Introduction To Adaptive Signal Processing

Prof. Mrityunjoy Chakraborty

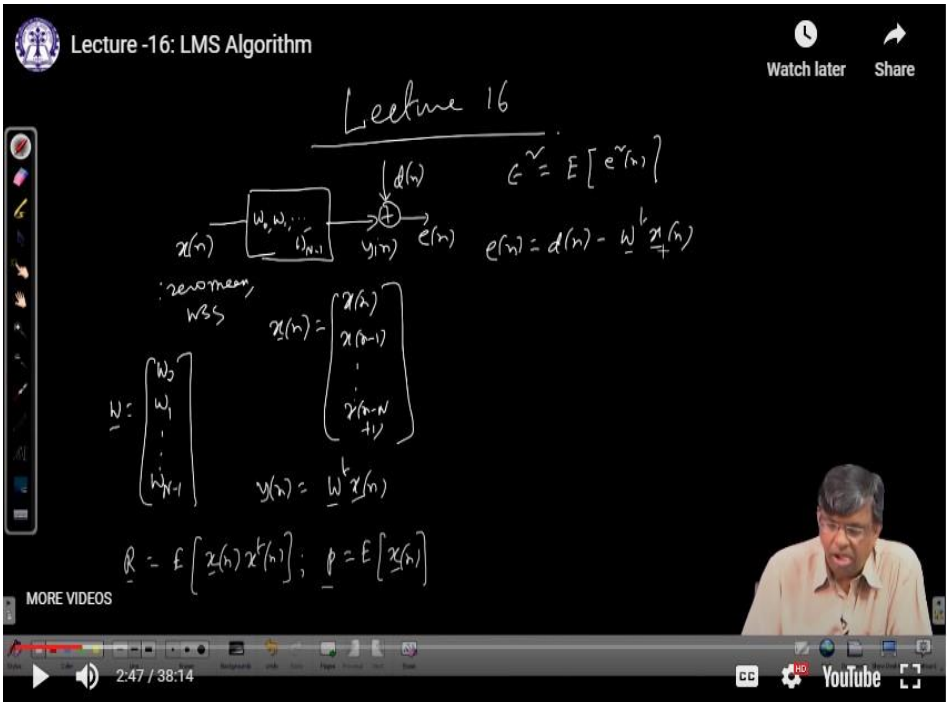
Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology, Kharagpur

Lecture No # 16

LMS Algorithm

So, in the last class we derived the optimal filter it was XN as 0 mean WSS, this is YN desired response this was the error ok. And our purpose is to minimize the error variance and by that process we obtain the optimal filter right Wopt. This is in general the filter coefficients again I write down our vectors and all that W these are all old stuff I am just rewriting for our convenience. These are filter coefficient vector or filter weight vector input data vector YN we have seen is W transpose XN ok we have already seen. Then R matrix is the input autocorrelation matrix because of WSS nature of XN all the samples it was independent of N and the cross-correlation vector again XN and all its samples and DN and all its samples they are jointly stationary. Therefore, this vector this XN vector every term multiplied by DN and expected that is a cross correlation and they are all independent of N. So, this vector is a cross correlation vector P then what you obtain is this epsilon square was expected value of e square N that is output error power expected average power of the output error and that we try to minimize ok how because EN was DN minus this is YN YN is W transpose XN.



If you really expand W transpose XN as nothing but W0 current sample plus W1 is a convolution XN minus 1 dot dot dot dot WN minus 1, XN minus N plus 1 and then if you square up e square N so square up than expected. So, there will be from this side when you square it up there will be X square N its expected value will give the variance, but there will be W0 square there will be the W1 square ok. Variance of correlations are given to us. So, XN into XN minus 1 there will be a term W0 W1 corresponding product here will be XN into XN minus 1 expected value of that this correlation with gap 1 which is given to us all correlation values as may be necessary and given, but in terms of W0 W1 you see there will be a W0 square term W1 square term dot dot WN minus 1 square term W0 W1 W0 W2 W1 W2 all second order then d square N and then again dN with when you multiply dN with this term W0 will remain as it is there is no W coming from this side.

So, they are first order terms W0 W1 W2 just XN multiplied by dN or XN minus 1 multiplied by dN and so and so, but here when I square it up this part W transpose XN which is this then I get second order term. So, what all it is a second order function and we actually work it out it was variance of this dN minus twice W transpose P this is the first order term W0 P0, W1 P1 like that which is coming basically by from the product between dN on one side and the terms on this side. So, dN into XN dN into XN minus 1 they give

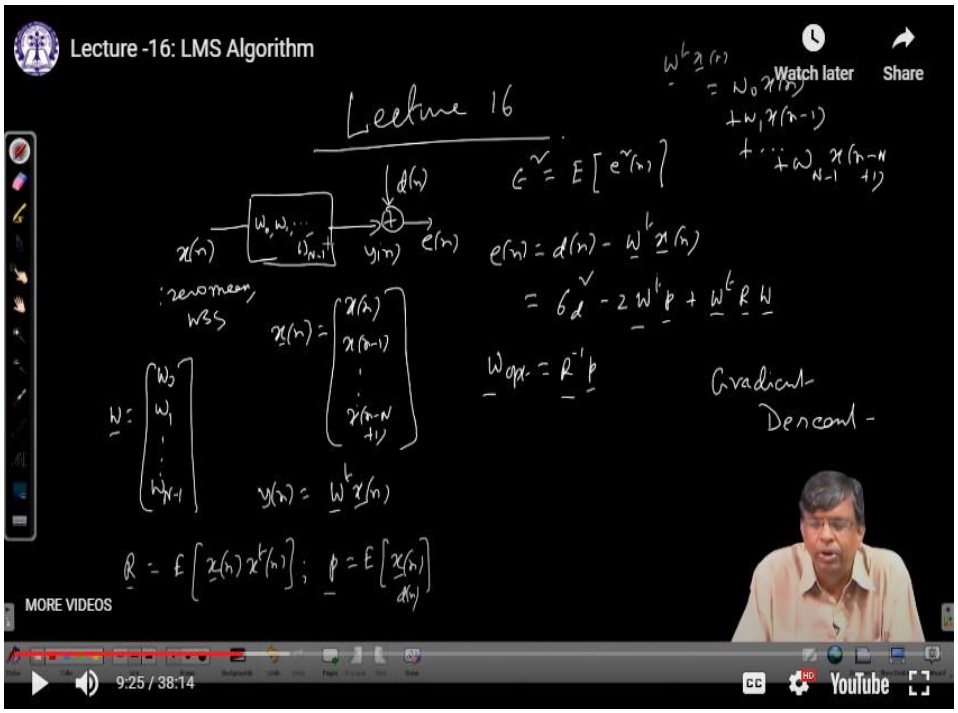
the elements of P vector, but we all have only W0, W1, WN not W0 square W1 square etcetera that is this is that linear term first order term and from here the other one was W transpose R W this was I mean the second order terms like if you square it up whatever you get that is what we will get here. So, in general this was a compact expression in terms of vector and matrices this is just elaborate expression, but in either case this variance is a second order function of the filter weights because it has got second order terms like W0 square, W1 square, W2 square W0W1, W1 W2 and dot dot dot also first order terms W0 W1 W2 all multiplied by some scalars with the correlations of variances. Then I said any second order function will have a unique minima maximum in this case minima.

So, I just take the partial derivative with respect to each weight of this equal to 0. So, I get capital N first order equations because second order stuff when derived with respect to these terms W0 W1 etcetera you get first order. So, you get capital N number of first order equations involving N unknowns these weights solve them you get this W opt right. Now, here the problem is that is W opt was R inverse P here problem is you need to know R and P and you are assuming R that is input autocorrelation matrix and P there is a cross correlation vector between xn and dn P is xn dn sorry I missed out P they are fixed they are not changing they are fixed all right. But in practice it does not happen like that often input both xn and dn their properties change statistical properties change because the circumstances the forces we generate them, circumstances we generate them they change from time to time as a result R changes from time to time P changes from time to time.

If you design the filter with one R inverse P for certain R and certain P that is an optimal filter it will work fine for a while, but if R matrix changes and the P vector changes due to environmental changes outside circumstances changes outside, then the filter that we are using now will no longer be the optimal filter for the new input statistics. But you cannot go on changing you cannot go on you know disbanding the circuit manually and then again recalculate and again you know construct the circuit again use it for a while, that that is impractical that you cannot do. So, we look for a filter which will learn from the input incoming input learn the changes in R and we will adjust the filter coefficients iteratively that is clock cycle iteratively. So, that if R changes and P changes it will take these

coefficients to the new W opt which is new R inverse new P new P ok and it will go on such a filter will be called adaptive filter. So, there will be an adaptation process by which the filter coefficients will be adapted, clock after clock or index after index anything index 2, n plus 1 is index 2, n plus 2 is index therefore, will be changed by an adaptation algorithm that is what is needed and I have to go there from here.

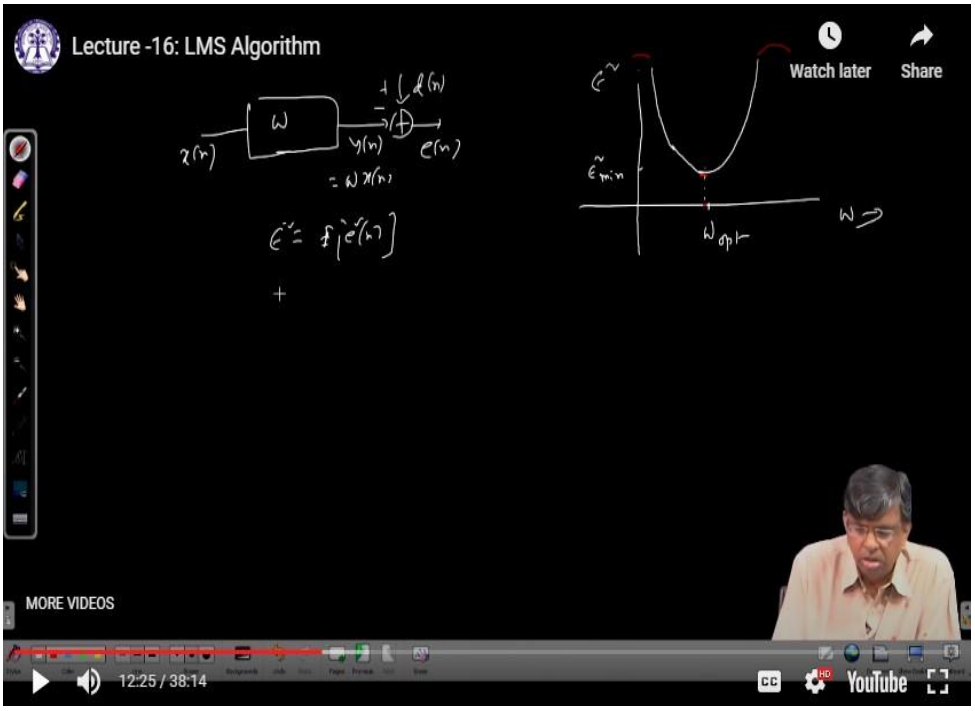
But there is an intermediate step called gradient descent step gradient descent ok. From here we have to go to the gradient descent approach from there I will we are going to arrive at the adaptive filter and again I have taken all real case here fine. To go to the gradient descent step from here I have one particular way of you know explaining that is this. That suppose you know R you know P, but you do not know how to calculate R inverse it is a big matrix you do not know. So, you tell me that look I do not know how to calculate R inverse. So, I cannot calculate W opt though it is offline calculation only, can you suggest me some other method may be an iterative method where I run iteration after iteration after iteration, but no inverse of R is required and it takes me to W opt in the end I say yes and that is this step.



Now, that to go to that step we first note that if I plot this epsilon square as a function of all these weights. So, it will be a quadratic surface quadratic function because it will have a unique minimum, that will have a unique minimum it is a quadratic function. Let me show this by taking an example of just one filter coefficient not W0 W and all that suppose only W Xn. So, Yn which is just W times Xn as an example. So, En is Dn minus W Xn you square it up Dn square W square Xn square expected value will be R00 RX is 0.

So, there will be a W square RX is 0 0 there will be Dn square expected there is sigma D square and there will be a minus twice expected value of W Xn Dn which is a cross correlation. So, it is a second order thing a sigma D square minus twice W into a scalar plus W square into a scalar from second order. Any second order function now, if I want to plot it I have taken only one coefficient because then it becomes easy to plot and explain. So, this epsilon square in this case is as before. So, epsilon square it is a quadratic function of W and as I know any quadratic function will have a unique minimum in this case minima.

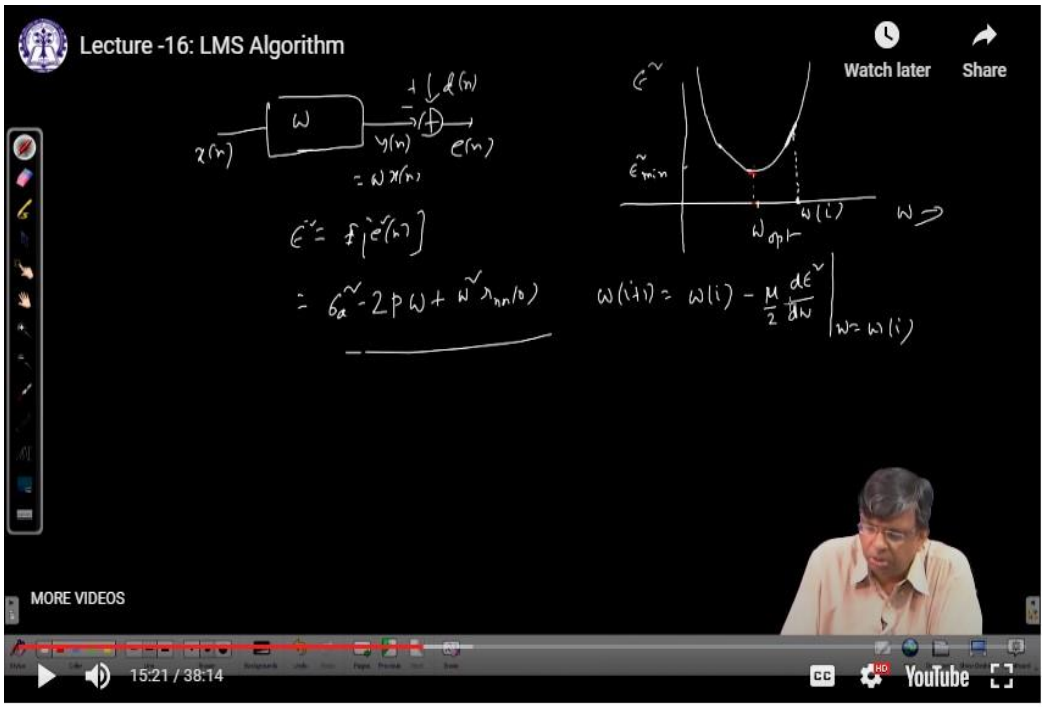
So, it will have a minimum maybe a W opt and this is my W axis and it will have minima value this much. And moreover, if I go along on both sides, it will only go up up up, it can never it can never come down like this or come because when I differentiate it with respect to W, I get and equate to 0, I get only a single solution not multiple solution. So, gradient that is derivative will be 0 only at one point. So, if it is that point where is minima it can have a unique minimum, only one minimum it cannot have multiple minima or multiple maxima. That is if I derive with respect to W I get a first order equation first order equation has one solution. Like here e square n if you plot it en is dn minus W xn as I told you dn minus W xn you put here and square up an expected value e of d square n will have sigma d square maybe I use this pen.



Minus 2 e of W xn dn, so W is outside and xn dn is a cross correlation you can call it p small p, just one coefficient I am calling it p and W and then W square xn square W square xn square expected value will be R0 Rxx0. So, it will be W square Rxx0. So, the second order stuff you see. So, if I derive it I will get a first order equation and when I make that equal to 0 and find out the minima maxima I will get a unique minima only one solution. So, if that is the unique minima, I cannot have multiple minima which means this function can only go up up up it can never bend it can never bend that is most important ok.

Whether it is a function of this one weight or have got all the weights. Iit is a quadratic surface in this case a quadratic curve in general a quadratic surface. Now, suppose I am running an iteration ok and at ith iterate I have got a coefficient Wi all right ith iteration. So, it can be either to the right of this or to the left of this I am not assuming this is equal to this because there I am already at the solution. So, that is a trivial case. So, let us suppose it is on this side at ith step of iteration I got some value of this some estimate of this Wopt that is W equal to Wi which is in this case is to the right of this could have been to the left as an example I am taking on the right.

What I do then I find out a Wi this value epsilon square and see the gradient at that point and from this I subtract I will explain I subtract I first take the derivative, in this case I can write in terms of d with only one variable. So, d dw there is a derivative and that I evaluate at W equal to W that point Wi and I take something proportional to this, there is a proportionality constant mu, mu times that is a quantity which is proportional to the gradient. This is a gradient proportional this is a proportionality constant it is called mu instead of mu, I take it mu by 2 because then a 2 will come after sometime as it will cancel with this mu by 2 is a proportionality constant. This is what I do I subtract it from the current iterate and whatever be the result that I take to be the value in the next iteration. What I am doing here actually you see if I am on the right side of Wopt then gradient is positive.



So, this is positive and mu is positive of course. So, from this I am subtracting something positive that means, from here I am not going to the right side I am going some I am subtracting something positive means I am going in this direction, which is the correct direction to approach Wopt ok. So, maybe I cross this I subtract something it takes me here on the now if I am here gradient is negative. So, negative negative positive, so I add

something to this iterate. So, maybe it takes me there and like this like this like this finally, it will converge here ok this is the method.

This mu is very important mu is called step size. Remember one thing I am taking this to be positive or negative, if it is positive, I am subtracting, but the amount by which I am by way of subtracting that this whole quantity also depends on the magnitude of this. If I am here where the gradient is very steep value of magnitude of the gradient is high. So, this quantity is called the upgrade part, upgrade part or increment or decrement part that itself will have large magnitude. So, I will that is why I will jump here, but if I was here then the gradient may be positive, but magnitude of the gradient is small.

So, this whole thing will be small. So, I will be just going by a small amount. So, when I am further away from on either side from the Wopt this came will take me will go by a big jump, but when I am around this, jumps will be small, but will go like this ok. And this is this upgrade part also proportional to mu if you make very small even if this is large this whole quantity will be small. So, from Wi you will be going back by a small margin this small margin, small small further small small and like that.

So, it will be taking lot of time to convert. On the other hand, if you now become greedy you say no let me increase mu then of course, it will be it can be like this it goes here, then here, then here, then here, then here, then like this. Finally, you say no I am still greedy I have to make it still further. So, let us increase mu further then it may so happen you go here you subtract and go here, but next time then from here you go here.

So, it will be like this. So, we will be oscillating between the two will never come down. So, that is a point of divergence you cannot cross mu take mu to that level. So, that this phenomenon occurs then it will never come down and converge here. If I still become greedier then from here, I will not go here I will go further to the left. So, I will go up to this and from here I will go like this then like this I will go up it will diverge.

This shows it has it has an upper limit for convergence alright. Physically it is clear mathematically it can be worked out, but as I told you steepest descent is not my cup of tea here, because I will be this just an intermediate stage to take me to adaptive filter ok. Descent because I am subtracting. So, from top I am coming down coming down say descent and steepest because actually I am going down along the direction of the gradient. If it is a function of many variables suppose it is a function of two filter weights W0 W1 then this plot will be a bowl kind of thing.

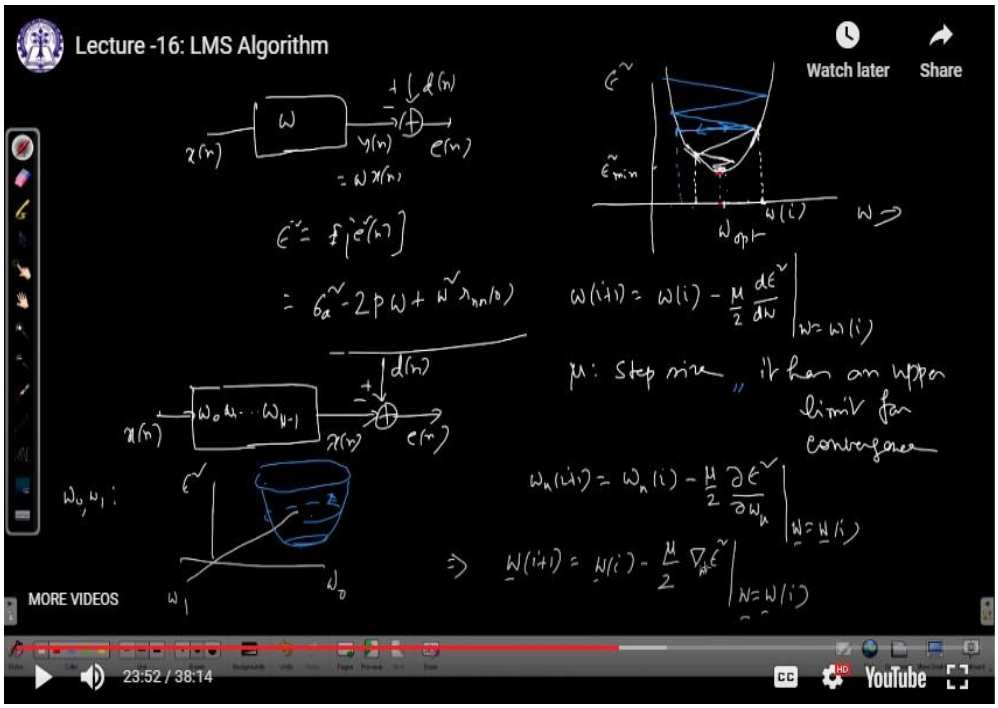
So, at any point I take a gradient both with respect to W0 W1 and I carry out the subtraction. So, basically, I go down along the direction of gradient that is the steepest gradient. Anyway, this was for one coefficient right. Suppose I extend it to multiple, that is now I got W0, W1, dot dot dot W N minus 1. In this case for example, if it is only W0 W1 this whole curve this will be a 3-dimensional plot this is W0 axis this is W1 axis and this is epsilon square.

In that case your plot will be like this. It is a bowl cup shaped ok. So, here at any point you go down by subtracting because at this point, I have both this point is a function of both W0 and W1, this point I can have gradient both with respect to W0 also W1 also. If both gradients are positive, I come down ok like here if the both gradients are positive, I come down ok. If one gradient is positive one gradient is negative in one case, I come down another case I go up and like that ok. So, I follow the gradient and that, but by that I am coming down and down vertically in the vertical direction.

So, that is why it is descent. So, in general to play with same so I can extend it to now not just one filter coefficient all the coefficients together W0 i plus 1 is W0 i minus mu by 2 this derivative with respect to W0. Not others W0 gradient with respect to and it will be partial derivative now del epsilon squares del W0 at that point where overall W is Wi ok. That is this is W0 I, this is W1 I, this is Wn minus an i at ith iterate their values. So, W standing there, there I find the gradient subtract that is for W0. In general, for any Wk it

will be minus mu by 2 partial derivatives now because is a function of so many not one. So, Wk derivative with respect to Wk I go in the negative I mean I subtract this derivative from Wki.

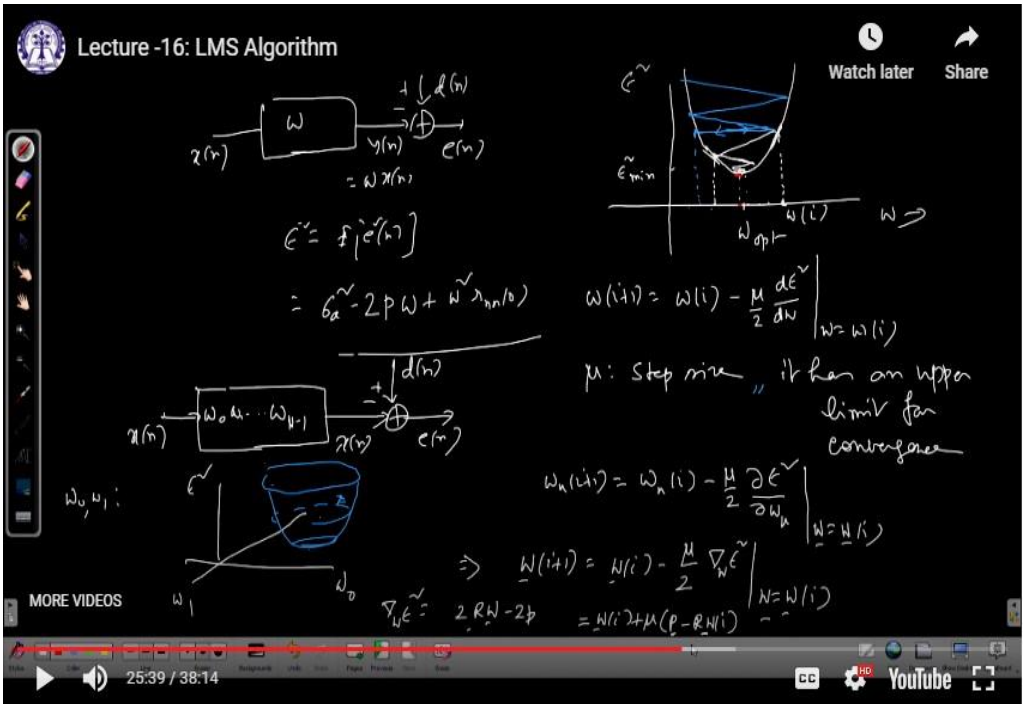
But this has to be evaluated when W is standing at Wi that is W0 I, W1 I, dot dot dot Wn minus 1 i because epsilon square is a function of all the weights. So, this derivative also will be a function of all the weights I am deriving with respect to Wk, but after derivative still it may be a function of all the weights. So, there all these W's have to be put in W0 i, W1 i and all that. And now if I stack all these updates on the left-hand side on the right-hand side. Then it will be one vector with W0 i plus 1, W1 i plus 1, W2 i plus 1, dot dot dot which is nothing, but in the vector notation Wi plus 1 and that is your Wi minus mu by 2 and now I have got del notation del epsilon squares del W0, del epsilon square del W1 they are stacked. So, it is del W so that is a stack of the derivatives evaluated at this is the steepest descent algorithm.



This derivative we have already found out in the previous class this derivative and that was, that was what? del W epsilon square, it was equal to twice r W, if I recall correctly

minus twice P, you can check this is what we derived. So, you can put it here this 2 is common 2 and 2 cancels take minus out. So, you have Wi plus mu P minus r W and put W equal to Wi. So, this is the equation. If r and P are known there is no inverse of r required here. So, that is what you wanted no inversion of r fine.

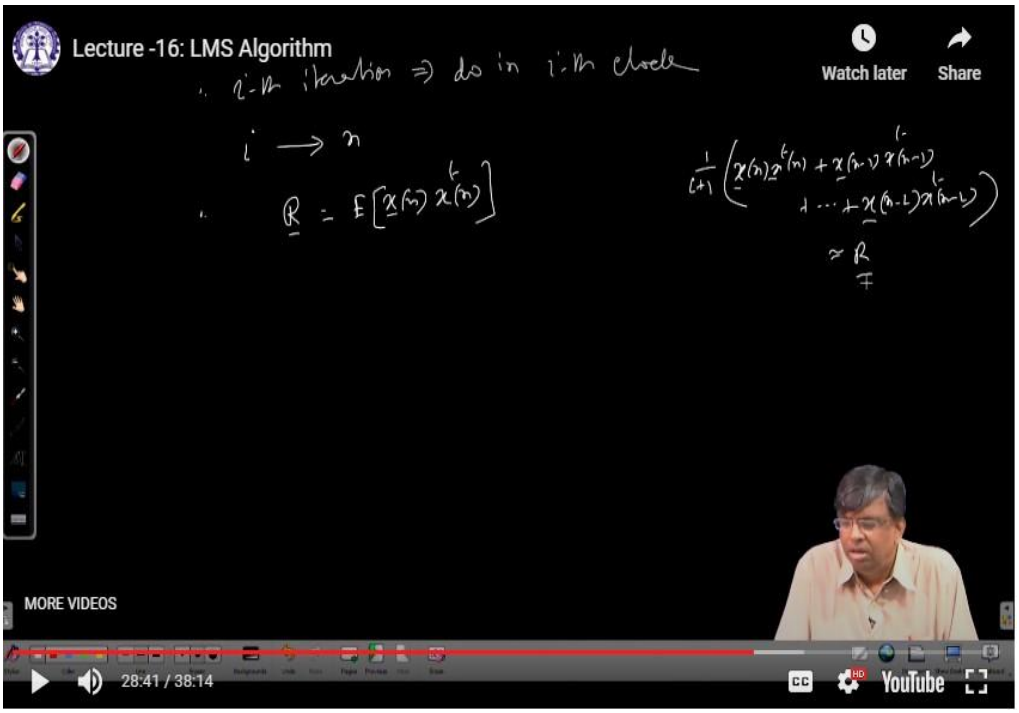
So, go on from Wi to Wi plus 1 calculate like this, then again like that Wi plus 2i plus 3 like that and if mu is chosen correctly within the limit this diagram itself shows that it will go like this like this, like this finally, will hit the optimal vector optimal weight exactly this absolute convergence exact convergence this is a method of steepest descent. But still you see I have that problem this also depends on knowledge of exact r, exact P, exact r, exact P ok. So, that is what I want to avoid.



So, what I will do let us assume that I am carrying out the iteration in real time that is we are carrying out the iteration by looking at a clock, clock means the clock waveform not your wristwatch clock waveform. So, nth iteration or maybe ith iteration, ith iteration do in ith clock So, I choose change replace i by n because normally you know n is the letter that is used typically to indicate a clock, nth clock or nth index all that not ith index.

So, at nth clock we carry out nth iteration I prefer n rather than i. So, i replaced by n. So, that whole iteration you are looking at the waveform, at nth clock you are doing the nth iteration that is from you are going to wn plus 1, then at n plus 1th clock, you are doing from wn plus 1 to wn plus 2 and so on and so forth all right. So, we are running in real time suppose I do that and also, we know r this is E. Now, suppose in real life you want to estimate this what you should do you should take one sample vector fully call numerical data I repeat numerical data, random numerical data take this matrix.

So, you get a matrix of data then take another case maybe x n minus 1 vector their vector transpose. So, again some data here is transpose another matrix and dot dot dot dot maybe you take many cases n minus L and average them. how many cases I have L plus 1, 0 1 2 up to N plus 1. So, this will be a good estimate of r is it not you take one data vector multiplied by its transpose another data vector multiplied transpose this thing. So, you are taking case by case some I mean basically it is called sample averaging. you are taking samples of x n obtaining one you know one instance of this matrix, again samples of x n minus 1 another instance of this matrix, x n x n transposes this matrix ok. And like that and then you average, more the number of such cases less is the average.

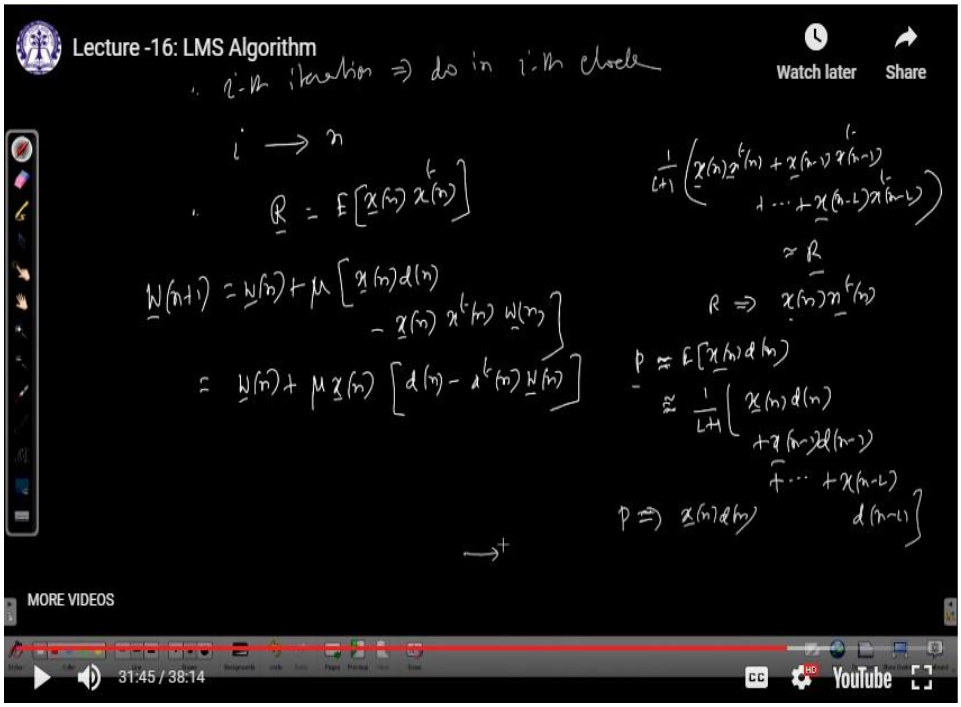


But suppose to start with I am not bothered about I mean how good the estimate is, actually it will be very good estimate we will see later, but I just say that I will take only one case L equal to 0 only one case. So, it will be bad is so called quote unquote bad estimate, but suppose I do not mind. I can still show convergence all that. So, in that case we replace r by just one instance only, just one sample vector into its transpose that is all, it is not a good estimate because I am not doing averaging, but suppose I do that. Similarly, p vector p vector approximately now what is p vector E of x n d n ok. Which means approximately you should have x n one sample data vector random data vector times random d n then x n minus 1 vector into d n minus 1.

So, another random data vector d n minus 1 and dot dot dot dot dot x n minus L, d n minus L ok. But again, I do not mind having a so-called quote unquote bad estimate I will just keep only one. So, p I will replace by just x n d n. In that case let us see what happens to that, this one i goes to n. So, it will be w n plus 1 at nth clock it will be w n plus mu p I will replace by what estimate I got r replace by what estimate got a w n ok.

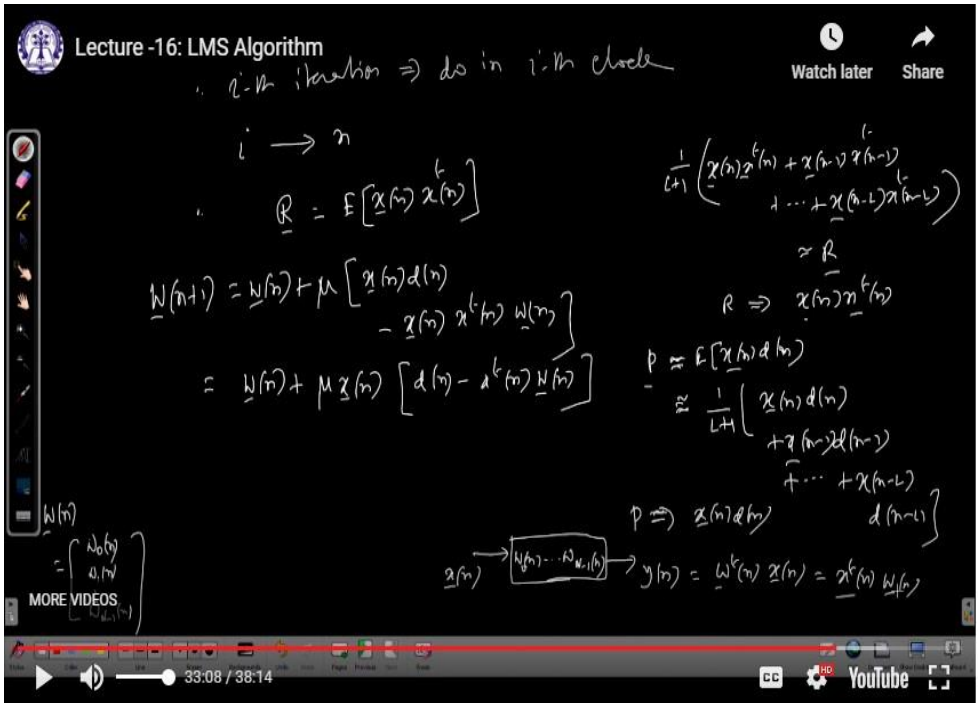
So, I will get w n plus 1 as w n plus mu p minus r w n, p, p is x n d n x n vector into d n minus r w n, r is x n x transpose n column vector row vector and w i it was now it is w n because i is replaced by n. Now I take x n vector here also here also I can take it common. So, let x n come out. So, I am left with d n here.

So, x n into d n and x n into x transpose n w n.

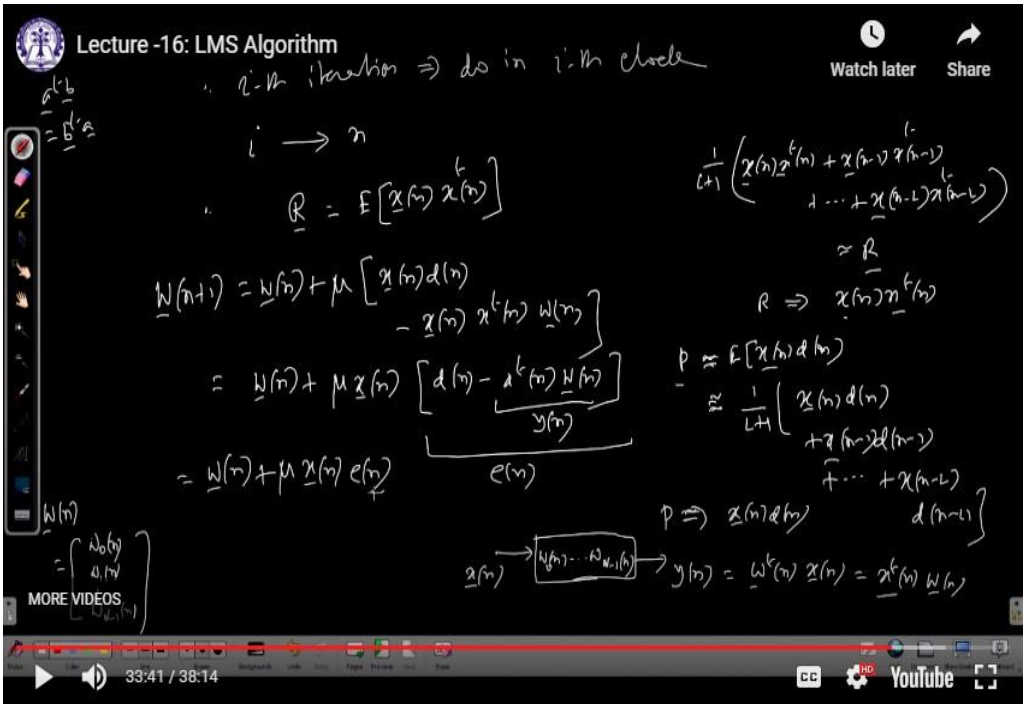


Now suppose I constructed the filter, this filter coefficients at the nth clock or not just w 0 because they are changing from clock to clock ok, iteratively because I am moving them from w n to w n plus 1. So, at nth clock I have got w 0 n to w n minus n ok. Then and you define w n ok. At nth clock I have got the filter coefficient vectors nth iteration, nth iterate or nth clock that is why n is coming ok because I am changing from n to n plus 1. So, very next clock their values will change that is why I have to bring in the n here to show their dependence on n and this w n this is your x n vector.

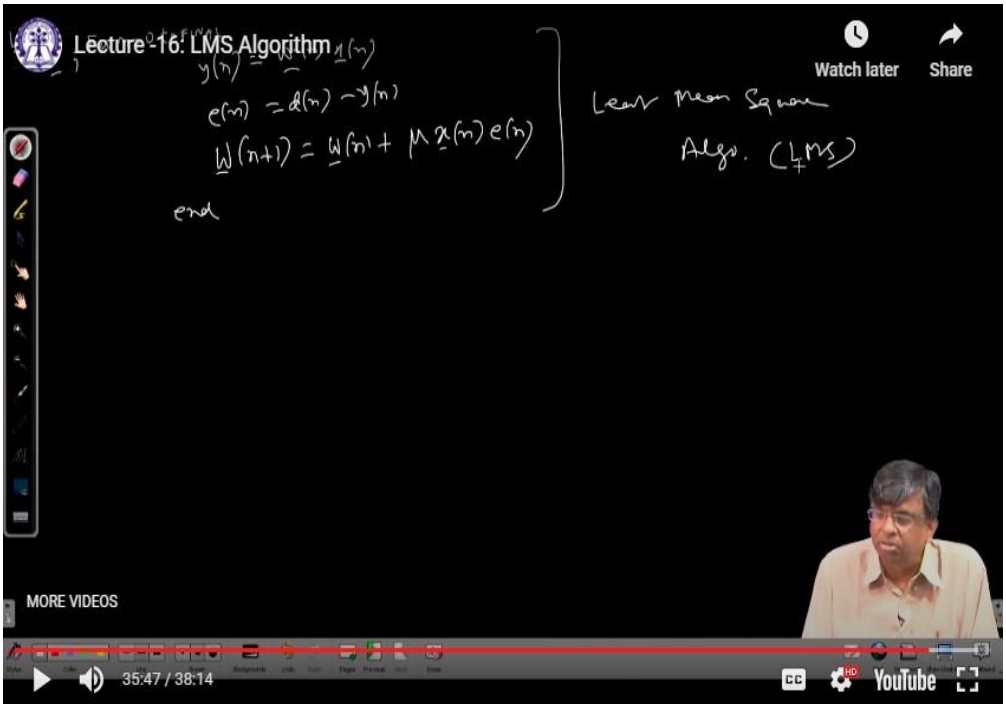
So, output y n you know will be w transpose n now this vector transpose w transpose n x n, but that is also same as x transpose n w n.



You know that if I give you two vectors column vectors a and b, a transpose b is same as b transpose. a is it not a 1 b 1 plus a 2 b 2 plus a 3 b 3 like that or b 1 a 1 plus b 2 a 2 plus. So, that is what I have here. So, this is y n and d n minus y n is the error e n. So, that is that equation which I will expand in the next class.



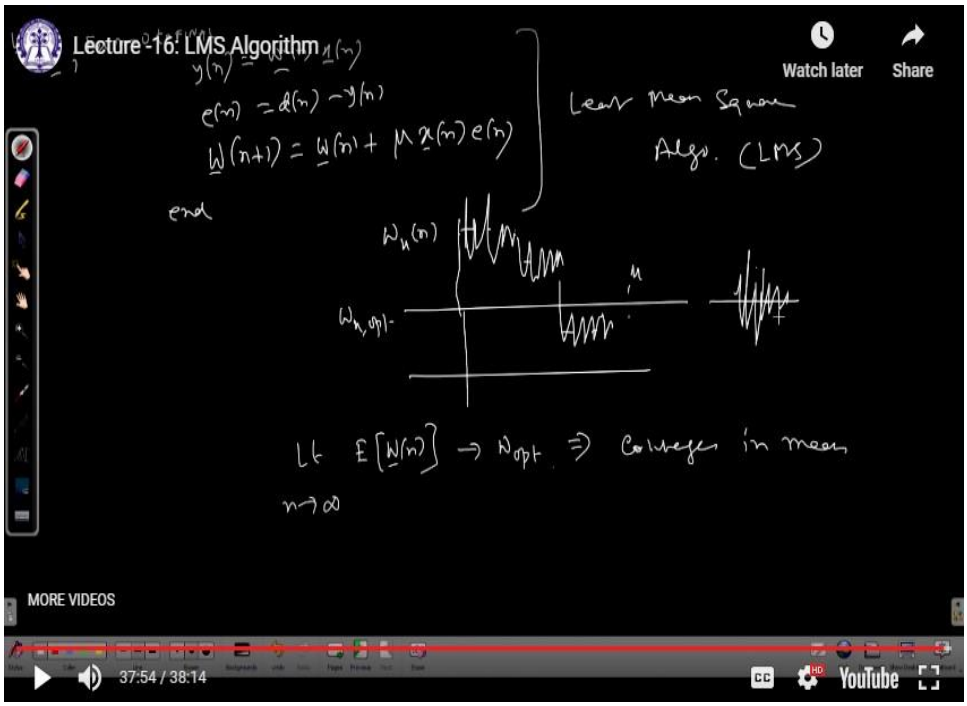
This is a famous adaptation equation called LMS. So, I rewrite I will just take one more minute. So, this is my equation update or adaptive filter equation. At nth clock I got some filter weights I change it in the next clock to this by what by adding an extra term called update term, mu into sorry x n e n, where e n is d n minus y n, this desired response minus the filter output and filter output it uses the filters here coefficients here and the input vector you can write this way and this will continue for n equal to maybe 0 to end n, equal to let me write this somewhere else because there is no space here. So, for n equal to 0 to anything may be fine some index of your choice call it final. you do this end only thing is at n equal to 0, w 1 it requires w 0. So, w 0 is an initial value that you have to provide any initial value will work normally we prefer w 0 as 0 vector. This is called least mean square algorithm LMS the one of the two most popular famous adaptive filter algorithm.



It will also converge, but remember R matrix was not given the proper value P vector was not given the proper value just some approximate I mean wild estimates were given x n into x transpose n and x n into d n. So, for that you pay some price here the convergence suppose you take a particular filter coefficient w k n and this is the w k opt this will not converge on this absolutely unlike the previous case of steepest descent what it will do this is also now random because you see from nth index when I go to n plus 1th I am adding an

update term which are data dependent and data is random which makes it random. So, it will be randomly fluctuating. So, mean of fluctuation average maybe here, after sometime average might be here, then maybe here like that like that, but finally, it will be oscillating randomly fluctuating around the w k opt, this is what we can ensure. It will not converge on this absolutely. So, what we will observe what we will say all the weights if you take the mean that is the average and go in the steady state means for large n that will go to optimal it is only for kth case here all the cases put together.

So, as n tends to infinity expected value of the fluctuations of each that is the average was here, later here, later here or in the end average of the fluctuation we can say it with the corresponding optimal. So, it will converge in mean that is a problem of LMS it is not converging absolutely. now you can ask me fine it is converging in mean, but if still the range of fluctuation is large then what will you do sometimes jump here sometimes jumping here. Then we show and we will not be able to maybe show in this course the theory, but I will tell you there is a way to keep this fluctuation range small by again mu if you choose mu less than this also will be less. So, first LMS algorithm converges in mean and then by appropriate choice of mu you can you know keep the range of fluctuation very small around the optimal one so that is fine with us.



So, this is the famous algorithm LMS we will prove its convergence also maybe in the next class. Thank you very much.