Introduction To Adaptive Signal Processing

Prof. Mrityunjoy Chakraborty

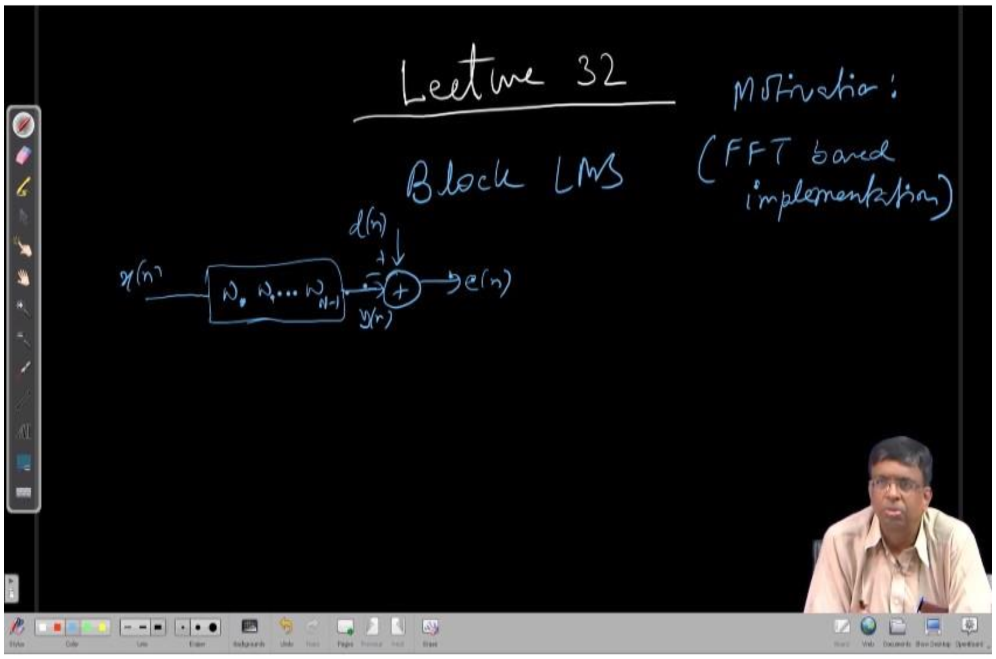
Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology, Kharagpur

Lecture No # 32

Introduction to RLS Algorithm

So, today we cover another interesting version of the LMS algorithm called block LMS. It is particularly useful for very efficient practical implementation using DFT ok. DFT based DFT means actually FFT, I would say FFT because when you implement DFT you use the first algorithm FFT. A FFT based implementation that is the motivation. As before the structure remains same that is you got a filter w0, w1 dot dot dot w capital N minus 1. This is capital N coefficients, this is xn, these coefficients have to vary so that output if it is desired response and I am dealing with real case here what may not be required ok general case is en.



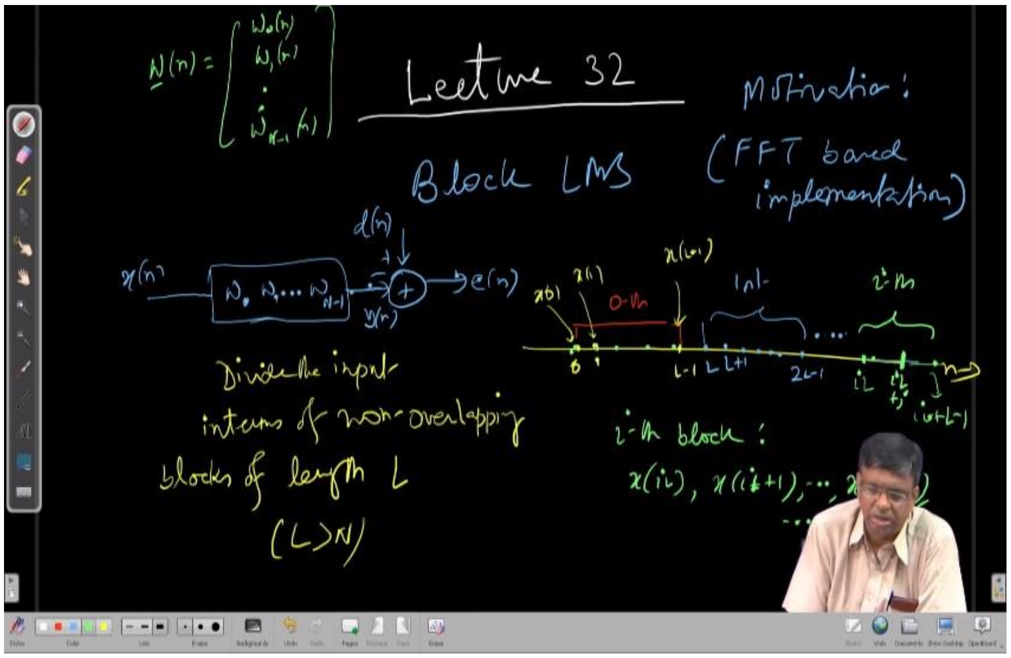
So, we have seen in the case of additive filter that you adopt this coefficient may be LMS formula. So, that progressively the vary I mean this error variance goes down to the minimum I mean ideally actually this filter coefficients we update so that ideally, they should go to the optimal ones, but in the case of LMS you know they do not converge absolutely to the optimal ones they dance around fluctuate around the optimal ones. That is why this variance of this goes down, but does not go down to the minimum possible epsilon square mean, but there is an additional component called xs min square error. Nevertheless, there was a setup that time there is a setup this time also, but here there is slight difference here.

So, here remember we have capital N coefficients. So, here this is your data. So, maybe we choose block we divide the input in terms of non-overlapping of length I will explain what I mean L and L should be greater than this N that is this is your x0 0th position you have got x0 here this time x is x0 here then 1 dot dot dot up to L minus 1 you have got all the data x of 0 here you have got x of 1 here you have got x L minus 1 and dot dot. So, then I put consider to be 1 block under 1 block and call it 0th block then I start with the next L L plus 1 dot dot dot dot dot dot to L minus 1. So, again another block of length L I call it first block dot dot dot dot dot in general Ith block Ith block will be what see 0th block starts with 0 then 1 up to L minus 1 first block starts with L that is 1 into L goes up to 2 L minus 1 second block would be the next index 2 L.

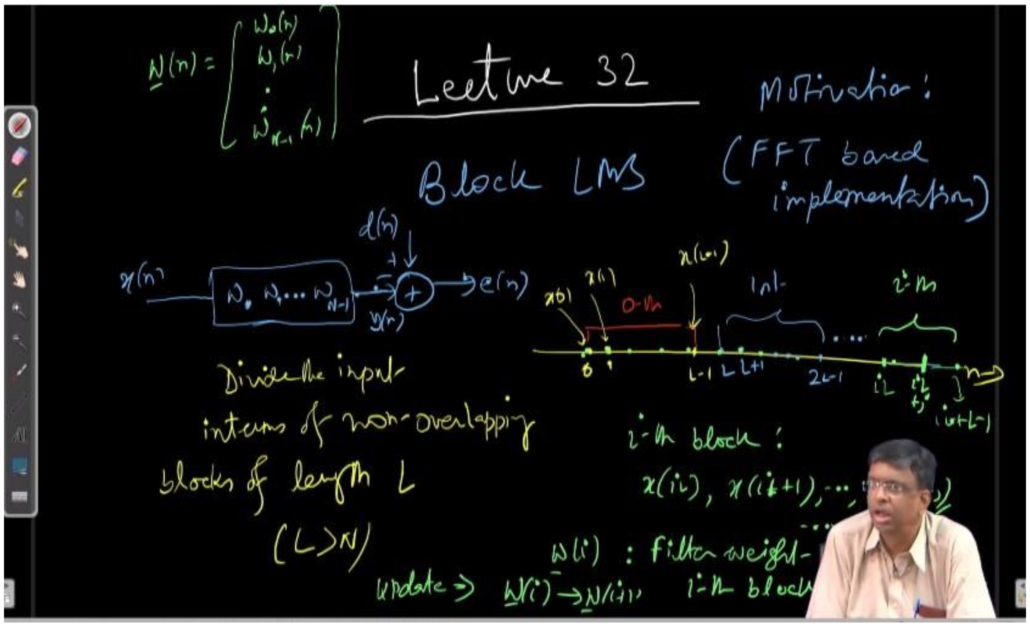
So, second block means 2 L that is 0th block 0 first block 1 into L second block 2 into L dot dot dot. So, Ith block will start at IL in general and go up to this IL means IL plus 0. So, it should be IL plus L minus 1 because total length is L. So, IL IL plus 1 dot dot dot dot dot up to IL plus L minus 1 that is my Ith block that is Ith block in general means input x IL general term may be here IL plus G dot dot dot x IL plus L minus 1. Now what we do in the case of LMS I was updating this weight vector that time it was function of N W0 N W1 N dot dot dot.

$$\underline{w}(n) = \begin{bmatrix} w_0(n) \\ w_1(n) \\ \vdots \\ w_{N-1}(n) \end{bmatrix}$$

So, this was updating from Nth clock to very next clock means N plus 1 N then very next clock N plus 2 N dot dot dot here I do not do like that. Here for every block once I know the weight vector I do not change. So, I use it here also here also here also here also here also at all these points I go on using the same weight vector for filtering the data at this point this point this point like that. Then I move to the next block then again I update and again once I update I hold it fixed it is at the block. So, use that same again to filter the input at this point this point this point this point all these points dot dot again once I reach Ith block I do not change it I use it to filter the input at ILth point IL plus 1th point dot dot dot IL plus L minus 1th point.



So, filter coefficients are updated only once per block and then held fixed within the block for doing the filtering operation ok. So, we will have Wi this is the filter weight vector for the Ith block and we will update update means Wi to next block. So, once I have Wi I will be using it at all the points within the Ith block all right. That means I draw that line again this was 0th block I am writing the this is the first block Ith block Ith block I had this is repetition of previous page IL plus 1 dot dot dot IL plus J and last point is IL plus L minus 1. So, at every point so for n time index n either here or here or here or n equal to any index here IL plus J, J could be 0 I am here J could be 1 I am here.


Lecture 32
Block LMS
Motivation: (FFT based implementation)
$\underline{W}(n) = \begin{bmatrix} w_0(n) \\ w_1(n) \\ \vdots \\ w_{N-1}(n) \end{bmatrix}$
$x(n)$
$w_0\ w_1 \dots w_{N-1}$
$d(n)$
$y(n)$
$e(n)$
Divide the input in terms of non-overlapping blocks of length L
$(L > N)$
$x(0)$ $x(1)$ $x(L-1)$
0-th
1st
i-th
0 $L-1$ L $L+1$ $2L-1$ iL $(iL+L-1)$ n
i-th block:
$x(iL), x(iL+1), \dots$
$\underline{W}(i)$: filter weight
i-th block
update ⇒ $\underline{W}(i) \rightarrow \underline{W}(i+1)$

So, J could be 0 1 dot dot dot dot L minus 1 if J is 0 I am here if J is 1 I am here if J is L minus 1 I am here. So, at any of this index n filter output yn will be W not W transpose or W if it is real I write W transpose for our convenience let us assume real. So, W transpose not n or if you assume complex does not matter in this if it were complex I had put WH remember WH, but it is not n it is not that filter coefficients change from index to index index to index here for all the n indices n IL plus 0 IL plus 1 IL plus L minus 1 these are the indices they are the values intake is not that W filter weight vector changes what this indices know as long as they are in the same block indices indices are in the same block and that is what I am doing I am taking n from here filter coefficient vector remains same and that is Wi the same one is used for doing this filtering Wi H xi alright Wi H xi if you are not very comfortable about complex here when we start with real and then I will convert I think that is better I will convert it to complex. So, because here it will be little more complicated in terms of expression and all so maybe take real. So, just W transpose i xi which is also equivalent to x transpose i is a n it is not i it is n I am very sorry this is n because xn here xn here you know n is the real index actually time index.

$$For\ n = iL + j, j = 0,1, \ldots, L-1$$

$$y(n) = \underline{w}^t(i)\underline{x}(n) \geq \underline{x}^t(n)\underline{w}(i)$$

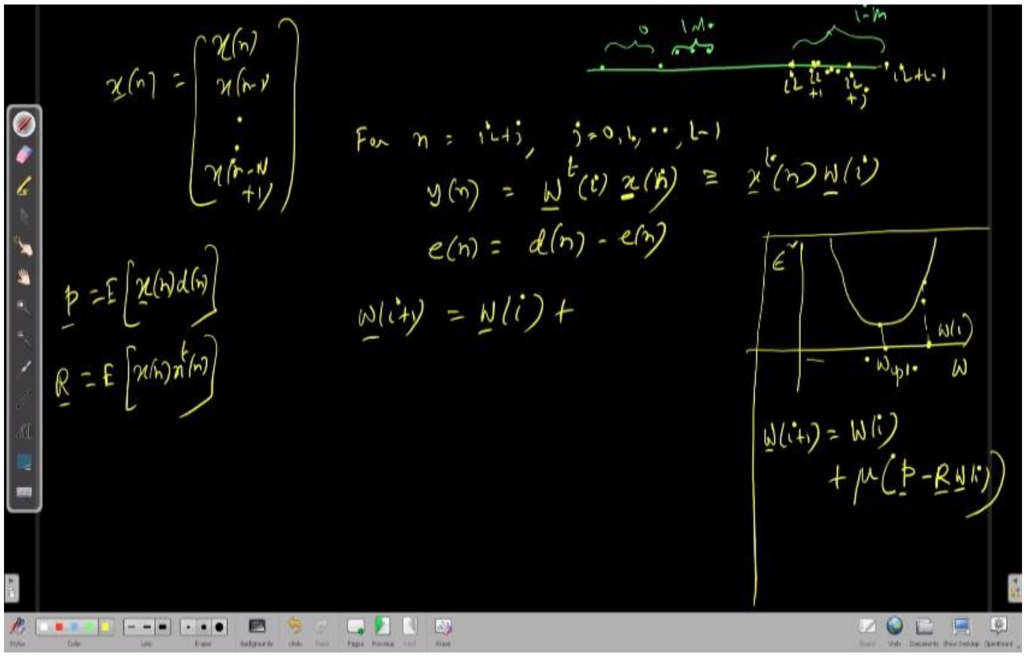
So, that anything index I have got the input data vector xn vector where xn as we all know x of n but filter coefficient is fixed once I enter i in the block it is fixed Wi same one I am using for doing the filtering these are filtering operation. So, n can be here n can be here n can be here but same coefficient here and a transpose p same as b transpose s you can write the same thing as x transpose n Wi in the real case let us consider real case first maybe it will be easy to extend the incompressibility and we also do error output error calculation as before bn minus en, but filter coefficient.

$$e(n) = d(n) - e(n)$$

So, this I go on doing at all these points I do not at these points I do not change the filter coefficient vector this remains Wi and then you can write in a program like manner anyway. So, once this is done then you do Wi plus 1 in terms of Wi plus some updater that is what we will see what it would be. You remember this is a side story otherwise go back to my previous lectures and see when I did steepest descent while arriving at the LMS equation from the optimal filter you remember I covered the steepest descent without I just to start with I took only one coefficient and output if you know expected value of the power expected value of E of that is expected value of small e square n e square n expected value with the output variance that as a function of a single coefficient filter I took a single coefficient filter first there is a quadratic function easy to plot it was something like this.

So, this is the where it is minimal w opt to start your iteration may be at any Wi find the gradient here then from Wi subtract a quantity proportional to the gradient that is minus mu by 2 mu by 2 is a coefficient is a proportionality constant mu was called the step size that times the gradient here you subtract from W. So, gradient is positive we go to this direction there is a approach this if you are here gradient is positive gradient is negative. So, negative positive so you go to this side. So, you go around this and finally, here that was the philosophy no point in repeating that, but just reminding you and in case you have forgotten all that you can quickly go back to those lectures and see and then by the steepest descent method we derived this update equation after working out the gradient expression general gradient expression equals like this Wi plus mu i th iteration to i plus 1 th iteration mu p, p was the cross correlation vector E of any xn into dn I am not assuming complex. So, no point in putting star and all that also R was auto correlation.

So, it was R Wi this I did that time. So, if I bring that here and then derive try to derive LMS this will be i th iteration. So, in the i th block I have got Wi fine I am updating in the next when I move to the next block that is when time index n crosses this point enters here for the next block that time from Wi I should go to Wi plus 1.



$$\underline{x}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \end{bmatrix}$$

$$\underline{p} = E\left[\underline{x}(n)d(n)\right]$$

$$\underline{R} = E\left[\underline{x}(n)\underline{x}^t(n)\right]$$

For $n = iL+j,\ j = 0, 1, \dots, L-1$

$$y(n) = \underline{W}^t(i)\,\underline{x}(n) = \underline{x}^t(n)\underline{W}(i)$$

$$e(n) = d(n) - e(n)$$

$$\underline{W}(i+1) = \underline{W}(i) +$$

$$\underline{W}(i+1) = \underline{W}(i) + \mu\left(\underline{p} - \underline{R}\,\underline{W}(i)\right)$$

So, I should ideally follow this, but as I told you we will be replacing small p because p is not known to us we have to approximate it by some data value data driven approximation I have to do for p and R in the case of LMS we need a very bad approximation that time p was replaced by xn into dn because xn into dn know many such product and averaging and R was replaced by xn x transpose n this time I will have a better average I will show you I will have better average because once I entered here I have got data here here here here so I will use all of them p will be replaced by what the xn vector at this point into dn then xn vector at this point into dn xn vector at this point into dn this I will go on over this and so how many times how many cases arise 1 2 3 up to L so I divide that summation by L that is a good average right so p will be like this take the general index iL plus j okay at that point data vector times the desired response d the same index iL plus j and so this I will go on doing at j equal to 0, j equal to 1, j equal to j, j equal to L minus 1 all of them so I take a product and sum so I am getting a better average now j equal to L minus 1 so total L such products I add so I get a resulting thing I divide by L so very good average better average than L is this okay.

$$\underline{P} = \frac{1}{L}\sum_{j=0}^{L-1} \underline{x}(iL+j)d(iL+j)$$

Similarly R that time it was just xn x transpose n but now I have the luxury of not updating over this entire block so I can use data here for the same calculation from w i to w i plus 1 I can use all the data here so that is what I am doing take x iL plus j and transpose so at this point the data vector into its transpose so they will do it here here here every time you get a matrix column vector that is matrix so matrix here matrix here matrix here matrix here all of them you add and divide by L.

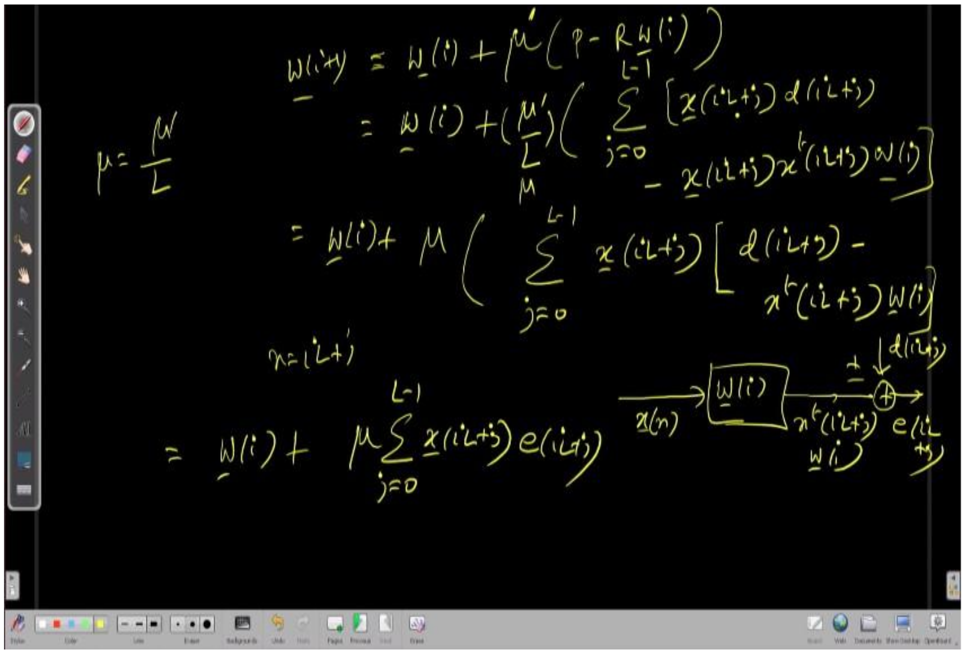
$$\underline{R} = \frac{1}{L}\sum_{j=0}^{L-1} \underline{x}(iL+j)x^{t}(iL+j)$$

So this is a better estimate than LMS so this is what I will be using here okay this is what I will be using now you can see one thing at one hand I am not updating by filtered coefficient vector at all the time clocks at all the indices after I update I am entering the block I hold it fixed I do not change then again after the block is over then again I change again hold it for this block then once this block is over then again change and like that all right. so that way we can say that I am becoming slow because I am not changing myself fast I am not updating fast so it can give you the idea that I am making it slow that may be true but on the other hand unlike LMS I am using better estimates of p and r, as a result my algorithm is a will then converge better converge faster so one negative effect takes gets compensated by another positive effect so in terms of speed there you know excess mean square error and all that we do not lose anything by some is elements these are all at par all right because here I am searching block to block okay I am updating over block to block so I am not moving that fast not every clock I am updating but at the same time I am using much better estimate of p and r so one you know takes care of the other.

So if I now replace them here in this okay let us do some mental maths p if you replace by this where r you replace by this 1 by L you take common summation this minus summation this into Wi so Wi is fixed in the summation summation is over j okay so this you can write as you can take a summation common both are over j from L to L L minus 1 j to L minus 1 so I can take a common summation or maybe I move to the next page because I am not having space here and I do not want to delete anything here let it be there so I move to the next page so let me erase this part so W this was the equation block update not time update block update updating from block to block ith block to i plus 1 it once ith block is over all the data generated you know all the data used up during that time that available I calculate this extra part and what was that there was mu p minus r Wi so mu p minus r Wi and now I replace you remember for your sake you can say p is 1 by L and this is 1 by L p and r from both 1 by L will come out same summation j equal to 0 to L minus 1 j equal to 0 to L minus 1 that can be common let me call it mu prime instead of mu so mu prime by L I use the same summation j equal to 0 to L minus 1 and that was for i th block i is fixed so it was i L plus j times d i L plus j that is from one term x i L plus j d L minus this into this x i L plus j into this transpose times Wi all right this mu by L this together I call mu so mu is mu prime by L for every j you carry out this summation this you know vector times i L plus j so pick up a j from here for that you take this vector multiply every element by this scalar similarly this vector times its transpose matrix that times this constant vector this is a constant because i is fixed so this fixed.

So here I can take you see this element and this element are same these two vectors I can take them outside this square bracket I am left with so what is this at i L plus j th index suppose N is i L plus j so this is what you got a filter Wi for the i th block at n th clock you got x, N is this particular index i L plus j so that time I got this input vectors x N which is x i L plus j I am writing as x N so output is this transpose this or this transpose this they are real so x transpose N is i L plus j times Wi this is my filter output and if I subtract from at that n th index this N so what I have here is the i L plus j so it is the error at that index e i L plus j that means update equation e i L plus j.

You can see one thing if L is 1 so block is same as only 1 block has only 1 index they did go to that goes back to the original LMS because block size 1 means I mean L is 1 so 0 to 0 so just L is 1 so I have only x i L look L is 1 so x i that is your N so N is i again e i mu times that so Wi plus summation has only one term because L is 1 so 0 to 0 so only one term is mu into only one term L is 1 j is only 0 so mu into x i into 1 i if i is the index mu x i again e i that is LMS so like LMS here also you can prove convergence you know using independence assumption and all those things with absolutely similar manner you can derive in the upper bound on mu and all those things for convergence that I am not doing it now because that is quite easy.



$$\underline{w}(i+1) = \underline{w}(i) + \mu'\left(p - R\underline{w}(i)\right)$$

$$= \underline{w}(i) + \left(\frac{\mu'}{L}\right)\left(\sum_{j=0}^{L-1}\left[\underline{x}(iL+j)\,d(iL+j) - \underline{x}(iL+j)\,x^t(iL+j)\,\underline{w}(i)\right]\right)$$

$$\mu = \frac{\mu'}{L}$$

$$= \underline{w}(i) + \mu\left(\sum_{j=0}^{L-1}\underline{x}(iL+j)\left[d(iL+j) - x^t(iL+j)\,\underline{w}(i)\right]\right)$$

$$n = iL+j$$

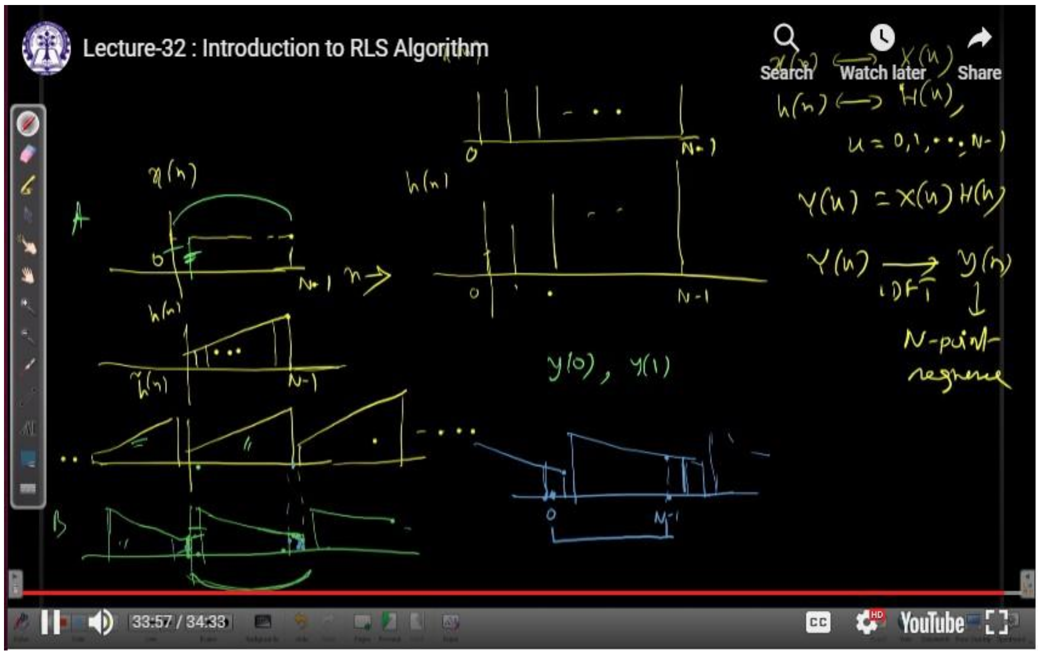
$$= \underline{w}(i) + \mu\sum_{j=0}^{L-1}\underline{x}(iL+j)\,e(iL+j)$$

And if you are interested you can see books okay absolutely similar steps this is given there so I would rather be trying to because of my lecture number of lectures limited number of lectures so I would rather try to show you a good DFT based implementation of this so that I will do in the next class but before that one thing I want to do actually these are these are the things that I have you know I mean what I am going to say you are supposed to be knowing it through your DSP like some you know I mean elementary things from DFT circular convolution and all that but in case you have forgotten so suppose I have a sequence x n suppose I have x n and another sequence h n or maybe let me draw a bit a better picture maybe x n is as a special case take suppose all 1 or constant x n 0 to n minus 1 and h n is like you know going up something like this but both are capital n point sequences so x n suppose you take its capital n point DFT capital XK h n you take capital n point DFT K hk so K will be 0 1 up to n minus 1.

Then you form you multiply that to xk, xk into hk and then yk inverse i DFT if you do inverse DFT you get another sequence yn that yn is called the circular convolution between xn and nk all those details I will not explain but the way to calculate circular convolution because yn also will be a capital n point n point sequence the same range so to calculate I mean yn what we do is this let me redraw again xn is constant so it is like this you know an envelope let it be as it is hn is like this maybe let us draw these points I am just drawing like an envelope you have to do circular convolution that is between these two what is the what we do is this we hold one as as it is write it as a function of small n as it has given no problem other one other one other one I make a periodic version and then so I make a periodic version so I repeat dot dot dot dot dot dot dot dot I make a periodic version you can call it h tilde n this guy is just repeated then we flip it ok flip it across this y axis so then you get these three bits as it is this guy and then this part goes to that side so this this fellow comes here and this fellow goes here this here till one more term because total length is capital n from here to here length is capital n but this is starting here so it has to go to this and again like this.

Then what we do you chop off the left side and right side just focus on this part 0 to n minus 1 and then if you call it a if you call it B over this part this by range this by zone you do sample wise multiplication this guy by this guy this guy by this guy double lines are double dot dot dot and this by this and add so that will give you y 0 that is with 0 shift y 0 then next to find out y 1 what will you do you take this periodic thing give a shift by 1 to the right then again use a chopper chop off the left side of the y axis a right side up to the point capital n minus 1 so what do you have is this this will go further to the right this by 0 this will go further to the right next guy will come here so it will be like this so this fellow had already gone out this fellow has gone out this will go further into this block this will also go out so my n minus 1 then one fellow another fellow and then again I use a chopper so I take from here 0 to the n minus 1 only this much so whatever I see from here to here this and this samples there will be sample wise multiplication and add that will give me y 1 and I go on doing it then I give a shift again to the right then use a chopper chop off the left chop off the right left of 0 right of capital n minus 1 so you get only one part.

Like if you chop off this goes this goes and then do term wise multiplication all with this other one which I kept fixed and add and go on doing it this is the way to do circular convolution so if you do DFT DFT multiply the two DFTs and inverse DFT what you get is circular convolution between xn and hn and this is how it is.



I would advise you please come back in the next class after studying little bit of this DFT circular convolution some properties of circular convolution because these are, you know if this course is not supposed to be covering them okay I will still I will still derive few one or two basic results which I need you know but it will be much better if you come prepared with this in the next class yeah thank you very much.