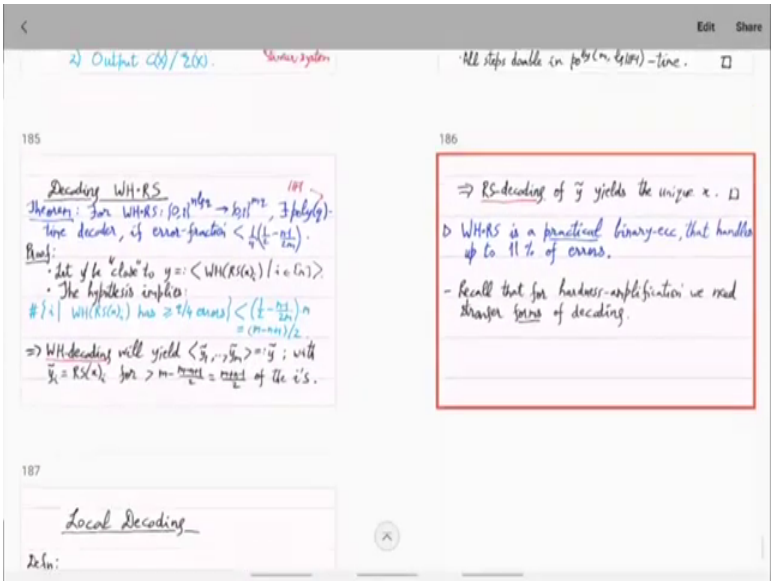
Randomized Methods in Complexity
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture-24
Local Decoding

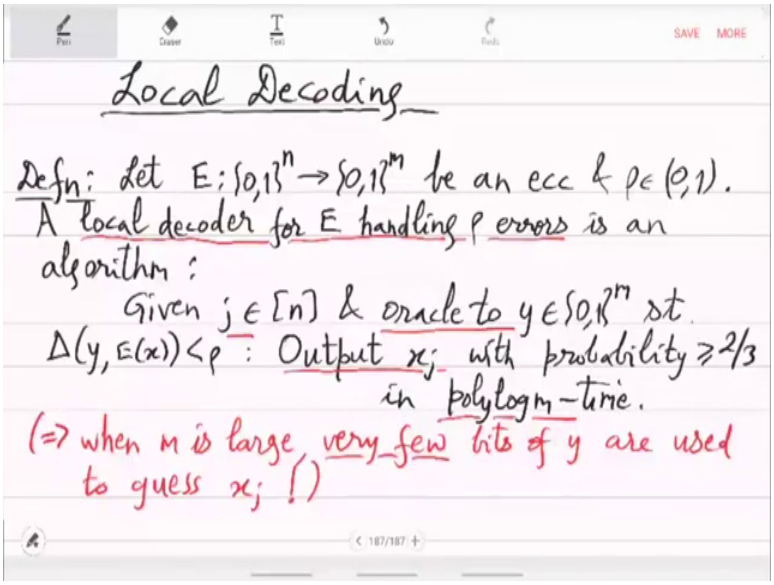
(Refer Slide Time: 00:20)



In the last class we did the decoding, unique decoding of Reed-Solomon and then based on that Walsh-Hadamard Reed-Solomon concatenated code. So, for Reed-Solomon we showed that the errors that it will be able to handle is 22% in practice it is a 22% of errors can be uniquely decoded in deterministic polynomial time and Walsh-Hadamard Reed-Solomon is that will be 1/4th of 45% which is more than 11%.

So, Walsh-Hadamard Reed-Solomon is a practical binary code and there is a deterministic polynomial time algorithm to correct up to 11% of errors, which is pretty good but for hardness amplification of worst case hard for functions to average case hardness we need stronger form of decoding basically we will want to limit the number of queries we make to the corrupted string the number of places we are querying and that is called local decoding. So, let us define it.

(Refer Slide Time: 01:34)



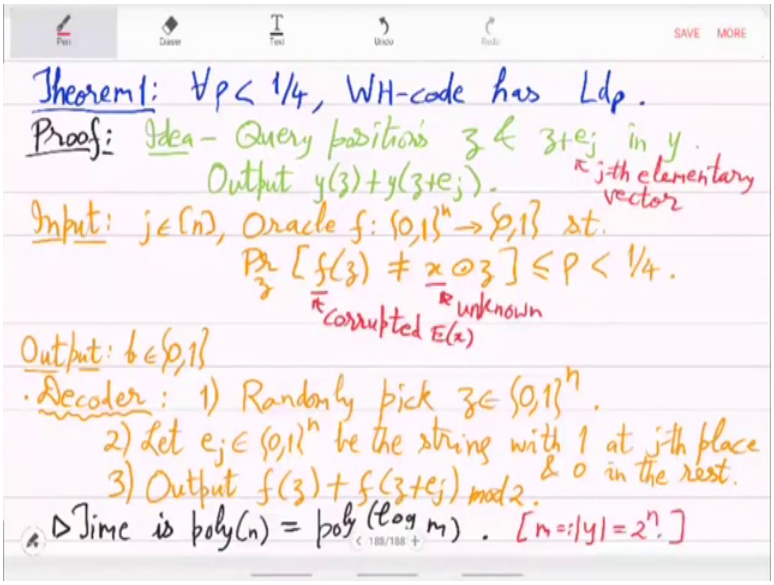
Local decoding :

Defn:let $E: \{0,1\}^n \rightarrow \{0,1\}^m$ *be an ecc* & $\rho \in (0,1)$. So, this will be the parameter associated with local decoding basically, how many errors can it tolerate? So, a local decoder for E handling ρ errors is an algorithm *Given* $j \in [n]$ & *oracle to* $y \in \{0,1\}^m$ $s.t\ \Delta(y, E(x)) < \rho$

So, why is rho close to the code word E (x) that is what is given and of course, you have to find this x_j bit of x of this code word E(x) you do not have to find the whole string x not n bits but only 1 bit. Output x_j with probability$\geq 2/3$ in poly logm-time so, the j should be output and the time taken should only be polynomial log m. So, remember this is exponentially smaller than the length of phi which means your algorithm can query at most poly log.

Many locations it cannot look at the whole string l case you could compare this with the log space computation. Except we are allowing you polynomial in log m. So, for example, $(\log m)^2$,$(\log m)^3$ is fine. So, in other words when m is large, very few bits of y are used to guess x_j So, this is consistent with the term local, basically just by using very, very local information about y you are deducing x_j with high probability, remember that x_j has only 2 choices 0 or 1. So, your answer back should be correct with 2 thirds probability at least and now, surprisingly, will show that all these codes that we defined they have local decoders.

(Refer Slide Time: 06:08)



So, let us start with **Walsh-Hadamard**; So, Walsh-Hadamard code has a local decoder for any $\rho < 1/4$, $WH\ code\ has\ Ld_{\rho}$. So, if the number of errors is less than 25%, say 24%. Then any bit can we locally decoded let me define short term for this Ld_{ρ}. This is especially helpful in Walsh-Hadamard code because that is a very long code n bits stretching 2^n and now here local decoder will only need polynomial in n many queries only those many bits it will not require the whole 2^m.

When many bits, so this is especially good to know what is the idea. So, the idea is quite simple. If you want to know x_j bit, jth bit of x out of the n bits, then basically you look at the string via this corrupted code word, randomly pick a location and pick a location that is correlated with j. It is basically just shifted by a vector which is the elementary vector jth elementary vector.

So, you look at you pick a position z and pick up pick another position $z + e_j$ and take the sum of those 2 take the parity that is the idea. So, query 2 positions z & $z + e_j$, where e j is jth elementary vector which is simply flipping the jth bit in z, so it is z and z with the jth bit flipped these 2 locations in y and then you take the parity of those 2 locations. So, output $y(z)+y(z + e_j)$ location that is the idea. So let us see the algorithm. Input:So $j \in [n]$,

Oracle $f : \{0, 1\}^n \rightarrow \{0, 1\}$. So, f is just an Oracle, given an n bit string, it will output a bit in that

location of y. So, $Pr_z[f(z) \neq x \odot z]$

This probability is given to be smaller than ρ $where\ \rho < 1/4$ and in the output what you want is, so, this x is unknown first of all and f is corrupted E (x), so, f is given to you as an Oracle that is a corrupted version of E (x) the code word x is the unknown for which you want the jth bit. That is your input and output will be just a bit : $b \epsilon \{0, 1\}$. That bit has to be with high probability exceeds.

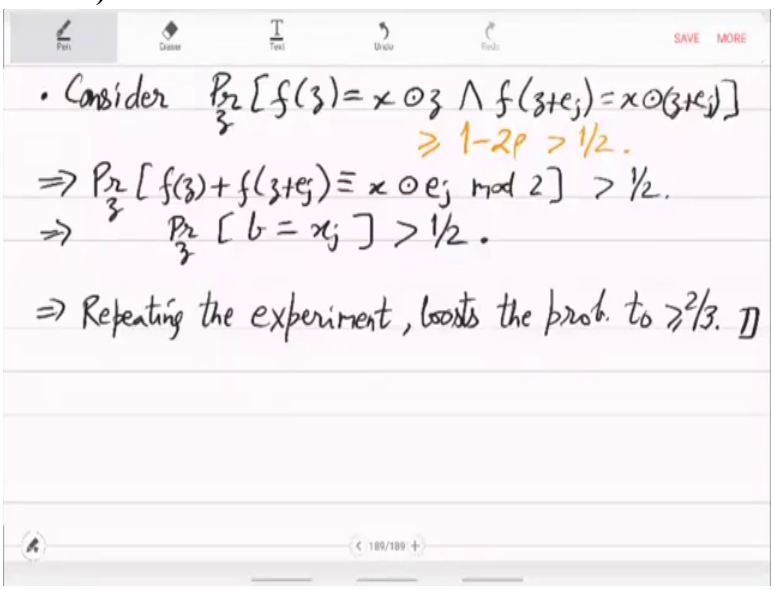
Decoder : 1)$Randomly\ pick\ z \epsilon \{0, 1\}^n$

2)$Let\ e_j \epsilon \{0, 1\}^n\ be\ the\ string\ with\ 1\ at\ the\ j - th\ place\ \&\ 0\ in\ the\ rest$

3)$Output\ f(z) + f(z + e_j)\ mod 2$

The algorithm is exactly what the idea was and now we have to analyze why it works, why would it give x_j. So Time is poly (n)=poly (logm) [$m = 2^n$]

(Refer Slide Time: 13:46)



Now, let us do the analysis.

$Pr_z[f(z) = x \odot z \wedge f(z+ e_j) = x \odot (z + e_j) \geq 1 - 2\rho > 1/2$

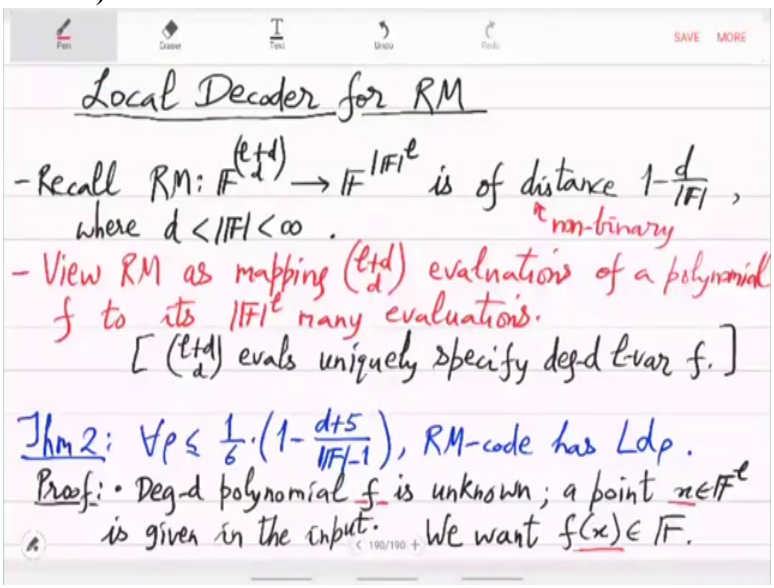
$\Rightarrow Pr_z[f(z) + f(z+ e_j) \equiv x \odot e_j mod 2] > 1/2$

$\Rightarrow Pr_z[b = x_j] > 1/2$

but you wanted 2 thirds, so you have to repeat this experiment repeated, let us say 3 times and then take the majority vote.

So, repeating the experiment boosts the probability $\geq 2/3$, so then the repeat the experiment, take the majority vote and then output the majority as b that with very good probability is x_j. So, that finishes this theorem, which is the local decoder for Walsh-Hadamard for errors up to 25% less than just less than 25%. Now, we have to do a similar thing for Reed-Solomon, instead we will do it for Reed-Mueller, it will imply for Reed-Solomon. So next do that.

(Refer Slide Time: 17:16)



Loca decoder for RM :

Recall RM : $F^{\frac{l+d}{d}} \to F^{|F|^l}$ *is of distance* $1-\frac{d}{|F|}$ *where* $d < |F| < \infty$

So, that the distance is at least half if you take it bigger than the distance will be close to 1 but remember this is non-binary distance. It is not unlike Walsh-Hadamard, this is non-binary. So, to make it binary, remember, later we will apply Walsh-Hadamard that we have to give a different local decoder let us first do Reed-Mueller. So, let us take this view that instead of l variate d degree polynomial coefficients being the domain of RM we will take that many evaluations.

So, view RM as mapping$\frac{l+d}{d}$ evaluations of a polynomial f to its$|F|^l$ evaluations and note that

$\frac{l+d}{d}$ evaluations uniquely specifies f because it is degree d and l variate. So, $\frac{l+d}{d}$ evaluations will uniquely give you all the $\frac{l+d}{d}$ coefficients hence uniquely specifying f. So, we could have used either of the representations, coefficients or evaluations, let us look at the evaluations for local decoding is equivalent and with that understanding the theorem

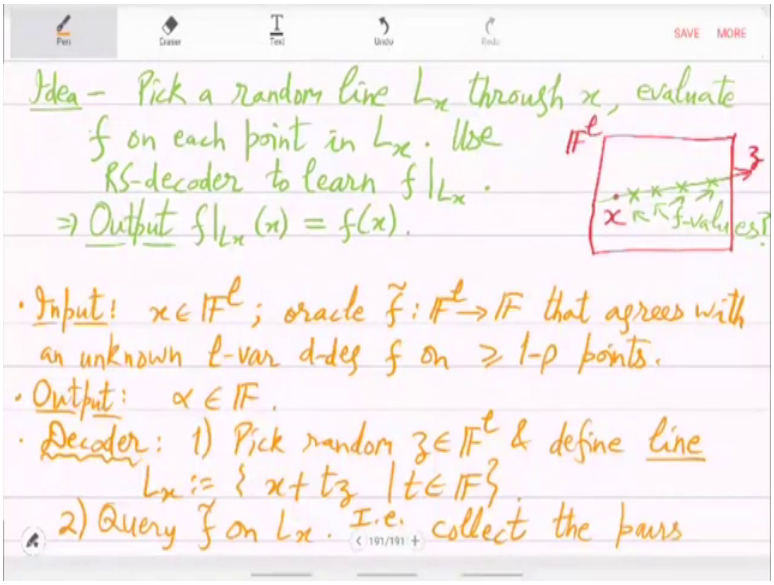
Thm2: $\forall \rho \leq 1/6. (1 - \frac{d+5}{|F|-1}, RM\ code\ has\ Ld_{\rho}$

. What is the idea, the idea will be actually a generalization of what we did for Walsh-Hadamard. So, there we picked a random location and a correlated location in the Reed-Mueller case, what we will do is we will again pick a random point in space.

Because this polynomial F is given to you by evaluations. So, we will pick a random location, which is actually a random point in the space F^l and we will just pick a random line through that point. So, on the line, we will query f that is the idea. So that all these points on which we are querying, they are somehow correlated. So, the point that will take in space will be this point, which is given to you in the input, you wanted to evaluate f at point x.

Take that point x, that is a fixed-point x in the space and then pick a random line. So, in a way, opposite to what you did for Walsh-Hadamard. But in the end, all these points on which you are querying there, they are all heavily correlated by x. So let us first give the setting so Degree -d polynomial, f is unknown and a point x in the space,$x \epsilon F^l$ is given in the input and what you want is we want $f(x) \epsilon F$ and then you want to evaluate f at x. It is this question, so we will draw a random line through x in the space, that is the idea.

(Refer Slide Time: 24:11)



Idea pick a random line L_x through x ,evaluate or query f on all these points. So, what we are doing is, this is your space F^l. This is the point x in the space given to you and what you have done is drawn a line. That is a random line and this is where you will query f this is what you query f value is what are the f values at these points and how many points there are on this line, well in the real space, there would have been infinitely many points.

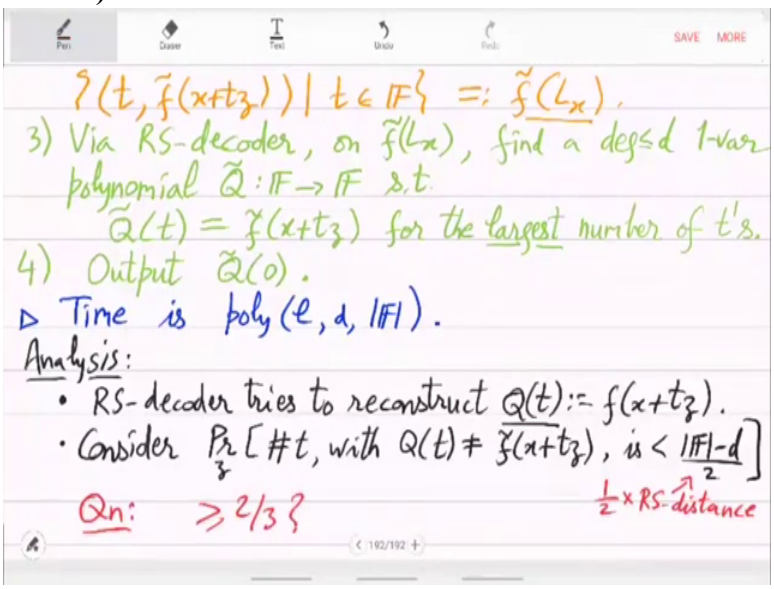
But fortunately, we are in this finite field based space. So, the number of points will only be size of the field which is finite. So, you query small f on all these points and from that try to learn f x that is the goal. So, pick a random line L_x through x evaluate f on each point in L_x. Now, remember line is like evaluate or querying f on L_x like reducing f to univariate case. So, f now becomes a univariate polynomial and that means you can invoke Reed-Solomon decoder.

So, use Reed-Solomon decoder to learn if that is the idea and once you have learned f correctly on L_x you can evaluate f (x). So, $f \mid L_x = f(x)$. So that is what the output the idea is simple. Let us now do the actual calculations and see what is the success probability and why does it work? So let us go through it formally. So, the input is you are given an $x \epsilon F^l$; $oracle\ \overline{f}: F^l \to F$
Basically, you are given these this Oracle that will allow you to evaluate f at any point you like in this space. So, Oracle f tilde that agrees with an unknown l variate d-degree f on $\geq 1 - \rho\ points$ a lot of points good fraction, so on greater than $1 - \rho$ and what do you want in the output you want f (x), so you want to output of field element, which would be with high

probability f (x) . So let us achieve that.

$Pick\ random\ z \epsilon F^l\ and\ define\ line\ L_x := \{x + t_z | t \epsilon F\}$, you can think of this as direction is direction being z. Second step is query $\overline{f}$ which is a corrupted version of f query this on each of these points on L_x that is collect the pairs.

(Refer Slide Time: 30:30)



$\{(t, \overline{f}(x + t_z))|t \epsilon F\} =: \overline{f}(L_x)$. So, on the random line, but passing through x, you have computed $\overline{f}$. Now, how many of these pairs are correct? That is the match with f we later analyze that actually, it is a good fraction and so, it makes sense to do Reed-Solomon decoding. So, via Reed-Solomon decoder, this is the actual unique decoder, it is not we are not doing local decoding.

So, use the Reed-Solomon unique decoder on $\overline{f}(L_x)$ and it will give you find a degree$\leq$ d polynomial $\overline{Q}: F \to F$. So, it will give you a univariate Q tilde. So, basically, this has reduced the L variate question to 1 variate that was the whole idea and what is the property of Q tilde such that $\overline{Q}(t) = \overline{f}(x + t_z$ for the largest number of t's. That is the property satisfied by $\overline{Q}$ and finally, output so, if $\overline{Q}$ is the correct f on the line.

Then as you saw before t = 0 is what we are interested in you just output $\overline{Q}(0)$ good that hopefully will be the answer. So, let us just quickly observe that all this can be done in polynomial time. So, time is poly$(l, d, |F|)$ because the Reed-Solomon decoder will work and also the line computation will already take if and everything else it is polynomial time. So, next we will do the analysis. So, it is clearly an efficient algorithm. But why would work?

And we have to more importantly see that f(x)value is outputted with a probability of at least 2/3 rd. So, let us do that analysis. RS decoder is trying to reconstruct the polynomial univariate polynomial $Q(t) := f(x + t_z)$. So, basically, the $\overline{Q}$ is supposed to be Q (t) and we have to now check what is the chance that this actually happens that $\overline{Q}$ is equal to q? Because remember there is also corruption in $\overline{f}$, it is not really if and if you pick an unfortunate line L_x then you may get completely wrong answers $\overline{Q}$ may not be Q.

So, it is clearly an efficient algorithm. But, why would work and we have to more importantly see that f of x value is outputted with a probability of at least 2 thirds. So, let us do that analysis. So, RS decoder is trying to reconstruct $Q(t) := f(x + t_z)$but, so, basically the $\overline{Q}$ is supposed to be Q (t) and we have to now check what is the chance that this actually happens that $\overline{Q}$ is equal to Q, because remember, there is also corruption in f tilde.

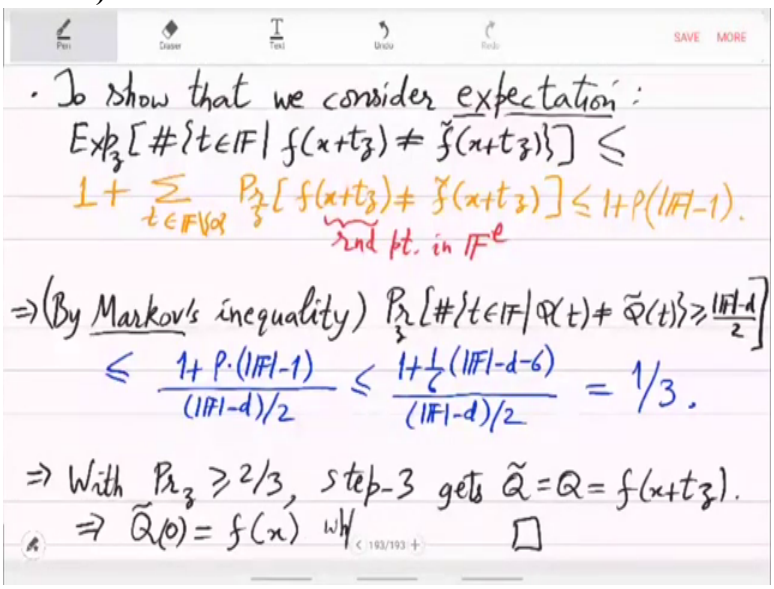
It is not really f and if you pick an unfortunate line L_x then you may get completely wrong answers $\overline{Q}$ may not be Q. So, we have to analyze that. So, let us consider $Pr_z[\#t, with\ Q(t) \neq \overline{f}(x + t_z)$ once you have picked a direction z you want Q (t) to be correct on many many t's.

So, in particular the places where Q (t) the t's where Q(t) is incorrect, you want that to be smaller than what Reed-Solomon RS decoder can sustain. So,

$Pr_z[\#t, with\ Q(t) \neq \overline{f}(x + t_z)$is $< \frac{|F|-d}{2}$ this will number of t should be smaller than $1/2 \times$ RS distance. That is what this is.So, once this happens once you have that this number of t's is smaller than half of Reed-Solomon distance.

Then Reed-Solomon decoder automatically as you saw before, when we analyze RS decoder, it will give you the correct Q. So, all you want to ensure is that this probability is high. It is at least 2 /3rd. That is the question is this probability 2 thirds and we will show yes.

(Refer Slide Time: 38:56)



To show that we consider expectation:

$EXP_z[\#\{t\epsilon F | f(x + t_z) \neq \bar{f}(x + t_z)\}] \leq 1 + \sum_{t\epsilon F\setminus\{0\}} [Pr_z f(x + t_z) \neq \bar{f}(x + t_z)]$. f is a polynomial; I mean x and z are both fixed. So both of them are univariate polynomials in fact. So, one variate they are both one variate. So they being different means that t is not 0. But how do I get this information. So here, actually, what you have to see is that $x + t_z$ is a random point in the space.

Because z, you picked a random direction. Secondly, the third. So that is what is important. This is a random point, random point in F^l and once you realize that f and $\bar{f}$ are being different on a random point in space, that is given to you by rho. That is the probability upper bound. So, $1 + \sum_{t\epsilon F\setminus\{0\}} [Pr_z f(x + t_z) \neq \bar{f}(x + t_z)] \leq 1 + \rho(|F| - 1)$. We remove t = 0 because that would have killed z.

But for all other t's, z is there in x + t z and so this is really a random point. So, you have ($1 + \rho$)times F as the expectation. Let us take that. So now by Markov's inequality, basically, it is just something simple. So, it is the following that is the expectation is m, then what is the chance that in the random experiment, the random variable takes value more than 2m or less than m / 2, these are both these the probabilities you can suspect already will be smaller than half.

Because once you do expectation, you also have a good understanding of the probabilities that the random variable is too large or too small, away from the expectation. So, in particular, what you get here is $Pr_z[\#\{t \epsilon F | Q(t) \neq \overline{Q}(t)\} \geq \frac{|F|-d}{2}$, this will depend on how big $\frac{|F|-d}{2}$ is compared to the expectation.

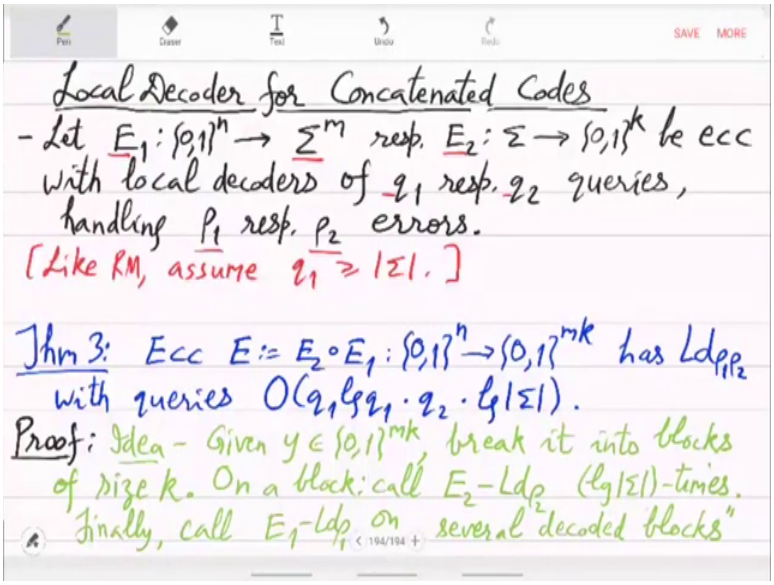
So that is $\leq \frac{1+\rho(|F|-1)}{(|F|-d)/2}$.. That is the idea and let us now just do the simple calculation or simplification. So, rho is given to you by an upper bound, that$\rho < 1/6$ of that distance.

$\frac{1+\rho(|F|-1)}{(|F|-d)/2} \leq \frac{1+\frac{1}{6}(|F|-d-6)}{(|F|-d)/2} = 1/3$ So, that is it. That is what we wanted. So probability over the random direction, when you take a random direction, then the number of t's being bad t's being too many. More than 1/2 of the Reed-Solomon distance, that probability is at most 1 /3rd. So, for 2 /3rd of the choices, the Reed-Solomon decoder will work fine.

So $with\ Pr_z \geq 2/3$ step-3 gets $\overline{Q} = Q = f(x + t_z)$ which further means which obviously means that $\overline{Q}(0) = f(x)$ with probability 2 /3rd so, that is the local decoder handles errors up to 1 sixth of the reed-muller distance and what you should note here is that the number of queries is log of the length. So, this is really $log \frac{l+d}{d}$ or I should say$lof(|F|^l)$.

So, it is poly log in the length of the codeword. So, hence it is fair to say that this is a local decoder it makes very few queries and gives you one evaluation of F with 2 thirds probability at least. So, once you know local decoder Walsh-Hadamard and Reed-Mueller.

(Refer Slide Time: 48:15)



The last thing is **local decoder for the combination the concatenation for concatenated codes**. So, let me sort of talking about Walsh-Hadamard and Reed-Mueller we will just abstract it out and talk about combination of E_1 and E_2. Where we know local decoders for E_1 and E_2 So $E_1: \{0,1\}^n \to \Sigma^m$ *resp* $E_2: \Sigma \to \{0,1\}^k$ be ecc with local decoders of q_1 resp q_2 queries in the local decoder algorithm and they handle say ρ_1 *resp* ρ_2 errors. So, we have abstracted it out of course, when you apply to Reed-Mueller than Σ will be your finite field. E_1 will be Reed-Mueller E_2 will be Walsh-Hadamard and these queries will be very few. But this is the setting. So now you want to show that the concatenation also has a local decoder.

Moreover, we will assume here that $q_1 \geq |\Sigma|$. Now, that is state the theorem statement.

Theorem3: $Ecc\ E := E_2 \circ E_1 : \{0,1\}^n \to \{0,1\}^{mk}$ *has* $Ld\rho_1\rho_2$

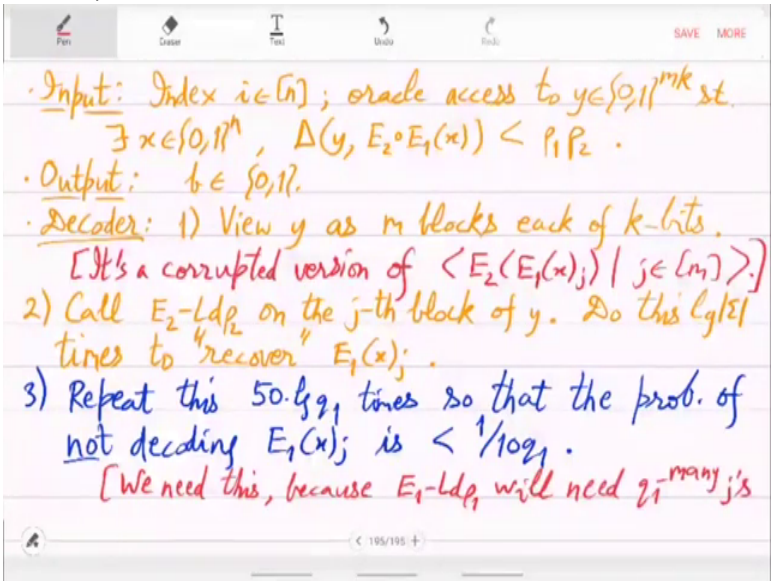
This is similar to what we did when we did unique decoding for the concatenated code, so, you get the product but more importantly the number of queries is small remains small. So, has local decoder this much with queries $O(q_1 log\ q_1 . q_2 . log|\Sigma|)$. So, if q_1, q_2 was small then this overall is quite small this is just like product queries. So, what is the idea given this corrupted y.

Given$y \in \{0,1\}^{mk}$break it into blocks of size k and then think of these m blocks each of size k as

these respective applications of E_2 and use E_2 decoder. So, on a block called E_2 - $Ld\rho_2(log|\Sigma|) - times$ \ So, basically you will get these symbols if you think of Σ as a field element then you will get the full field element and from that you can now move to E_1 decoder to get the bit out of the local decoder.

Finally, call $E_1 - Ld\rho_1$ on several decoded blocks this will first apply E_2's local decoder in a single block many times to get the full symbol and then on these interesting blocks some of the interesting blocks you apply E_1's local decoder and that will give you the information you wanted the bit you wanted of x.

(Refer Slide Time: 55:29)



So, let us give the full algorithm and proof. So, $Index\ i \in [n]; oracle\ access\ to\ y \in \{0,1\}^{mk}\ s.t\ \exists\ x \in \{0,1\}^n,\ \Delta(y, E_2 \circ E_1(x)) < \rho_1\rho_2$. So, **Decoder** : 1) View y as m blocks ech of k -bits..

Note that y is very long we are not actually working with the full y we are just doing this kind of implicitly or abstractly and accordingly we will make the queries but the queries will in the end will be very few actual queries to why locations will be few, so that you can check in the end. So, it is basically [It is a corrupted version of $< E_2(E_1(x)_j) | j \in [m] >$]

So, this is the let us say the jth symbol. So, think of this, the jth block is basically a corrupted version of this E_2 have E_1 (x)'s jth symbol, think of it as a field element and when you apply it to then it will produce a k bitstream. So $Call\ E_2 - Ld\rho_2\ on\ the\ j - th\ block\ of\ y$. So, remember, $E_1(x)_j$ is an element in Σ. So, you need to which is then log|Σ| many bits.

So do this log|Σ|- times to recover $E_1(x)_j$, that is the idea. But anyways, you will get some element and Σ by this. It may be correct. It may be incorrect, that we will see later. So do this on the jth block. But you cannot do this for all j. Otherwise, we will be getting you will be making a lot of queries that that you do not want to do. So, which j we will you use. So, repeat this 50 log q_1 times.

So that the probability of not decoding $E_1(x)_j$. So, the thing is that this $E_1(x)_j$ may be incorrect. Because the local decoder has its own error involved. It is already this 1/3rd error is possible. So, you want to boost it down we want to make the earth smaller. So, let us repeat this experiment $E_1(x)_j\ is\ < 1/10q_1$.

So, log q_1 repetitions would be enough because this will already give you a small enough probability which you multiply with log|Σ| and still you get 1 /q_1 and now, what you do is so, even x_j's are hopefully recovered for the j's we want and now from $E_1(x)$, we want to go to x. So, E_1 is they were m symbols in the output of E_1. So out of those m we have to query q_1 many, those are the j's that you are interested in. So, we need this, because $E_1\ Ld\rho_1$ will need q_1 many j's.

So, the above experiment will run for a lot j's which is q_1 many j's. So, we wanted the probability to be quite small compared to that. Now let us move to E_1's local decoder. So, what will that do? Before that we have to give a guarantee of how many of this j's E_2 work correctly. So, we will do this next time. We will finish local decoder next time.