Introduction To Adaptive Signal Processing

Prof. Mrityunjoy Chakraborty

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology, Kharagpur

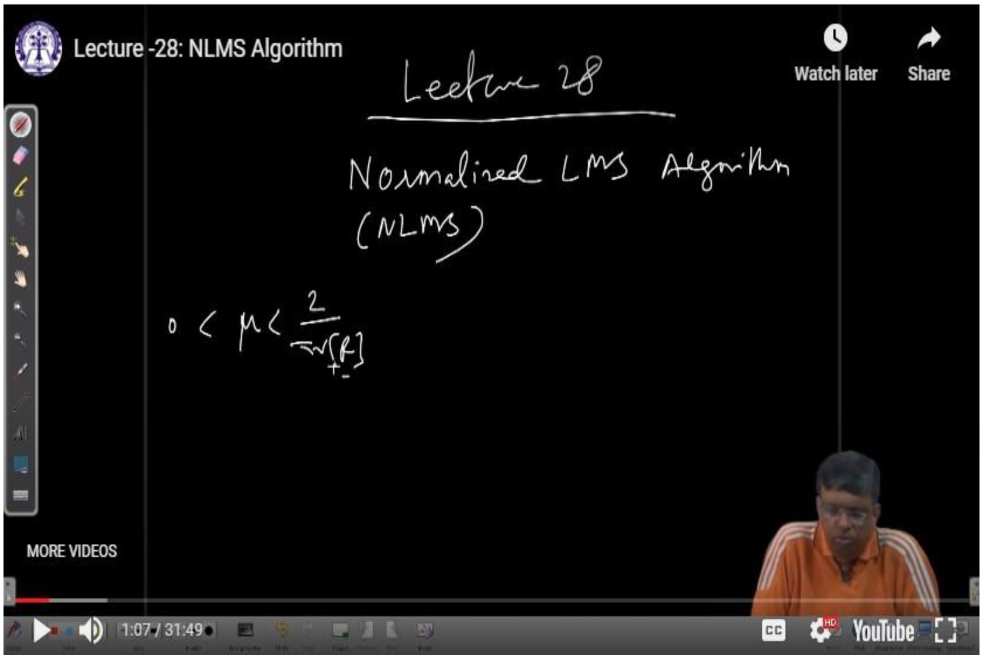
Lecture No # 28

NLMS Algorithm

Previously you have seen in LMS algorithm converges in mean required to be mu by trace r, where r is the input autocorrelation matrix. Now suppose input statistics changes from time to time. So, autocorrelation matrix also changes, trace r also changes. So, it will be very difficult to set an upper limit of mu, because upper limit is varying. Here the upper limit is independent of you know things like trace r which depends on the input characteristics. And it is more often we find NLMS used in practice rather than LMS.

$$0 < \mu < \frac{2}{Tr[\underline{R}]}$$

Now there are various ways of introducing NLMS algorithms. We will show it from a system identification point of view.



Suppose we have an unknown system with two coefficients, the impulse response having two coefficients say fire filter maybe w0 cap w1 cap all right. And also, we consider no observation noise.

So, this basically there is an unknown system xn comes ok output itself is dn ok. So, to find out so there is an adaptation there is no noise here. So, some adaptive filter which will be basically these are original coefficients, but when it is adaptive it will use this error sorry not this. So, this is your dn. So, this error and this input ok.

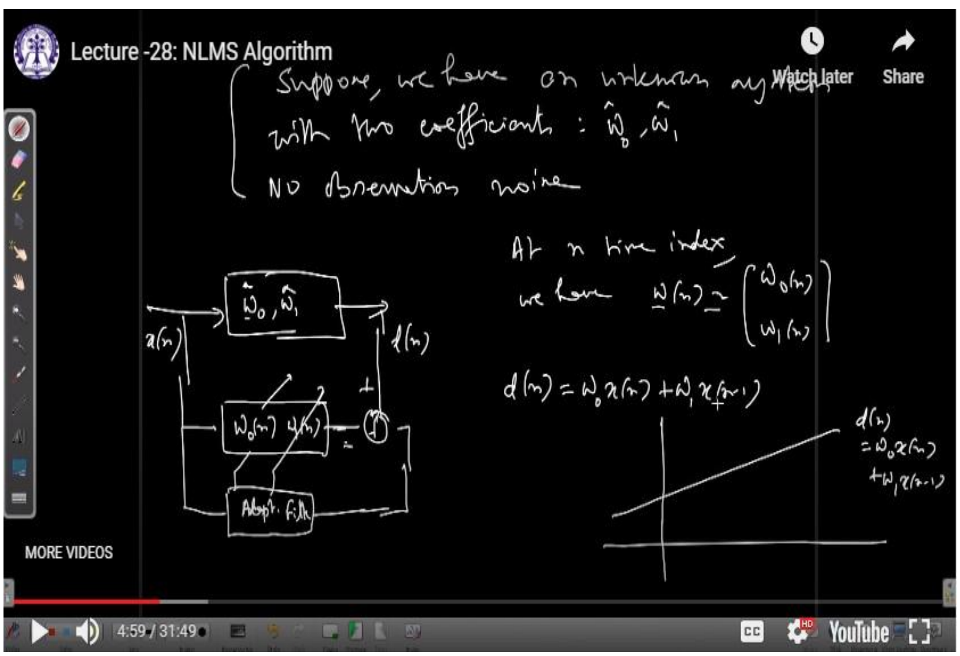
So, this is the adaptive filter. We all know this you have studied this this is adaptive filter sorry ok and by this we adopt them. Now suppose at nth time index we have wn that is w0n w1n. So, we are erupting. So, they are not equal to w0 cap and w1 cap.

So, there is a difference all right. Now you see I know I do not know the coefficients, but I know there are two coefficients. So, I know in general system output dn will be some general w0 xn and w1 xn minus 1, but w0 and w1 could be w0 cap, but w1 cap also, but this is an equation with two unknowns w0 and w cap ok.

$$d(n) = w_0 x(n) + w_1 x(n-1)$$

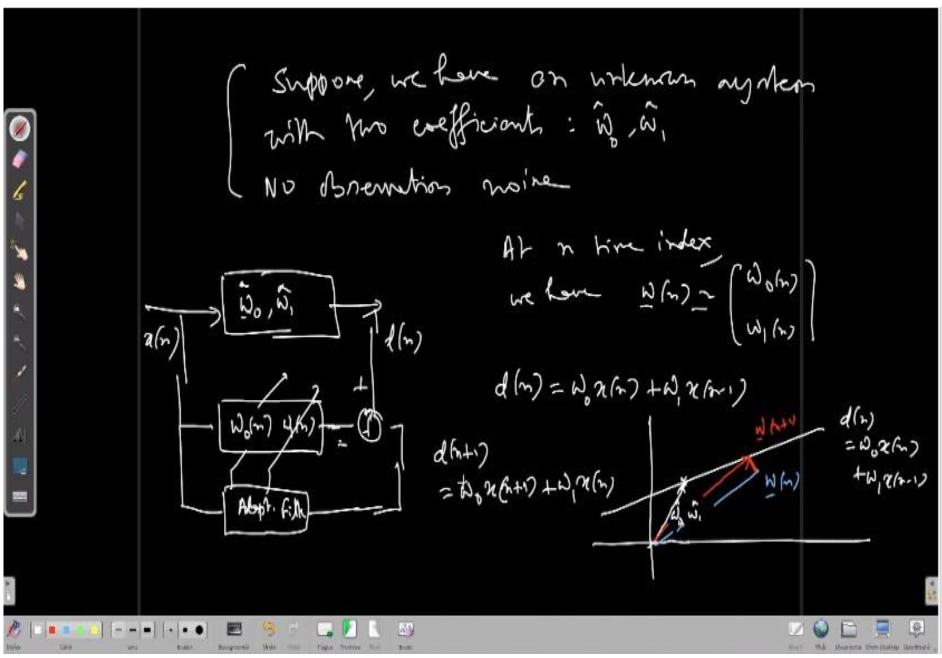
Therefore, there are infinite solutions of which one could be this, but I do not know these optimal ones I know input and output can be modeled like this because it is a two coefficient FIR filter system. So, output will be in terms of input given by this, but w0 and w1 are the general filter coefficients.

So, this is a straight-line right this is a straight line and suppose I draw like this this is the straight line at nth clock that is dn is equal to. So, this straight line contains all the points w0 w1 which satisfy this and therefore, contains this also because these are two coefficients they obviously, satisfy these equations. So, maybe this is that point. So, this is your w0 cap w1 cap this point this is optimal point and suppose currently I have wn. So, this part is wn well this is becoming parallel.



So, this is wn let me erase this because by mistake I made it parallel. So, why should it be parallel? The original one true, true value ok and now suppose this is your wn, wn is the adaptive filter coefficient vector at nth clock and wn is we want wn to converse to this this point as n goes to infinity. So, if what we do in this algorithm, we take a perpendicular we draw a perpendicular on this this point. So, whatever we get this we take to be the adaptive filter weight for the next clock ok next clock. So, this is now on this line we satisfying this ok, but in the next clock the very next clock I have n plus n plus 1 s clock.

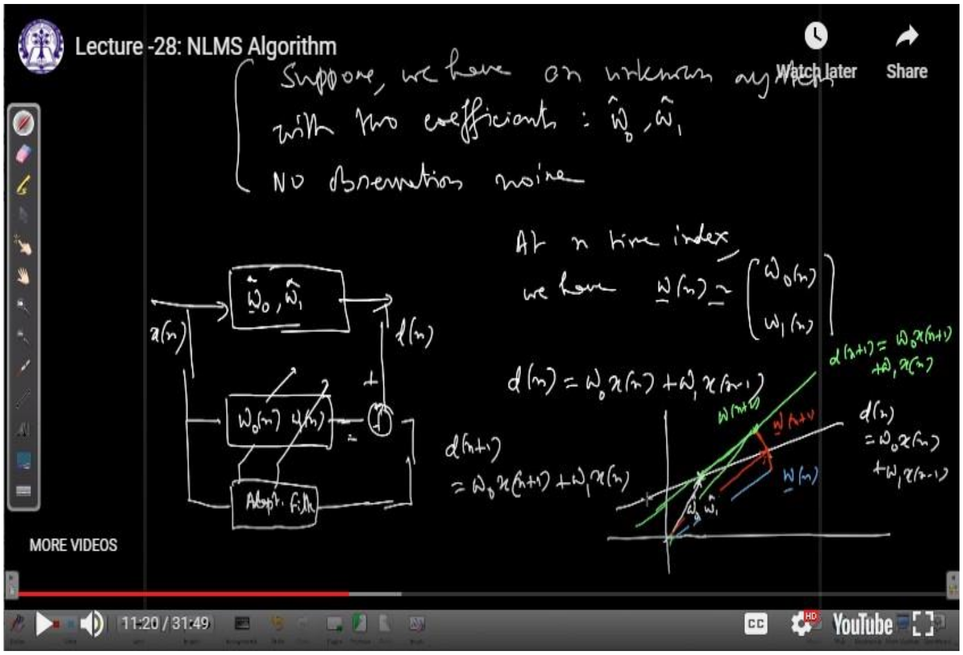
So, n plus 1 s clock we will have another equation the n plus 1 is equal to w0. So, you see x the data and here x n plus 1 it is here it was xn here it was xn minus 1 here it is xn. So, it is different straight line the straight line has changed ok. So, now, this is the model ok at n plus 1 s clock this is the equation ok. So, this equation will again be a straight line ok, but whether it is nth clock or n plus 1 s clock these equations must be satisfied by the true parameters because they are true parameters.



So, whether I am nth clock if I am nth clock w0 true value should be w0 cap w and this is w1 cap this is satisfied. When I move to next clock n plus 1 s clock again w0 should be w0 cap w1 cap and with that data I should get n plus 1 ok. This is how it is that means, this may be a new straight line, but this will again have this point this optimal point common that it will intersect this line at this point all right. So, that means, these are the line this is the new one the n plus 1 is all right. So, this was not perfectly perpendicular and so, yeah this is perpendicular looks 90 degrees here.

So, then at the n plus 1 s clock I have this new straight line which contains this optimal point, but I got w n plus 1 as the updated weight vector from the previous clock from wn I generated wn plus 1, but again wn plus 1 is not staying here not a point on this line. So, I again take a perpendicular I again take a perpendicular and this will become this will become my this point will become wn plus 2 to be used in the next cycle. Now see one thing what is the this triangle this triangle this is 90 degree right. As a result this side which is a hypotenuse is longer than this, but what is this these are the difference between wn and the optimal vector wn and optimal vector this was the difference. And what is this this is again the difference between wn plus 1 and optimal vector.

That means, this is filter weight vector at length n at time n and this is filter weight error filter sorry this is filter weight error vector where wn minus the w op. So, this is a filter weight error vector at nth clock and again what is this this is a filter weight error vector that is wn plus 1 minus w op at n plus 1th clock. And now see by this process the length ok has gone down. That means, my this point has come closer to this optimal one than the earlier because as I told you this right angle triangle this side is hypotenuse its length is longer than this length and I go on doing it. So, eventually this length error vector length will shrink and shrink and shrink and finally, it will converge on this.



This is the feeling of your normalize in a LMS algorithm all right. That means, what we are doing is this we are trying to find out and one more thing what is a perpendicular because if I I could have taken a point here I could have taken a point here I could have taken a point here, but this point is where if I draw a perpendicular this line then this is that point. What is that point? The distance between this point and this point that is minimum then only this is perpendicular. You take any other point from a triangle this again apply Pythagoras ok hypotenuse will be longer than this if it is 90 degree right ok. You this is from plane geometry only when the length is minimum then this is a perpendicular.

Any other point if I join with this point ok that will have longer length than this. Therefore, to identify this particular point which will be w n plus 1 vector I should look for a point on this straight line from which the distance between w n from which the distance with w n that is the distance has minimum length. Length means norm square in a general sense ok in an n dimensional sense norm square. That means, I should try to find out that is find out w n plus 1 so that is norm square is minimum, but w n plus 1 must must be lying on that line under the condition is w n plus 1 transpose x n x n plus 1 ok. There is an equation actually if you write I can write this also as w transpose x n all right.

So, now, I have instead of w I have brought in n because I am moving from index to index adapting so ok and this is again w transpose x n plus 1 all right this is the this is the thing. So, it is not just minimization if I just have this much obviously, answer is w n plus 1 should be equal to w n. So, this is 0, but that is not the thing I must have w n plus 1 on this straight line then which point is closest to w n. So, that the distance is distance that the norm square is minimum that will give me that w n plus 1 right. So, this is called constraint minimization.

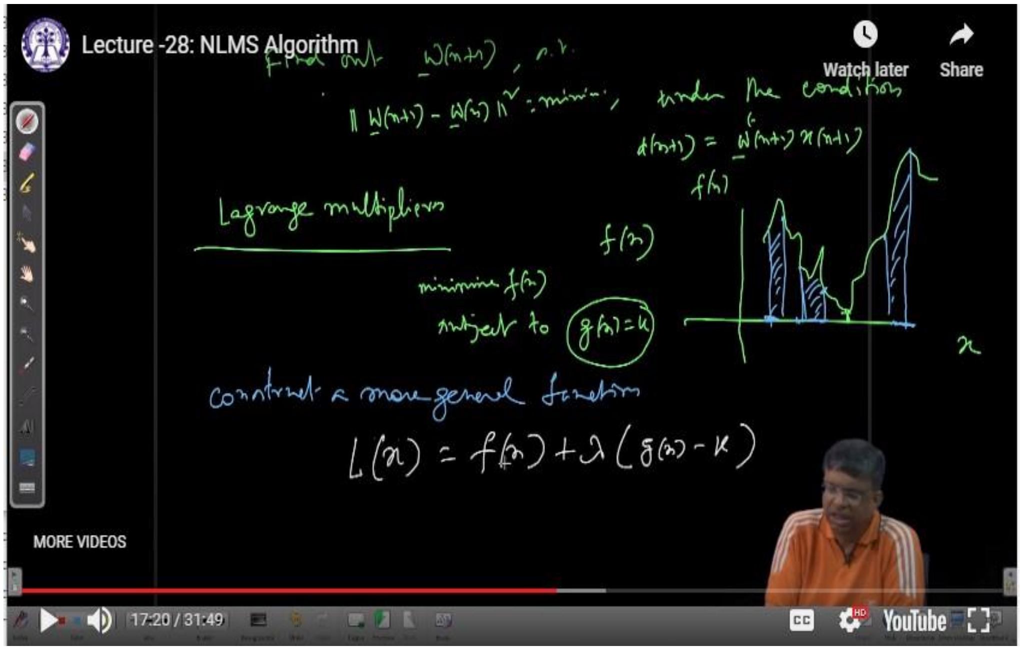
$$d(n+1) = \underline{w}^t(n+1)x(n+1)$$

So, I am not sure how many of you have studied Lagrange multipliers suppose given f of x is a function like this this is a minima you have to minimize it fine find the minima somewhere other this is a minima all right this is a global minima fine unconstraint minimization. But you are given that minimize f of x subject to the constraint some other function g x equal to say k. So, only when g x equal to k satisfied g x axis g f x only wherever this is satisfied you have to restrict yourself to that. So, only that part of this function will be concentrated there you have to find out minima. That means, it could be one patch it could be here it could be one point it could be here like that.

If these are the places where your this is satisfied g x equal to k is satisfied suppose these are the places ok. So, you take the curve only this much only this much and only this much and then find out what is the minima and this minima is this point we can see this point ok. These are constraint minima right. So, to do this to obtain this what we do we construct a more general function or L L x which is f x plus there is a parameter lambda I bring it which is called Lagrange multiplier it is a real constant times g x minus k.

$$L(x) = f(x) + \lambda(g(x) - v)$$

So, now, it is a if x is a scalar now this is a 2 dimensional function you will have one axis x another axis lambda and in that x lambda plane you can plot this.



In that plot we will become equal to this curve if lambda equal to 0 ok. So, that means, one segment of the plot is given f x, but it is a more general function then you try to minimize this ok. That means, one will be del L del x equal to 0. So, that should be f prime x plus lambda v prime x k is constant this equal to 0. So, you get one equation from here you can find out lambda from here ok one equation another is del L del lambda equal to 0.

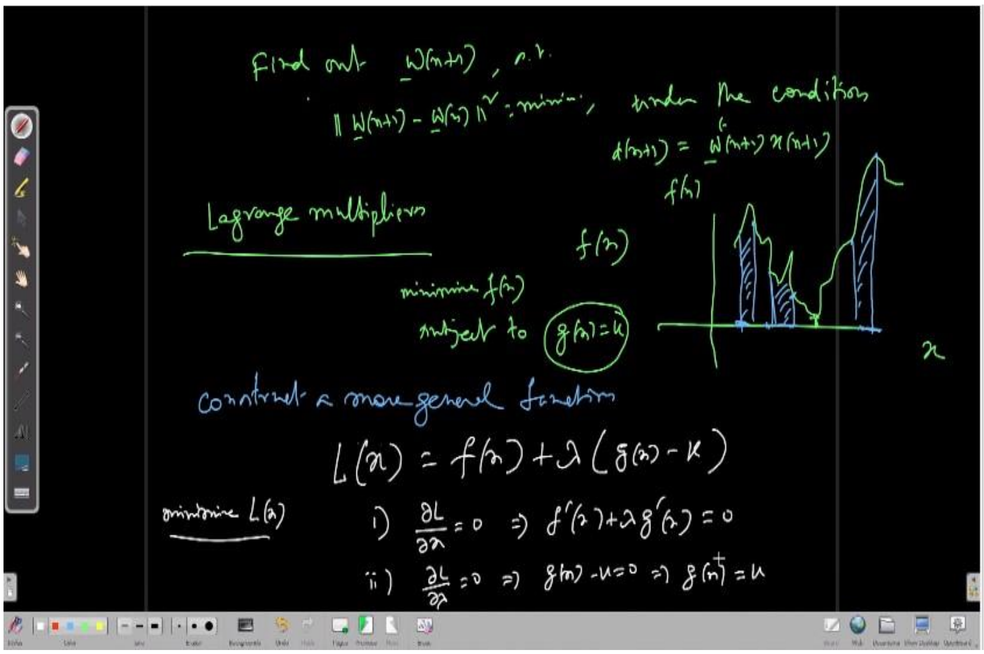
$$\frac{\partial L}{\partial x} = 0 \ \Rightarrow f'(x) + \lambda g'(x) = 0$$

So, del lambda equal to 0 means g x minus k equal to 0 which is my constraint g x equal to k.

$$\frac{\partial L}{\partial x} = 0 \ \Rightarrow g(x) - u = 0 \ \Rightarrow g(x) = u$$

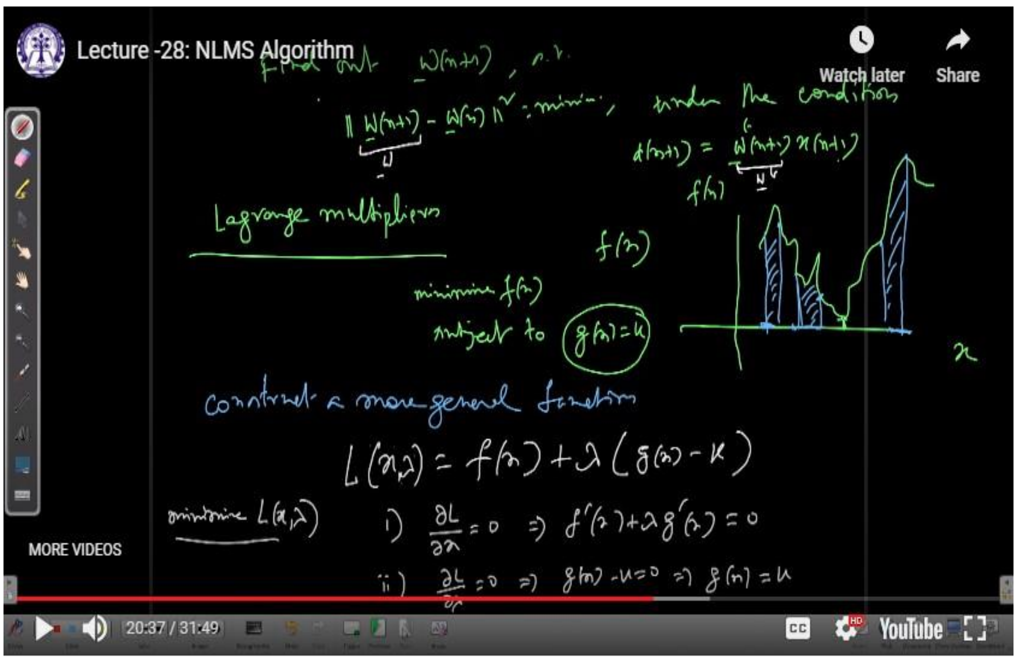
So, this will be satisfied therefore, if I minimize it that minima point will satisfy g x equal to k. And then g x equal to be subject to that I am minimizing that is g x equal to k is maintained and then I am minimizing L x. So, I will be basically minimizing f x subject to g x equal to k all right. So, both have to be satisfied simultaneously.

So, then only L x is minimizing, but if L x is minimized this point should be satisfied and under this L x minimize means f x also minimize because then g x equal to k means this is 0. So, now, this is unconstrained just simply minimize, but then I have got two parameters now. So, here I took x to be scalar, but if it could be vector of x 1 x 2 x 3 suppose. So, I have I should have del x del x 1 equal to 0 del x del x 2 equal to 0 del x del x 3 equal to 0 and of course, this all right. So, this is what I will do here I will try to find out this guy I can call it w is a variable w I have to find out this w minus this norm square.



So, that the w transpose x n plus 1 is dn plus 1 ok this is just for notational simplicity hm. That means, I will construct a more generalized function L w here w's are w's are the unknown here it was x single parameter L x we have got now w as a vector of swap mean coefficients in general say capital N coefficients. So, L will be a function of w and well I should also generalize means you know I should write it more correctly I should call it the function of 2 right hm. So, I will have a more generalized function L which will have w comma lambda this is a constraint all right. So, therefore, I take this L w w is just standing for w n plus 1 for notational simplicity I am just making it w comma lambda.

This will have the original function first which is to be minimized that was w minus w n norm square then lambda times that constraint, constraint is dn minus ok because if it is 0 it will be dn equal to w transpose x n. So, that w will be lying on that line and it will be minimized subject to that it will be minimizing this then only overall it minimized ok. Therefore, first is of course, del L del lambda which is this equation equal to 0 there is my constraint next comes del L del w. Now, we have done this kind of thing you know that is actually this is equal to del w L you remember this notation we used in the context of derivation of optimal filter and elevation algorithm and all that it is just nothing, but partial derivatives stacked ok. So, that will be that is for that I have to derive it.



Now first consider this term w minus this is w minus w n transpose now you write in an expanded form w transpose w which is this w transpose w minus w n transpose w and of course, we are taking on real I forgot to mention ok. So, w n transpose w and w transpose w n both are same and w n transpose w n so, norm square. So, if I have to derive it with respect to w what will I get this one what does it give it is like you know if w is w 0 w 1 dot dot dot dot w n minus 1 norm square of w is w transpose w which will be w 0 square w 1 square dot dot dot dot square. So, if I take it partial derivative of this with respect to any w say w k square. So, only this term will come under operation differentiation operation and it will be twice w k.

$$||\underline{w} - \underline{w}(n)||^2 = \left(\underline{w} - \underline{w}(n)\right)^t \left(w - \underline{w}\ (n)\right)$$

$$= ||w||^2 - 2\underline{w}^t\underline{w}(n) + ||w(n)||^2$$

So, then stack all the derivatives twice w 0 twice w 1 twice like that. So, it will be 2 w. So, this will give rise to 2 w all right minus 2 w transpose w n that is obvious w 0 times w 0 n plus w 1 times ok that is w n I am sorry I should have written it not this way. So, w transpose w n you can easily see w 0 w 0 n w 1 w 1 n dot dot dot and the direction. If I differentiate it will basically kth term will be giving me w k n.

So, if I stack them w 0 n w n. So, basically you get w n. So, from here you get w n all right and this is of course, independent of w. So, that goes and this we have to equate to 0 that is if I apply del w only on this much ok this part, but this is I have got one more term this is except for I mean integration to w minus w n norm square here you see there is another term. So, this I will not be this just to show that if I was doing unconstraint optimization, I would have got this and w would have been w n which is what I had told earlier that is if there is no constraint your solution is obviously, w equal to w n because then the error w minus w n is 0, but that we are not doing right. So, this is this we are not doing we are minimizing this whole thing.

So, first term when derived will give rise to this there is now del w L will be twice w minus twice w n and then I have got another term minus lambda w transpose x n. So, w transpose x n again derived when derived by the same way in a same logic it will be giving rise to x n. So, minus lambda x n and that is equal to 0. So, I take lambda x n on one side hm and this w minus w n is lambda by 2 x n. Now, this is a vector equation w is a vector w is a vector x n is a vector, but from this I have to find out lambda.

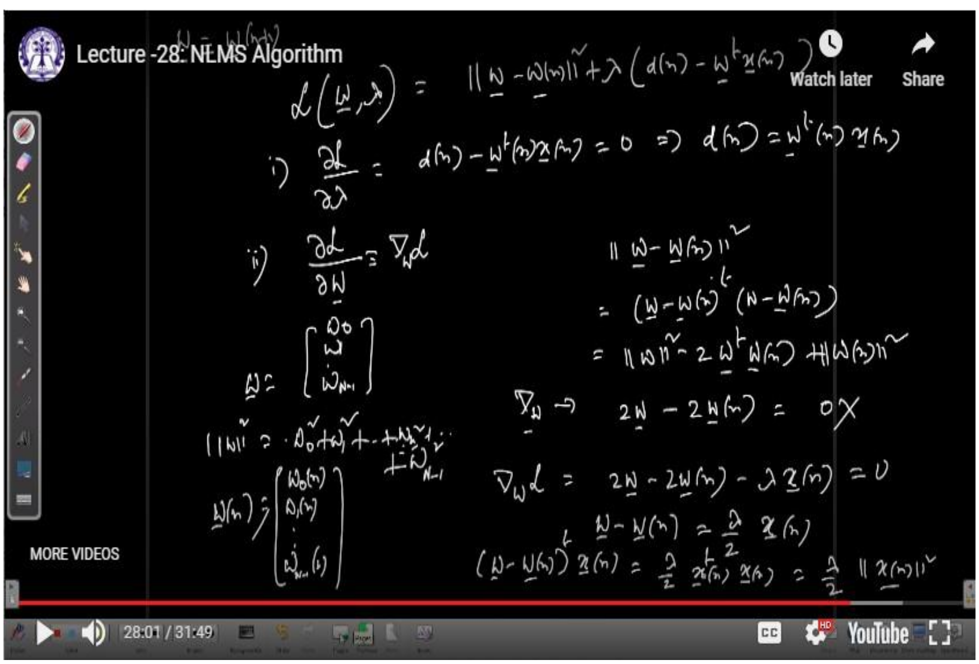
$$\nabla_w L = 2\underline{w} - 2\underline{w}(n) - \lambda \underline{x}(n) = 0$$

$$\underline{w} - \underline{w}(n) = \frac{\lambda}{2} \underline{x}(n)$$

So, what I do I take transpose of this transpose. So, this also to be transpose times x n here I will say lambda by 2 this is to be transposed x n which is lambda by 2 norm x n square.

$$\left(\underline{w} - \underline{w}(n)\right)^t \underline{x}(n) = \frac{\lambda}{2}\, \underline{x}^t(n)\underline{x}(n) = \frac{\lambda}{2} \left\|\underline{x}(n)\right\|^2$$

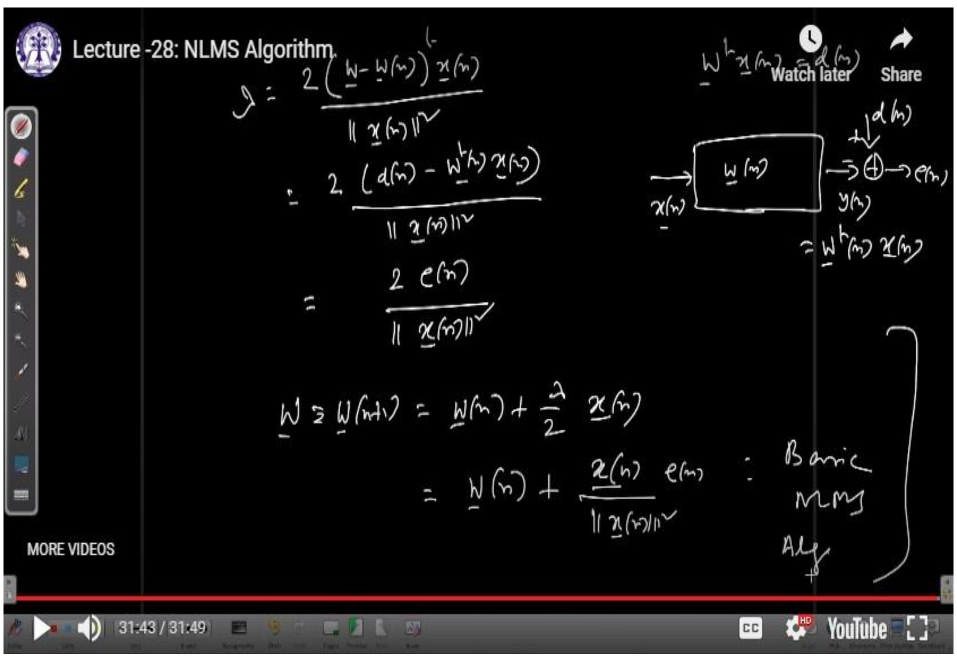
So, I get lambda from here I get lambda from here lambda is 1 minute lambda is 2 into 2 into w minus w n transpose x n by norm x n square. Now, one thing you see from this constraint when you derive with respect to lambda the same thing you get dn equal to w transpose x n that means, w transpose x n equal to dn that constraint. That means, w transpose x n is dn right that means, this is equal to twice dn minus w transpose n x n by norm x n square.



Now, what is happening is the filter x n and I have got this w n. So, output is right this we know w transpose n times input vector. So, this thing if I subtract as I am doing here dn minus d is filter output this is my good old filter output error which is used for adaptation I am calling it en. So, it is twice en by norm x n square this is lambda. Therefore, if I bring it back here use that lambda here then what I get ok w I keep on to one side ok.

So, w and wn and this term on the other side. So, w will be wn plus lambda by 2 x n right. So, w which is equal to actually wn plus 1 I just use the notation w here it will be w n ok this is equal to 0. So, wn plus lambda by 2 x n and now lambda if you substitute here is 2 and 2 cancels en by this n is a scalar. So, x n ok lambda I am replacing.

So, 2 and 2 cancels en x n n is a scalar.



So, en I am writing later does not matter whether I write by norm x n square this is the basic NLMS algorithm then we will work further on it at body phi and you know interpret that is in the next class. Thank you very much.