.

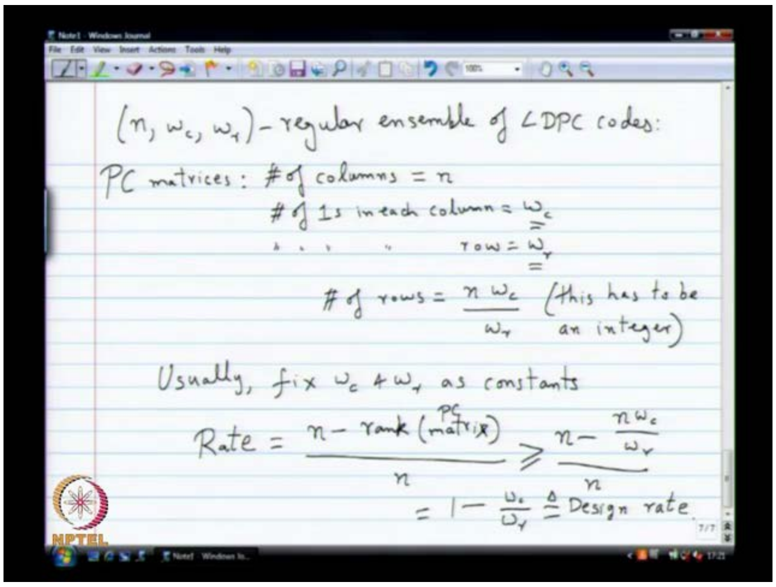
In the way you split it up and then another motivation for sparseness is. Suppose, I take two parity checks that are very sparse its very lightly, that they do not overlap. You do not have anything in common, which means the extrinsic information, that they give you are its going to be independent. You keep on adding more and more very easily.

So, if you have explicitly in your parity check matrix you have a good starting point, that is the idea. That is what kind of make sense at the end, but of course this is not how we started, but we can justify it. Justify the popularity or the reason it becomes so nice by all these other connection with the bit decoder and motivations from it. So, anyway this is the general definition that are specific classes of the L D P C codes that are much more that are that are actually popular.

So, which are a bit most specific you can actually construct them, but still not totally specific has not they would not specify a single parity check, but at least there will be a class or un solemn of parity check, which will behave in a same way, according to some definition. So, that is that is definition we are going to do now. The first class is so called regular L D P C class. So, in fact Gallagher ancestries are introduced to only regular L D P C codes. It did not introduce the other kind of L D P C codes, what will be the other kind of L D P C codes that are not regular irregular L D P C codes.

In fact regular L P D C codes are special case of irregular L D P C codes. So, you cannot say that they are another bigger class. So, what is a regular L D P C code, so you say sparse for L D P C. Then you have to say constant column weight and constant row weight. So, maybe we call this w c A constant row weight. These are the two things that define regular L D P C codes. So, L D P C codes are codes, which have a sparse parity check matrix regular L D P C codes are those codes, which have a sparse parity check matrix. This also there, every column should have a same weight. So, in the same number of once every row should also have the some other same weight. So, same number of once `w c and w r. So, you can now define what is known as a n, w c, w r regular ensemble of L D P C codes.

(Refer Slide Time: 44:28)



(n, w_c, w_r) - regular ensemble of LDPC codes:

PC matrices : # of columns $= n$

of 1s in each column $= \underline{\underline{w_c}}$

" " " " row $= \underline{\underline{w_r}}$

# of rows $= \dfrac{n\,w_c}{w_r}$ (this has to be an integer)

Usually, fix w_c & w_r as constants

$$\text{Rate} = \frac{n - \text{rank}\left(\overset{\text{PC}}{\text{matrix}}\right)}{n} \geq \frac{n - \frac{n w_c}{w_r}}{n}$$

$$= 1 - \frac{w_c}{w_r} \triangleq \text{Design rate}$$

I can call it set, but ensemble sounds a little bit more grand expression. So, it good to use these terms. So, what are these codes number of columns equals? So, these are basically parity check matrices. That have the following properties number of columns equals n number of ones in each column equals w c number of ones in each row equals w r. What about number of rows? Should I specify that it turns out that not independently specified.

So, once you have n w c w r the number of rows becomes what does the number of rows become. How do you do it? You have to compute the number of ones in two ways, compute them column wise and compute them row wise you have to, of course get the same number. So, you equate that you will get that the number of rows becomes n times w c by w r. This has to be of course, an integer.

So, thus that will impose some restriction on n for a given w c and w r. So, is that clear how did the number of rows become n time w c divided by w r. Think of the total number of ones in the entire made tricks. It is definitely equal to n times w c and it is also equal to number of rows times w r. So, that equation gives you number of rows n times w c by w r and this has to be an integer. Of course, everything has to work out this has to be an integer. So, w r has to divide n times w C. So, that has to be a condition that is satisfied. Otherwise, you cannot have an n w c w r regular on sum of. So, the next question is the interesting is how many such matrices are there for a given n w c and w r how many such matrices?

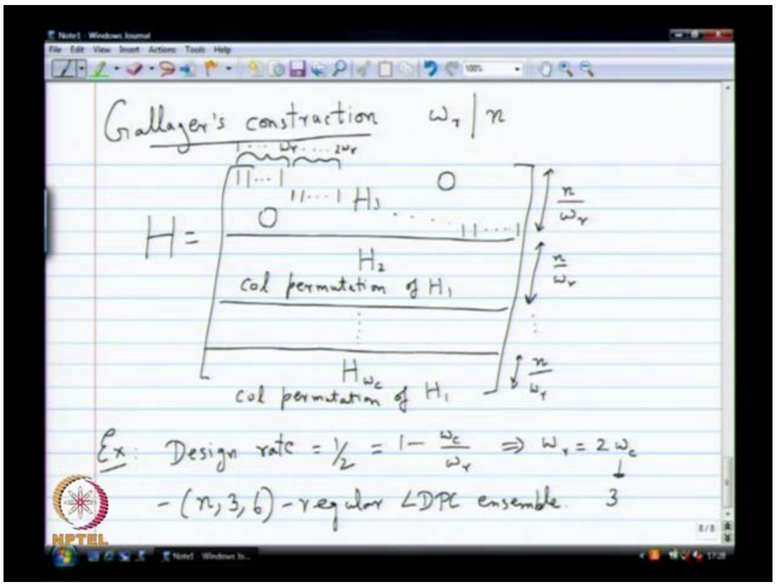
That is the answer I wanted, I did not wanted every precise answer a quiet of you. This is not one definitely not one there are there are lots of matrices. In fact as you increase n if you keep if you keep w c and w r fixed and increase n. So, that is what you might want to do see remember I want a sparse made tricks. So, usually what people do when they look at this parameters is usually you fix w c and w r as some constant. Then you look at this ensembles for increasing values of n remember. When you go to a larger and larger n i expect a better and better codes. So, larger n is better codes, so I want to increase n. So, as you increase n the number of codes number of such matrices is going to be too many, there are going to be lots of codes.

So, that is the first point then why do we consider all of them to be in one ensemble? Of course, the rest to be something that follows you make sense it turns out there is a decoder, when you run the decoder on any one code in the ensemble. You are very likely to get there is similar answers as when you run the same decoder on any other code in the ensembles. So, all these codes in the ensemble gives similar performance when decode according to some specific decoder. That is the reason why you wanted to think all of them together. So, in fact this w c and w r controls the performance of that decoder, which I have not yet described. So, eventually we will see that is the motivation for thinking of all these codes together as one, so before that what about the rate of this code?

Rate is what it is actually n minus rank of the matrix of the parity check matrix am I right? Divided by n this is the actual thing, but what do we know about rank of parity check matrix? You can really can not exactly calculate it depending on do it, but I can definitely lower bound it. The rank will definitely be less than or equal to the number of rows. So, this n minus rank will be greater than or equal to n minus the number of rows which is n w c by w r divided by n. So, this becomes 1 minus w c by w r. So, this quantity is defined to be what is known as the designed rate, it’s called the designed rate.

So, given the w c and w r and know my rate is one minus w c by w r, at least one thing we know. So, all the matrices in this ensemble have a common lower bound for the rate, rate is at least this for all the codes in the ensemble. At least that is one good starting point much more definite, but the decoder like I said behaves very similarly. So, that is that is main reason for this definition. So, the next interesting thing is construction. Suppose, I give you w c and w r and give you an n, how can you go about constructing? Let’s say one parity check matrix from this big ensemble, how will you construct? So, Gallagher gave a solution for that is call Gallagher construction.

(Refer Slide Time: 51:02)



Ex: Design rate $= 1/2 = 1 - \frac{w_c}{w_r} \Rightarrow w_r = 2w_c$

$-(n, 3, 6)$-regular LDPC ensemble $\quad 3$

It works when w r divides n it works when w r itself divides n. So, remember we wanted w r to divide n times w c. So, of course if w r divides n it also divides n times to b C. So, Gallagher construction works when w r divides n. So, the construction is very simple it works like this. So, you construct overall parity check H using w c sub matrices. So, you will have a H 1 and you have a H 2.

All the way down to the last one, which is H w c. So, the H 1 part of it will be very simple. So, what you will do is each guy the number of rows. I have to tell you basically be the number of rows in each one will be n by w r so on to the last one, which is also n by w r. Of course, this plot this drawing is not the scale. So, you have to imagine all this arrows equal lengths. So, n by w r and then I am going to put in the first row of H 1 I am going to put w c ones did they get that right w r ones. I am going to w r ones, what will I do in six second row? Just start from here w r plus 1 to 2 w r. So, disguise 1 to w r disguise all the way to 2 w r. This way if you keep on coming what will be the last thing here? It will go all the way the remaining things you put as 0.

So, is that did everything work out correctly? Somebody is mentally checking I think n by w r w r ones, I will get the correct answer. I think it's okay this works out. So, for H 2 what you should do is you take some permutation of column permutation of H 1 you do the same thing for H 3 same thing for all the way down to H. So, this was Gallagher's construction and clearly you can see it's not a unique parity check matrix. How many possible construction are there as so for H 2 you have n factorial possibilities. H 3 you have another factorial possibilities. So, clearly it's going to be n factorial rays to the power w c minus 1. So, it has a lot of matrices and you cannot even be sure, which of them will actually generate distinct codes.

So, right you would be careful you cannot just mix up the permutations, you might get the same code. So, what a lot of codes some of them will be distinct, some of them will be the same extract. So, this is Gallagher construction and it's clearly an ensemble of several codes, not just one code. Even within the Gallagher construction it's a big set for every permutation you pick, you get different code. That's it that is the construction nothing more do not need to no any image of finite fields and polynomials.

Everything you can just construct you can almost Mattel lab program for this construction. Mattel lab has permutations Mattel lab has matrices sparse matrices. You just write it down you get a quick construction for Gallagher's. Gallagher's construction, so typical choices what people do like I said is to fix w c and w r for a particular rate, then look at larger and larger n.

So, if you want to look at some simple examples. You might want to say have a designed rate of say 1 by 2 means it's a kind of a economical rate one half. So, I know I want this to be equal to be 1 minus w c by w r. So, this clearly imposes the constrain that w r is two times w c. So, once you have w c and w r are related by that rate and a very typical choice for w c is simply take 2 or 3 something very lower number. So, a good choice for w c is three good choice. So, you start with 3, so 3 means you have w r equals 6. So, you can think of for designed rate half n 3 6 regular L D P C on this. So, you know n has to be an even number otherwise this will not work out.

So, of course that a enough even numbers out there. So, you can just think of n, n is not a big problem. So, n 3 6, so another choice would be to take w c equals 4. If you take w c is four, then w r becomes 8. Then you have the n coma 4 coma 8 regular and for any value of n you can think of larger and larger constructions that will work out. Of course, the important question is out of all these random constructions is there any one that is preferred? Is there any one that you would prefer for whatever reason? So, one reason might be you want a good minimum distance. Another reason might be you want a very simple implementation of the decoder.

So, there are so many other reason that you bring in to choose a specific instance of this construction. You might want to do that and that is what people do when they propose codes for their standards. They will propose a certain code and say this performs really well. So, also have a good minimum distance it has a simple implementation at the decoder. The main criteria is going to be of course, coding gain what is the coding gain corresponding to a particular construction. So, it turns out its big difficult to quantify. This things exactly at least for a finite n, if you fix everything and say I give you a certain construction a particular permutation for H 2 for particular permutation for H w c. You tell me what the coding gain is for your favorite decoder? Like I said the best thing to do is to simulate.

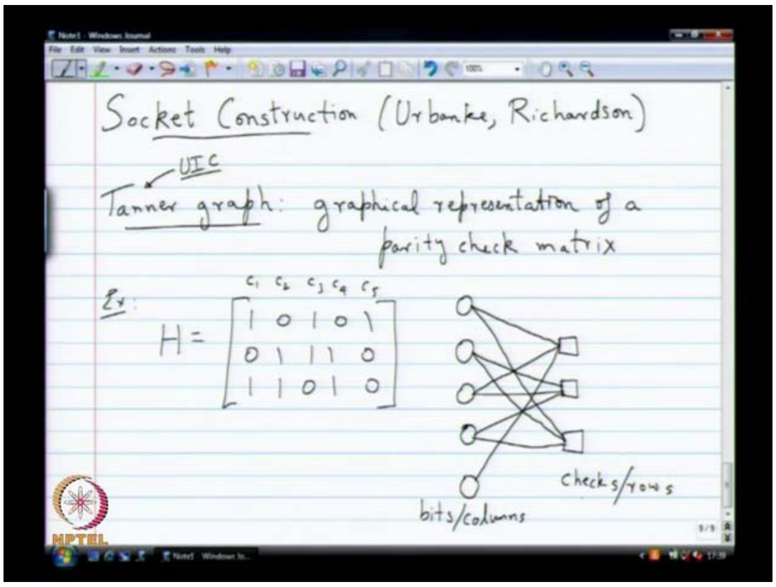
Try now of this simulations is going to take a long time. You want to simulate it 10 power minus 6, which means you have to simulate 10 power eight blocks. If you want to try each and every one of these codes two many of them. You will not finish before the earth dies or something like that there are no any point in doing communication. So, what you really want is some guide line. What is the what is the good guide line to look for what are the good choices for w c and w r? Should I pick 3, 6 or 4,8 which is better for design rate half? That is the question that you might want to answer given the all these complexities. Even that you are not able to answer all these things very precisely we will settle for any guide line or any approximate analysis, which leads us to a good choice.

Of course, it has to backed by a simulations, but it need not be very regress. So, many of the analysis I will do for L D P C codes. At least we will see it’s not very regress. It will be some kind of an approximate analysis, it should be will be may be backed up by simulations and that is good enough. In most cases people will do also not just approximate, but also asymptotic, what is asymptotic now as becomes a infinite.

So, very large, so we will do what is known as approximate asymptotic analysis for L D P C codes. That will gives us very nice designed choices, which you can definitely use in finite time without worrying about, which is also complicated. Actually if you want to program all those choices in it's quite complicated by itself, but at least it will run in our standard computers.

So, it will work very fast you can do it. This is of course, like a very different from red solemn codes and this came very nicely. There is no doubt about it. So, of course everything hinges on the decoder and before we go to the decoder let us see one more construction. You might say this w c and w r works only for some n what of about all n does not work w r dividing by n dividing n may not be may not hold for you. So, how do you do that? Just one more construction, which is called as a socket construction, which you can say due to Urbanke and Richardson is much more recent.

(Refer Slide Time: 01:01:17)



Socket Construction (Urbanke, Richardson)

UIC

Tanner graph: graphical representation of a parity check matrix

Ex:

$$H = \begin{array}{c} \begin{array}{ccccc} c_1 & c_2 & c_3 & c_4 & c_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

This I will describe using a representation of the parity check matrix, which is called as a Tanner graph. So, this Tanner graph is quite important, I am introducing it in the context of the socket construction, but it is interesting in its own regard for various research. So, it is basically a graphically representation of a of a parity check matrix. So, when I say graze it's not your y axis graph log scale or something. It's basically, but it is a notes graph theoretic graph not any of your other graphs. So, you will have vertices and edges connecting them, so that is the graph I am talking about.

So, best way to define a Tanner graph for example it is a very easy thing to define. It has a very nice very nice generalization very easy generalization. So, I do not have to define it in a general case. So, let me define it by example I will take a simple parity check matrix and show you the Tanner graph corresponding to them. Let's say we take 1 0 1 0 1 0 1 1 1 0 1 1 0 1 0 so this is my parity check matrix it has regular row weight, but not really a regular column weight, it does not matter.

So, Tanner graph is basically a bi parted graph a bi parted graph will basically have one side of column on the left side. One side of vertices on the left side another set of vertices on the right side. All edges will connect from left to right only. So, left all vertices on the left side will not have any edges between it only be from left to right. So, this vertices on the left will correspond to the columns of this parity check matrix or the bits of the code. Remember, the columns of the parity check matrix, also correspond to the bits of the code.

So, I can think of you remember those always this the in which we wrote down the codes words of the code on top of the column, here it makes some sense. So, here we have five columns, so I will have left vertices. We have three rows we will have three right vertices the right vertices will correspond to the rows left vertices correspond to the columns. So, of course the right vertices right vertices have to corresponds to the rows and the right vertices. Just to distinguish them I will draw them as squares. So, these are the left vertices they correspond to bits or the columns the right vertices corresponds to the parity checks right each row.

Of the parity check matrix is actually a parity check and they correspond to the checks of the rows. So, the left nodes are also called bit nodes the right nodes also called as check nodes. So, you can call them column nodes roman nodes if you like it. If I also call variable checks nodes so many names as many names is there. There are researches in this area, now connections let us see, which let's see if this is a concept without even telling you what the connections are, if you now you do not answer to this question.

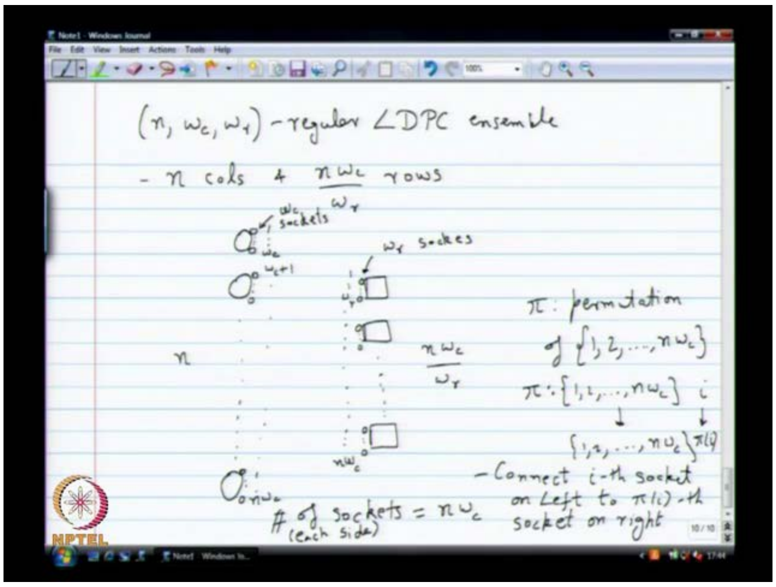
So, if you do not know if you never heard this word Tanner graph before, how can I connect vertices on the left with the vertices on the right? So, basically it will depend on the locations of the once. So, wherever there is a one I should have some edge connecting to that corresponding column and row it seems, like way of doing it.

That is the definition of the column of the Tanner graph. So, if you want to draw that you can look at the first column it’s connected to the first row and the third row. So, you can draw a line like this you can also do it the opposite way. So, if I look at the first check of the first row it’s connected to first row its connected to the first column third column and the fifth column. If I look at the second check its connected to the second column third column and the fourth column. If I look at the third check it’s connected to the first column second column and the fourth column that is it, that is your Tanner graph. So, clearly there is a one to one correspondence between the Tanner graph and the parity check matrix given a parity check matrix

Even go to the Tanner graph given you can go to the unique parity check matrix. So, no doubt about it, by the way the Tanner is stands for stands for what it’s a name of a person, a thing, anything else is this still an is this in a U I C. So, this graphs are of course, invented by him. So, its name is called Tanner graph you are saying it’s not sparse. So, Tanner graph is not necessarily for sparse Para digit for any parity check is subset with the density low density, all along I suddenly I give you high density. So, this definition holds for any Para digit meter.

Of course, we will work with low density it can be, is that clear? This is a very simple thing. Like I said generalization is obvious, if you have any parity check matrix Tanner graph can be done very similarly, there is no problem. How does the socket construction works?

(Refer Slide Time: 01:08:19)


(n, w_c, w_r) - regular LDPC ensemble

- n cols + $\frac{n w_c}{w_r}$ rows

π: permutation of $\{1, 2, \ldots, n w_c\}$

$\pi : \{1, 2, \ldots, n w_c\} \to \{1, 2, \ldots, n w_c\}$, $i \to \pi(i)$

- Connect i-th socket on Left to $\pi(i)$-th socket on right

# of sockets (each side) $= n w_c$

So, remember in a n coma w c w r regular L D P C code ensemble you have n columns and n w c by w r rows. So, any parity check matrix in this in this ensemble will have a corresponding Tanner graph. It will have n bit nodes and w c by w r check nodes, so let us draw them. So, by the way the socket construction will construct the Tanner graph not a parity check matrix, but of course you know how to go from a Tanner graph to you n f. Then here you have n w c by w r of them. Here you imagine, now that there are w c sockets in each bit node. You know that each bit node is connected to w c check nodes. So, let us imagine that each of those edges go into some socket and you have a explicitly w c sockets. We have w c sockets these are sockets. So, these are basically locations for edges to come out of like ways on each check node you have w r sockets .

So, how many sockets will you have can total on the left hand side? How many sockets will you have on the right hand side? So, both of them will be the same number of sockets on each side. So, it is on both sides n times w c, so you number the sockets on the left hand side starting from the top one to n w c. So, you number them number the sockets one this will be w c this will be w c plus 1. All the way down to n w c likewise here also you number sockets w r all the way down to n times w c is that you number the sockets on both sides.

Then you come up with pi which is arbitrary permutation of the set one two all the way to n times w c. So, phi is some permutation of this, so pi what is what you mean by permutation phi is? Some mapping from someone to one mapping from one to n w c to the same set. So, it is a one to one mapping, so permutation. So, this will basically take an i to pi or phi.

So given any I will go to any pi or phi. So, for any permutation I will take the i socket on the left hand side and connect it to the pi of i th socket on the right hand side. Connect on left to pi of i th socket on right. So, every permutation will correspond to a Tanner graph, but there is one catch here, which is slightly, which I have I have not explicitly pointed it out. Let me see if anybody notices that what can happen you can have multiple edges.

So, that is the only prop in this socket construction it seems like you can multiple edges between the between a single bit note and exactly another check note. So, you some you need some way of handling that. So, what you can do for instance is what simply throw away multiple edges. If there are multiple edges you just reduce the edges from the then what will happen? Then you will weight will fall you would not get a regular matrix. So, the socket construction produces regular Tanner graphs and not necessarily regular parity check matrices.

The only problem is because of the multiple edges, but of course it will also produce all the all the regular matrices, will also be produced. There will be some permutation, which will only give you the there will be lots of permutations, which will not give you any multiple edges

All those case will give a proper valid parity check matrix there is no problem in that, but it will also produce some other things with multiple edges. If you want you can throw them or you could have some other way of converting it into some parity check matrix, but then in the socket construction you have to think of the unsolvable. In terms of the Tanner graph and not in terms of the parity check matrix. We run out of time I think I do not know when we ran out of time, but any way its end of this class. So, we will stop here and pick up from here next week.