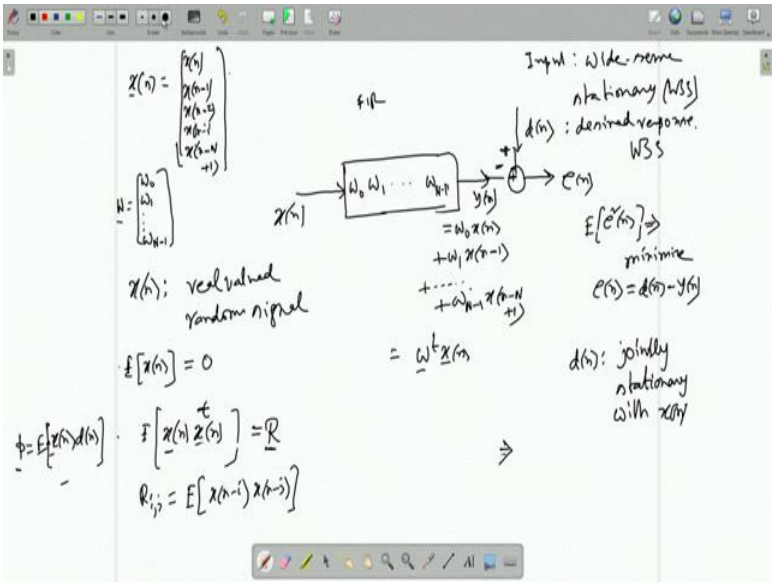
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Lecture 10
Adaptive Filter Basics

(Refer Slide Time: 0:31)



Okay, so just to sum up, XN is a random process 0 mean, (())(00:32) stationary. This mean is 0, correlation matrix is R where X n vector is given by this. W is the filter coefficient vector, this output can be written as, filter output can be a W transpose x n we have already seen okay. D n is another the target response, PR response. There is also WSS at D n and X n they are assumed to be jointly stationary.

There is wherever you take the correlation between D n and any sample of X n. It does not depend on the N, it depends only the gap between the two indices. So we are also given a vector cross correlation vector between scalar D n and the inter vector X n. So the D n times X n expected value, D n times X n minus 1 expected value and all, in all of them N disappears after subtraction because what matters is only the gap between the two indices.

The offence index is N, if it is X of n minus 1, index is N minus 1. If you subtract it becomes 1. Okay, a P is given. Now question is what should be the best filter? What is the, what is the best filter? Best filter is such which will minimize the error power between D n, error power, what is the error E n? E n is a error between the actual D n and the estimate united by the filter output Y n. Since D n is random, Y n is random therefore E n is random.

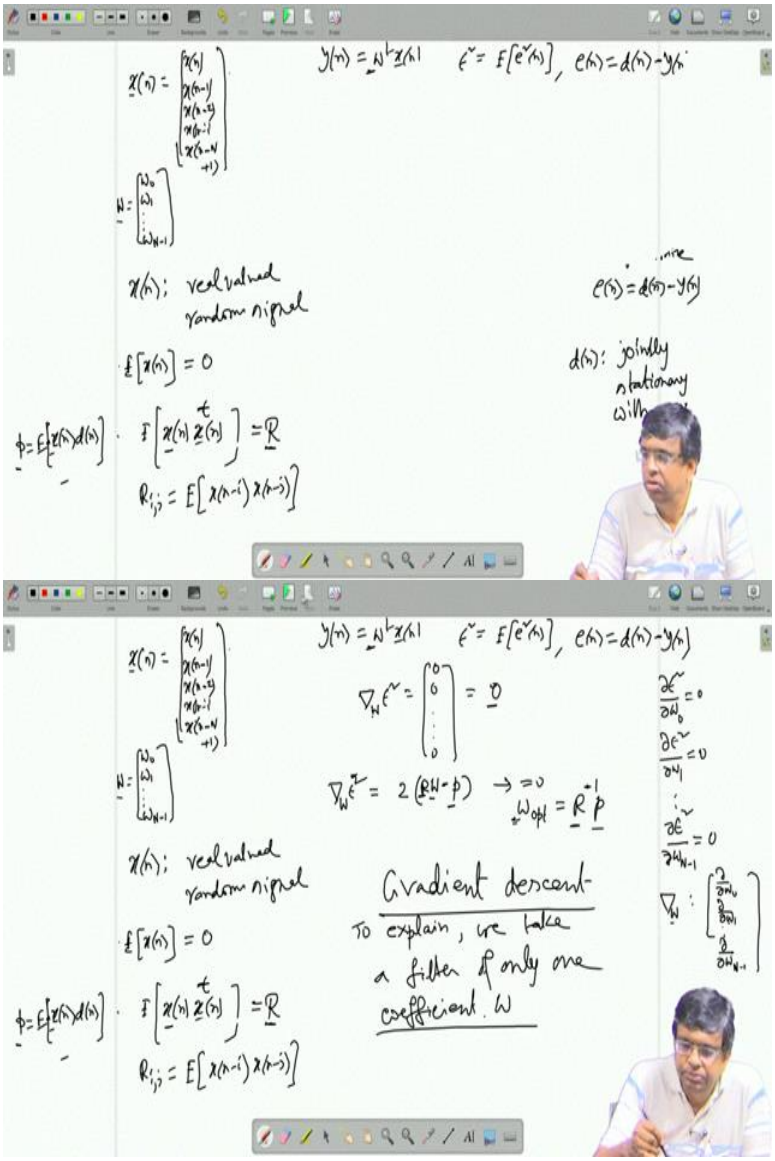
So just, you cannot just take E square n and minimize it, because you have to take a square in the statistical sense, average sense. Because it is random so sometimes in one trial it can show now small value, next time it can show higher value. So that is why you take the average power and then try to minimize it, average power now E n is D n minus Y n. Y n as I told you, is a sum of all this. So if you now replace Y n by this whole expression here, did not square up and then E over it.

It will be D square n term D n into X n term, D n into X n minus 1. All those will be independent up N because of stationarity like D n into X n or D n into X n minus 1.Those are all various elements of P which is given to you, those are known and they are independent of N. Similarly X square E n, X square n minus 1 or X n into X n minus 1 or X n minus 1 ionto X n minus 2. All cross correlation of X n.

They are also known, they are all elements of R metrics and independent of N okay. So in that thing will be basically quadratic function of unknown, which unknown? W 0, W 1 up to this because I want to find out which ones are the best, there is with the unknown. So it will be a second order function of W 0, W 1, up to these. So I will take a partial derivative with respect to W 0 equate to 0.

With respect to W1 equate to 0 dot-dot-dot with respect to W capital N minus 1 equate to 0. And when I take partial derivatives the all second orders things become first order. So, I will form each derivation I will get a first order equation, the state of first order equation fall, you get the best possible W and that gives you the best filter, optimal filter, wiener filter okay to get space I am erasing this part.

(Refer Slide Time: 3:42)



I am just rewriting the basic equation filter output was YN, which was W transpose X n okay. And what we are minimizing? We are minimizing expected value of the square of error, so error x power, you can call it epsilon squared, epsilon square together is the symbol, is a notation, it is not epsilon square of that. Epsilon square together is the whole notation and this is, we know E n and I am just rewriting here. Where E n is the same thing, E n is giving D n minus this Y n okay, these are all known stuff.

So, what I have to do? I have to take partial derivative of this quantity, which respect to W 0 equate to 0, with respect to W 1 equate to 0 dot-dot-dot for minima, maximum-minimum problems all of you know. It will give rise to minima if you stay, how do you know it is

minima or maxima? That is you take second order derivative and you will see the sign and all that, that will.

So you just, minima, remember epsilon square is the haunting is the variable. It is not squaring epsilon, okay. This entire thing again I can write in a compact manner. I define, just an operator del W, which is nothing but del, del W 0, del, del W 1 dot- dot- dot del, del W n minus 1, is just notation. That is, this will then mean del W working on epsilon square, that will be a vector of 0, 0, 0, 0.

How? Because del W what will come epsilon square means? First del, del W 0 on epsilon square so that is here, that is equate to 0. Then next is del, del W 1, working on epsilon, del working on epsilon square. So partial derivative of epsilon square with respect to W 1, del epsilon square, del W 1. Okay, there is a next term and there is again 0, 0 here so on and so forth. So using this del operator in the compact way you can write and this vector also instead of writing so many 0s in a vector, you can put a 0 and underline, this will become zero vector.

So this is a question you have to solve. So we have to find out this gradient, epsilon square you have to write as expected value of E square n, E square n, E n you write by D n minus Y n, Y n you write in terms of W transpose X n. That is how W will come into the business then only differentiate with respect to W. there is a coefficients okay and equate to 0. As I told you all second order thing will give rise to first order after derivation and you get a set of equations, first order equation you solve and then get the best solution.

Now to this gradient will give raise to the first order equation, right, gradient. This gradient derivation is part of adaptive filter course, will not do it here. I can only give you the expression is del W epsilon square. It will have worked out, it is twice R, R is these matrix R, auto correlation matrix W. So if you are deriving if you are okay, it is a function of W and W minus P. So gradient varies from choice of W for certain choice of W, it may be high percentages of W will be less and all that.

P is that vector, cross correlation vector, they are assumed to be known. R is assumed to be known. P is assumed to be known, okay. This is a long derivation that I will not do. So, that means this gradient has to be equated to 0. This has to be equated to zero from above. So if you equate to be 0, 2 cancels RW equal to P. So, you get what is called W solution, the solution I write as W opt. If you equate to 0 WR, W equal to P, solution is our R inverse to P, the solution I give a name W opt, opt for optimal.

Which means, if you are giving the autocorrelation matrix of input and the cross correlation vector P, then filter coefficients said W opt equal to R inverse P, just calculated this offline. So, you got a filter coefficients. Now construct the FIR filter that will be very good filter, optimal filter. If you filter the input for that particular autocorrelation matrix, a zero mean. Then it will be output will be the very good estimate for that D n which has auto cross correlation vector P within input vector okay.

So you as long as X n, D n do not change I mean they has a statistical characteristics, there is as long as P does not change, R does not change, you are happy, you are getting a best filter output which is a very good estimate of D n. Now suppose, what happens that input random process changes its characteristics, its autocorrelation values change. So it is no longer R matrix, it brings something else. DN also changes its characteristics. So correlation between D n and X n, it also changes.

So P vector will no longer have the same elements, it might take to P prime, R might change to R prime and it can go on happening again and again, continuously. That means you design the optimal filter once constructed happily, you know you are filtering, you are getting the best output, best estimated output for the given D n and giving X n. But suddenly, X n and D n are not changing their characteristics. So no longer your output is the best estimate of the current D n.

But if you are filter is such, it can understand this change in the input and in the statistics of X n and D n and adjust itself. So that after adjustment it becomes a new R inverse P that is R. If R changes to R prime, P changes to P prime, it changes to R prime inverse people. So new W opt then again it will be a very good filter and it will continuously change itself. Track and change whenever required. Then that will be an adaptive filter. That is what will be targeting. But how to go to adaptive filter? For that you know there is no smooth jouorney.

So I tell you this, then suppose R matrix okay, R matrix this R matrix what is this? Expected value of X n, X transpose n, okay, if I were to have an estimate of some idea about R matrix there is an expected value that means I would take one sample vector X n multiply by its rho, another sample vector maybe X n minus 1 to start at X n minus 1 then N minus 2, N minus 4 dot, dot, dot into its transpose.

Then another sample vector maybe X n minus 2. So you started to small X n minus 2, N minus 3 dot, dot, dot into its transpose. So many such cases you take and then average okay.

Maybe 100 such cases, you take average, you get a good estimate apart. Similarly P vector. How to get a good estimate? You take X n vector with small x n here, than X n minus 1 dot, dot, dot. We multiply by D n to get one vector with some numerical value. Then take X n minus 1 vector. So started at minus 1, X n minus 1 then N minus 2, N minus 3 like that multiplied by D n minus 1. So, you get another vector and this way maybe generate 100 such vectors add them and divide by 100. So you get a good estimate that is how to get RNP okay.

Now, so you can put that RNP here, okay. But before that I, somebody asked me this question, that even if you are given correct R or correct P, do not have to estimate like this. Suppose I do not know how to carry out the inverse of a matrix, okay, can you give me an alternate way where why I do not have to carry out calculation of these R inverse P but I can still obtain W opt offline, but maybe in a iterative way, can I generate this, this same W opt without explicitly calculating R inverse? Because I do not know how to calculate R inverse.

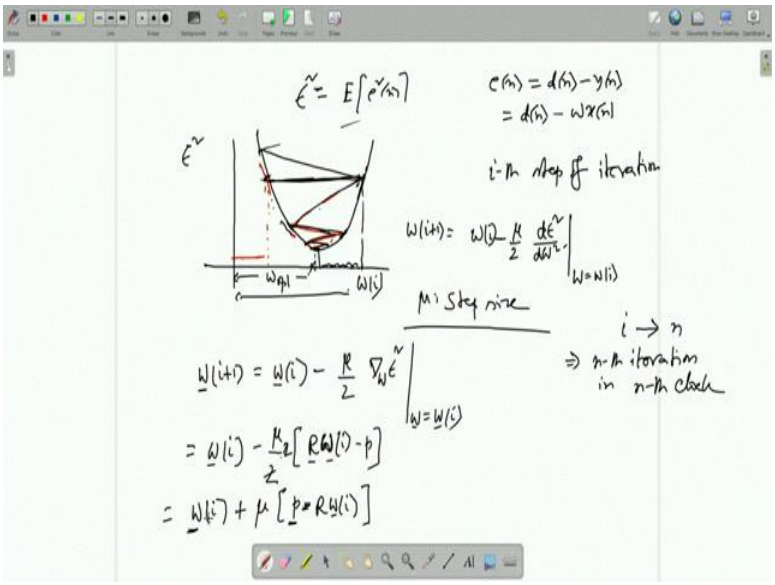
Then I will tell you yes, there is another way, which is an iterative way, not a formula, close for formula way like this, but an iterative way, iterative means there will be a loop for I equal to something to something, in the loop you continuously update something and eventually your filter weight will converse to W opt, which is equal to this there. But that process will (not) I can guarantee you that process will not require calculation of inverse. So, that process is called gradient descent process.

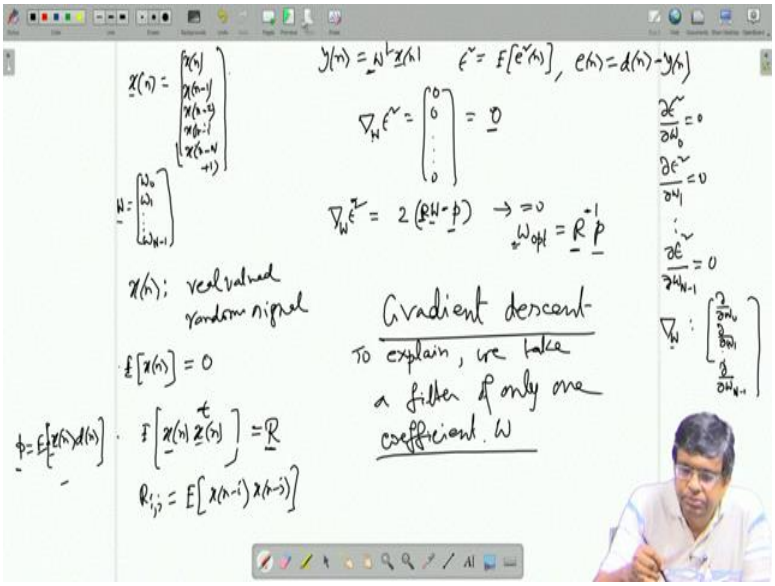
And that will take some you know, I mean, that will initially assume that R is given as it is the correct one. P is given as it is and then later I will say that no R exact value is not file take an estimate, P I will take an estimate, the way I told you from data only I take an estimate that is later. But first let us consider the case of gradient descent. The gradient procedure, gradient procedure. To explain, we take of filter of only one coefficient. That is for explanation purpose. Then I will generalized.

Here I have W0 coefficient, W1 coefficient, W capital N minus for so many, capital N coefficients, but for explanation of this alternative iterative procedure which does not require calculation of R inverse, but which gives this, which in the end still gives you the same W opt, which is called gradient descent process. I will start the explanation by taking a filter with only one coefficient.

Coefficient seems only for one coefficient I take it to be with W, no need to put a subscript here. Like here 0, 1 all that no need. There is only 1 coefficient, okay. I will take that and then I will generalise the ethic okay.

(Refer Slide Time: 15:23)





See this case, in this case your epsilon square which is E of expected value of these okay and E n is D n minus Y n, Y n is nothing but filter output but I have only one coefficient W. So W times X n, no X n minus 1, no X n minus 2 because only one coefficient. So, obviously, epsilon square, if you take this E of n square up. So than this D n minus W x n square up will be D square n, S square n there is cross term. D square n I mean, all those cross terms and all those, all those D n square expected value is known to me. It is a variance from DN.

X n square or D n, X n their expected value is a cross correlation between D n, X n and all those are known to me. But W is not known, it will be a second order function of W, will be D n into X n into twice W as a W square into X n square N and then we have to apply E over them. So, basically it will give rise to a second order function of just 1 W okay. In general is a second order function of all the coefficients W 0, W 1, up to W n minus 1, I mean this one epsilon square.

Now there is only one coefficient, you can directly see, if you square up, this square N expected value known to you, given to you. Than class W square, X square N, X square N if we apply E over them that is given to you. Because it is part of that R matrix and then minus twice W expected value of D n, X n, which is also known it is a cross correlation between D n, X n okay, that is part of the P vector, they are known to you.

But it will be W is unknown it is a second order function of W. And therefore, if you plot epsilon square, if you plot epsilon square versus W, second order function can have either minima or maxima, here it will be minima for sure and then if it is your minima. It can only go up, up, up, if you go to the right or to the left it can never comeback like this. Because then the gradient, here will be 0 again, 0 again.

But, second order function, if you take the gradient equate to 0, you get first order function, second order function, if you take this gradient, you get a first order function. Then equate to 0. So you get only one solution that means, it can have either one minima or one maxima that is all. Where that variant will be 0, there cannot be multiple points of inflection that is, it cannot bend. So that if it bends like this you will have gradient 0 here also, here also, here also.

Therefore at multiple points gradient is 0 that is not possible. Because it is second order. When you derive it becomes first order. First order of an equitant to 0 only one solution okay. Therefore, either one minima or one maxima and whatever it is to the right of that or to the left of that it can only go on increasing it cannot bend back okay, alright. So, and the minima here is that W opt, suppose I am running an iterative procedure at Ith step of iteration, I am in the Ith step of iteration and here.

Either on this side or right side or left side of the optimal point, this is my optimal point. Suppose I am here okay. So I call it Wi, this much is Wi okay. If it is Wi, my W opt is to the left that means I should not go to the right of this point, I should go to the left. If I were here

on this side, I should go to the right. So what I do, I take a gradient of the epsilon square okay, that is C gradient, I am using D notation not del because it was a parameter of, it is a function of only one W in that case simply D not del.

So I think this okay, I evaluated at W equal to this point. At this point, I find the gradient, if the gradient is positive, like for gradient is positive here. I should go back that was from Wi I should subtract some amount. So from WI I should subtract some amount. So what I do I subtract I first take the gradient, multiply by sum up, proportionally constant Mu by 2 okay. What does it mean? And then I call it my new position Wi plus 1. Let us examine this in detail.

Suppose I am indeed here. So this derivative is positive because gradient is positive, okay. So, some positive amount multiplied by Mu by 2. So this entire thing is positive which means W from Wi I am indeed subtracting some amounts, so I am going to the left, okay. If and then if the magnitude of the gradient is high. Than if it is very steep, the amount by which I go back there is large, so I jumped by a huge amount. If I am, somewhere close to the optimal one, there is a slope is not so steep. Then this amount of the magnitude of the gradient is small. So I will go back by a small amount.

If I want this side, if suppose I want this side, if Wi were here. Then my graduate is negative. So this value is negative. So negative and negative positive, so to the Wi this much is Wi I give. I add something because it is positive. So again, I go to the right direction. So if I am to the right of optimal point, I go back. If I am to the left of optimal point then I go forward and so and this continues. So maybe I do like this, go backward.

Now I find slope to be negative. So I go forward. Then again, positive go back and like that, like that finally I am here, okay. This is a iterative procedure. Alright choice of Mu is very important, Mu is called step size, choice of Mu. If Mu is large you will go, naturally the amount of this correction or you know this, this term is called correction term or update term because originally I had Wi now I am updating okay. So, amount of that, the magnitude of the update term will go up, if Mu goes up we go down if mu goes down.

So suppose Mu is large, suppose Mu is very small then from here, you will go down like you know the small amount, small, small and further small. So you will take a lot of time. If we become greedy, now I want to be fast, it will raise Mu, so you jump to this side, there will

because the amount magnitude of this whatever be the gradient that will be high. It is quite steep here, so mean also high, this also high, so product is high, jumping by a huge amount.

Then gradient here is not so steep, you are not jumping by get much but still jump quite much, quite some amount. Then again gradient is not that high but Mu is high. So we are not jumping by that much but still quite an appraisal amount, so we are here and like this, like this. So you are converging fast, if you are still very greedy, you have raised mu finally a stage will come when you will go here and from here you go back here, you go here go back here, you never come down, so you will not converge.

And if you still go up. Then from here you go up, go up, go up. So you divergence that is why mu has an upper limit, you have to choose mu within that, they are all part of adaptive filter theory you do not need them here at all. But you know this since I am talking of these you know discussing this I thought up mentioning them okay. Now instead of one coefficient I have got multiple coefficients. Then I have to do these interactions at the same thing for all of them. So in that case, when I not have only single coefficient W but all of them as before.

So W is a vector Wi sorry, Wi plus 1 will be Wi minus this is a coefficient vector. These is a vector, first guy is W 0 coefficient, here also W 0 coefficient. So from W 0 i you get back W 0 i plus 1, next guy is W0, W1, W1 i gives you W1 i plus 1. So we are writing everything just in a vector form but line by line you can take they will be of this time. So gradient, if you put all the gradients down to the partial derivative because I have got so many coefficient.

So if you put the partial derivative one upon another, it will get backward good old del W epsilon square and W at wherever I am standing, W i vector okay. Now this gradient expression we worked out earlier. If you bring that here it will be W i minus Mu by 2 R, if you see this is expression we had minus sorry, RW, RW and W we have to replace by Wi, RW i minus P, which leads to and there was it two they are agree.

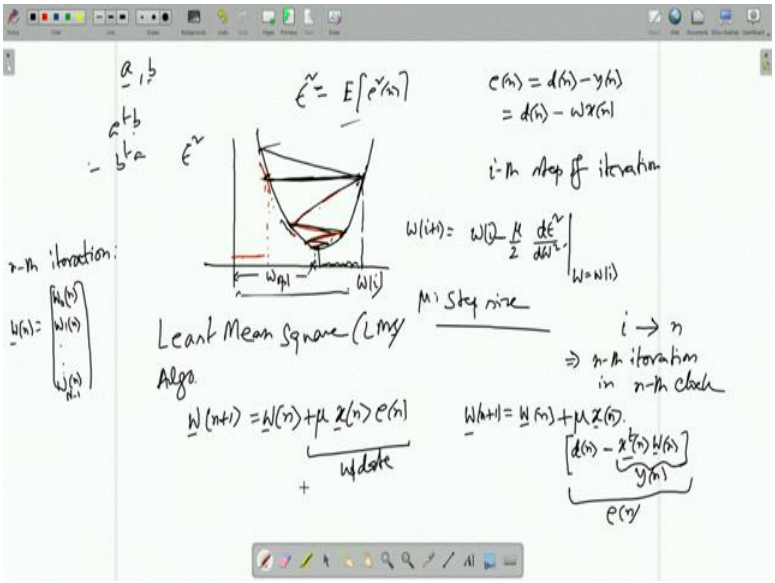
If there is no two there will be two there. Yeah, there is a two here RWi minus P there is a 2, that is why it is Mu by 2 so that the 2 is cancel, okay. So it will be Wi and this I write P first so minus and I think minus out, it becomes Mu 2 and 2 cancels because I have got 2 here, 2 and 2 cancels Mu P minus RWi, okay. So this is the iterative procedure from Wi vector, current iteration vector to calculate this. No R inverse is required, but R is required P is required, this is actually minus and you get Wi plus 1 and go on doing it, then for a correct choice of Mu, you will finally converge at the bottom and we will get back that optimal point

without requiring R inverse. But then you have to go through this iteration again, again, again that we may be lengthy, this is an alternative procedure.

Now, suppose I want to make it online, so I will run the iteration in time Ith iteration, instead of i now I will be adopting an index N, so Nth iteration, instead of Ith iteration everywhere I call nth iteration. Just for some you know I mean this thing, so that we remain in teamed with common notation in adaptive filters.

So instead of Ith iteration I will call it Nth iteration throughout and Nth iteration suppose we are carrying out in real time at that Nth clock, so at Nth clock will calculate W n plus 1 from W n and hold it for using the next clock N plus 1 Nth clock. Again the N plus 1 Nth clock from W n plus 1 will calculate by this kind of formula W n plus 2 and hold it for using the next clock. There is N plus 2th clock, so and so. That is why I will be replaced by N, meaning and also Nth iteration in Nth clock okay. So this entire thing I can rewrite.

(Refer Slide Time: 28:34)



Okay, but still, I still use R reduced P. So my earlier problem remains. That if suppose the signal, input signal changes its statistical characteristics okay. So R changes, D n also changes its statistical characteristics, characteristics. So that cross correlation between D n and X n also changes, so P changes. So, W opt changes, so you obtain W opt by this formula and remain happy but R and P have changed. So W opt you get is no longer the real optimal one.

Okay, you have to adjust yourself that you should still remains. To take care of that, now what I do we know that R is equal to E of X n, X transpose n. How to estimate it? Now, from data you take one sample vector of X n multiply by its rho. Take another sample vector of X n multiply is rho, take maybe 100 such cases add all of them, we are going to take one sample vector multiplied by rho, you get 1 matrix.

Another sample vector of X n multiply by its rho function, x transpose function, which is rho okay, you get another matrix and so on and so hundreds such cases maybe or thousand such cases we add divided by all the elements of the reality metrics by thousand or hundred whatever you get a good estimate that is how to obtain R, similarly P. P is so, you take one sample vector of X n multiply all the elements of D n.

Then again take another sample vector of X n multiply all the elements of D n, like that you generate many such data vectors add them, divide by the number of data vectors, you will get a good average. Suppose I do not mind using a bad average because I later we will see that, you know, I mean, we do not really pay that much price. So, I take only one case instead of taking so many cases I take only X n into its transpose.

There is only one sample vector into its transpose version that is rho, no further averaging just take one case that is, if there is a random variable X, I would be happy to take many cases of, many samples of, many observations, say hundreds observations add and divide by 100 to get an average value. But suppose, here I am told that I do not care if it is a badest event just in one case. So, whatever you observed that itself by take to be the estimate, so I observe X n into X transposon n that itself is the estimate of R I pick, I know some prices to be paid, but actually the price is not big.

Similarly here, X n, D n whatever X n into D n I have that itself is estimate of P n, P. If I bring these things here, this is here and this thing here, you can see X n, you write here X n, then X transposes n, W n. So initially I have X n vector then X transpose n, W n, in P also you have got X n into D n. So in both X n is common in the first, so it will become, you can take X n common then plus Mu, from here also, from here also take X n out, from here you have got D n and from here you have got X n, X transpose n. So X n has gone out as common outside here, So X transpose n into W n, okay.

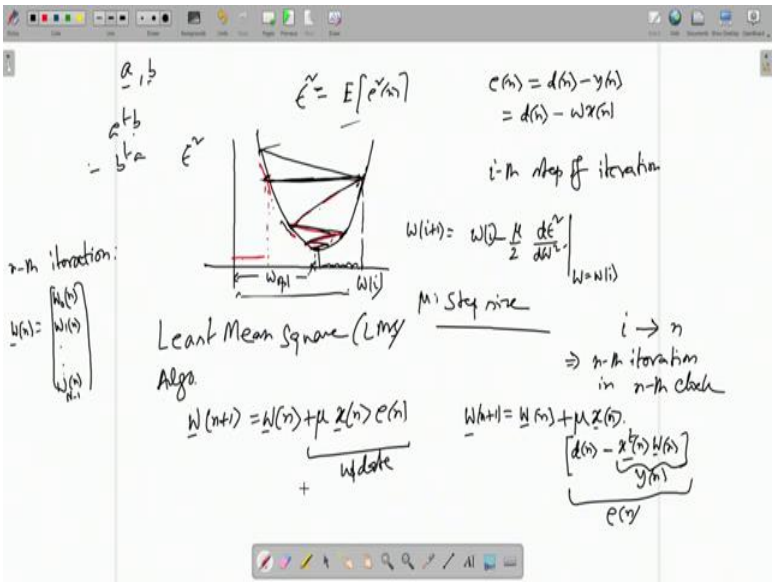
Alright, now just another couple of minutes then I will leave. All right, now what is X transpose n W n? W n is what? W is a prolific filter coefficient vector. But you are doing it

iteratively. So at any iteration, you have got W n Nth iteration, W n will have filter coefficients, W0, W what as before. But they are all now function of that index because they are changing from since its iterative, from clock to clock, they are changing. That is why I am writing as function of N.

Nevertheless, if you have got a filter and you just put this W0 n, W1 n dot, dot, dot Wn minus 1 n output will be nothing but as we see this W n transpose X n, could you save us? So W n transpose X n we can also see this nothing but famous X n transpose W n that is because if I give you 2 vectors A and B. A transpose B and B transpose A, they are same. A becomes rho vector and V column vector. Then you multiply term by term add or B become rho vector and A is column vector multiply term by term, you get the same thing, is very easy to check. So, what is this is? This is nothing but this Y n.

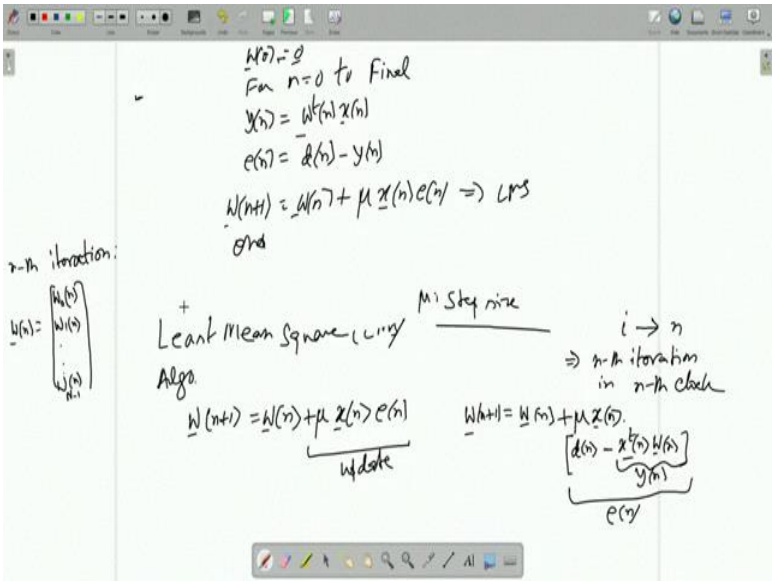
So, DN minus YN means, this is nothing but EN. So, that gives rise to the basic adaptation formula which is called the LMS algorithm.

(Refer Slide Time: 35:29)



Least okay, so this part is the update, adaptation, of course, it is the approximate because R matrix and P vector have been replaced by very bad estimates. So update parties we say noisy, incorrect, some error has gone in. But nevertheless the update part, updated Nth index. Alright update So, at Nth clock what you do then is the following the whole equation is now like this,.

(Refer Slide Time: 36:38)



At Nth clock, you first calculate Y n giving W n, Y n as either W transpose N, X n you calculate. Then you find out E n is equal to D n minus filter output. Then using them E n and all that X n vector, you find out W n plus 1, which is this formula LMS formula. WN plus Mu X n E n okay, we write an algorithm for N equal to 0 to end final. Whatever be the final point, okay start with initial value you need W0, some initial value you can take at all 0 sorry, I am sorry, this one should be here, you started W n, W0 equal to normally we take, this shall has to be zero. Then for N equal to 0 to final end, this is the LMS elements.

The famous LMS algorithms they are 3 steps, in the next class we will develop an architecture to implement this which will calculate Y n from incoming X n and whatever will be the available W n vector. Then calculate error and from W n change it to W n plus 1 by this adaptation formula, use it in the next cycle again and so on and so forth.

Once we have that will try to retime it by cut-set retiming and we will say we cannot retime. Because of the problem that if there will be loop and the loop does not have sufficient delays. Then we will, what you come back to the algorithm again we will go to what is called delayed LMS, for update will be a delayed update. Instead of X n, E n and it will be delayed update. Okay and then we again redraw architectures will see we can retime it that will be very interesting that will do the next class. Thank you very much.