Introduction To Adaptive Signal Processing

Prof. Mrityunjoy Chakraborty

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology, Kharagpur

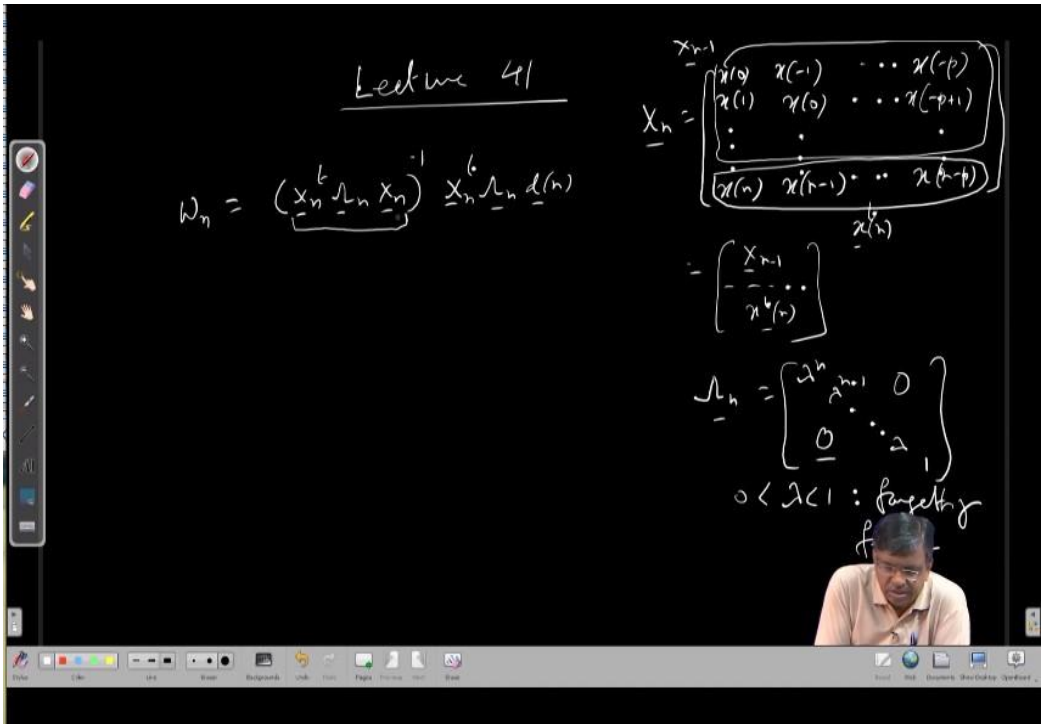
Lecture No # 41

Derivation of the RLS transversal adaptive filter

In this class, we will be concluding the recursive least squares adaptive filter RLS that is recursively least squares adaptive filter. We have already done the derivation discussed at length just a few issues couple of issues are left one is that if you recall we had the data matrix at nth point of time. So, filter has p plus 1 coefficient that is a p plus 1 column. So, this vector we wrote as x n transpose and this part was just you can you have seen x n minus 1 this part. So, this is basically x n minus 1 and x transpose n and then we found out that least squares estimate of the filter coefficient vector at nth clock that is using data from n equal to 0 from time from clock from 0 index 0 to index n that was this x n transpose lambda n x n inverse x n transpose lambda n d n d n is the real response vector and lambda n was this where lambda forgetting factor.

$$w_n = \left(\underline{x}_n^t \underline{\Lambda}_n \underline{X}_n\right)^{-1} \underline{X}_n^t \underline{\Lambda}_n \underline{d}(n)$$

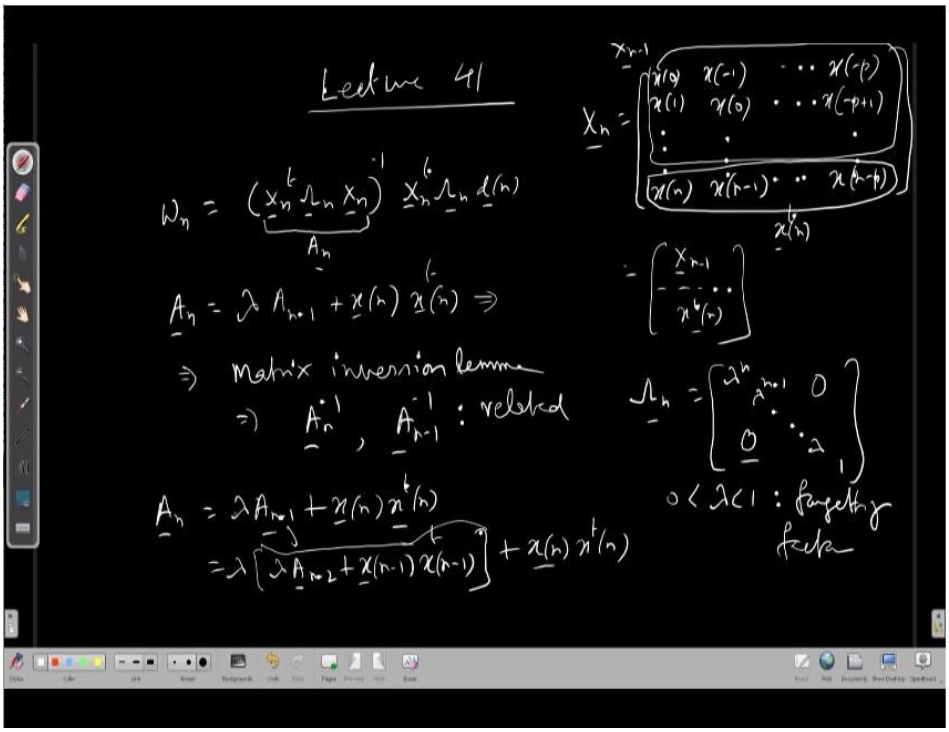
And here we assumed that this matrix we call it a n this is invertible and that is what the problem is invertible and under that assumption in fact we related these are all results which have been derived in the previous lectures.



So, we will not reroute them only recall them then a n was related to same matrix, but at n minus 1 is clock like this. And then assuming that both a n and a n minus 1 are invertible we use the matrix inversion lemma. So, that a n in so that we use matrix inversion lemma. So, that a n inverse and a n minus 1 inverse they get related through some expression that is we express a n inverse in terms of a n minus 1 inverse and then proceed derive something called gain vector and all those things algorithm is derived.

But this assumption that they are invertible that has to be enforced because you can see as we discussed earlier here number of columns is p plus 1 number of rows as such is n plus 1. If n is less than p then you have number of rows less than number of column and you have seen earlier you have seen that in such cases this matrix which is Hermitian a n is Hermitian, but it is not positive definite and therefore, it is not invertible that we have seen at length you know. And this will happen because n has to start from 0 then 1 then 2 then 3 then only it will go. So, in the initial stages number of rows will be less than number of columns and so this formulation will not work. So, what to do then how to make sure that the invertibility issue does not bother us.

So, for that what we do consider this expression. Here we again write a n minus 1. So, we write like this first write what we have written in there then this we replace again by the same expression replacing n by n minus 1 in the left-hand side. So, then we get a n minus 2 ok. So, this becomes this entire thing a n minus 1 by the same formula if you replace a n by a n minus 1 in the left-hand side you get lambda a n minus 2 plus x n minus 1 x n minus 1 transpose that is what we replace this lambda is this lambda and good old x n x transpose n.



Again, you replace a n minus 2 by the formula it will be lambda a n minus 3 plus x n minus 2 x n minus 2 transpose replace that here and go on doing it. So, if you do that you will finally, get lambda to the power n x 0 x 0 transpose ok. Let me write it in the next page. It will be like this lambda to the power. So, I go back by n plus 1 steps.

So, it will be a minus 1 see this when I have lambda to the power 1 a minus 1 when lambda

to the power 2 a n minus 2. So, we have a 0 lambda to the power n when I have a minus 1 lambda to the power n plus 1 ok that is why lambda to the power n plus 1 a minus 1 and then you start like this ok. Now, this is 0 because our business starts from n equal to 0 previous to that whatever that I have that all that is 0. So, this is 0. So, actually we have this.

$$\lambda^{n+1}\underline{A}_{-1} + \lambda^{n}\underline{x}(0)\underline{x}^{t}(0) + \cdots + \lambda\underline{x}(n-1)x(n-1) + x(n)x^{t}(n)$$

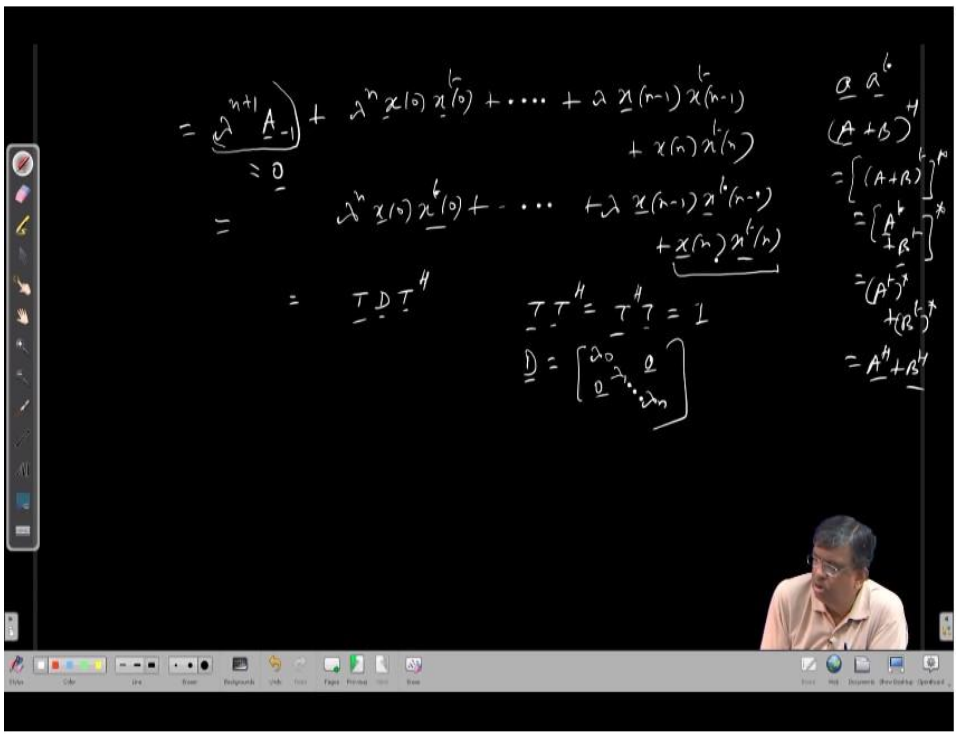
$$\lambda^{n}\underline{x}(0)\underline{x}^{t}(0) + \cdots + \lambda\underline{x}(n-1)\underline{x}^{t}(n-1) + \underline{x}(n)\underline{x}^{t}(n)$$

Now, this is a Hermitian matrix we have already seen a n is a Hermitian matrix. Now, here you can see this is Hermitian because any vector column vector into its transpose its Hermitian. I mean if you take the transpose here all are real. So, no conjugation. So, only ordinary transpose is same as Hermitian transpose.

So, if you take the transpose of this you get back a a transpose. So, this is Hermitian this is Hermitian this is Hermitian and summation of Hermitian matrices is Hermitian. For instance if a and b are Hermitian then a plus b also Hermitian because we know a plus b h means what plus a plus b transpose then conjugate or vice versa. When you sum the two matrix and then transpose you will get the same thing if you first take transpose of both and then add and then start. So, if you add any i comma jth element of this a transpose with the i comma jth element of b transpose and then conjugate it is same as if you first conjugate the i comma jth element of a transpose and i comma jth element of b transpose and then add.

So, it will be a transpose star plus b transpose star which means a a plus b a very silly thing very simple thing. So, now this every component is a Hermitian matrix this is Hermitian all right. Since this is Hermitian we can write it in this form TDth where T is unitary Tth is thT is identity what is this it consists of mutually orthonormal Eigen vectors of this total Hermitian matrix place side by side and b diagonal matrix consisting of the Eigen values of this Hermitian matrix all are real. Now, since in the early stage of the algorithm as I said

this is not positive definite and therefore, you know it is not invertible that means, determinant is 0 then only a determinant of this matrix is nothing, but product of the Eigen values we have seen earlier in the early part of this lecture series ok. Determinant with the entire thing is nothing, but determinant of this diagonal matrix which is product of the Eigen values.



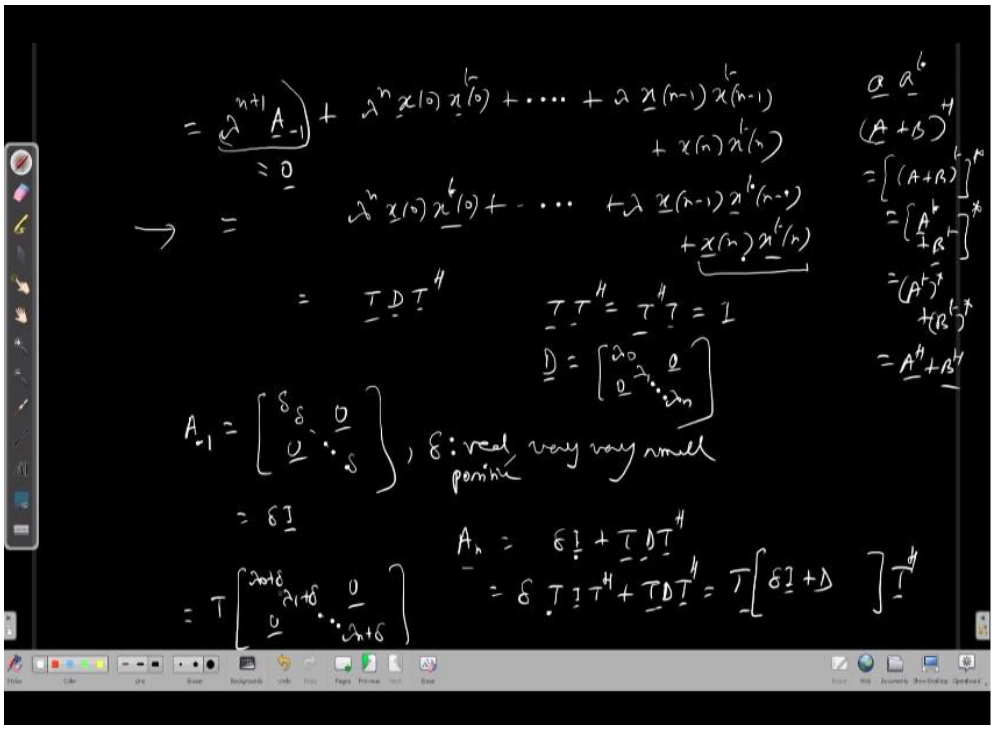
So, if the matrix is not invertible in the early stage of the algorithm that at least one Eigen value should be 0 then only determinant is 0 and that is what we will try to avoid. So, what we will do we will take this summation and add with this as we take this initial value for this a minus 1 ok. What a diagonal matrix where delta is real and very very small positive very very small. So, then that is we can write as delta i. So, therefore, a n will not be just this it will be delta i plus this t d t h whatever I got here and then i I can rise as t because this is unitary.

$$\underline{A}_n = \delta \underline{I} + \underline{T}\,\underline{D}\,\underline{T}^H$$

$$= \delta \underline{T}\,\underline{I}\,T^H + TDT^H$$

$$= T\left[\delta \underline{I} + D\right]T^H$$

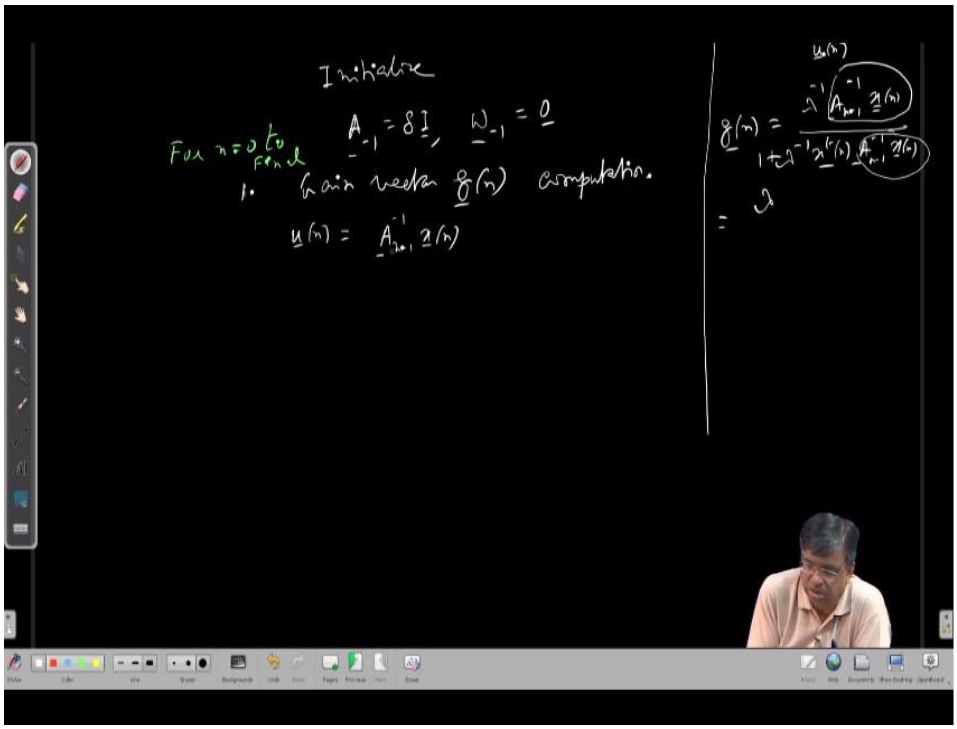
So, t identity t h identity into t h is t h means identity matrix and t t h is i. So, delta i and t d t h as before. So, take t common and t h common delta i plus d which means I have got lambda 0 plus delta lambda 1 plus delta dot dot dot lambda n plus delta. So, all the Eigen values this is a positive semi definite matrix. So, they are either 0 or positive, but since it is not invertible at least there is one 0, but now everybody getting added by some small positive number.



So, now, nobody is 0 therefore, determinant will be positive and it is invertible. So, this is that initial value instead of ideally 0 I take it to be delta i, but delta is very very small. So, it does not create you know any error much error in the program in the algorithm because ideally this would have been 0, but still I keep it as a positive constant delta time i and so this matrix invertibility issue is resolved. This is one issue I have to discuss and now I write the algorithm. One thing you remember by the way a n min or maybe I will do it little later.

Let me formulate the algorithm initialize a minus 1 as delta i as I said I need w minus 1 also in that prior error calculation take it to be 0. Then first step is gain vector g n computation ok. Now what was g n? g n was lambda inverse let me check x n divided by 1 plus lambda inverse x transpose a n ok which is nothing, but lambda and I call this to be suppose this a n minus 1 inverse is given to me ok like a minus 1 is given. So, a minus 1 inverse is nothing, but 1 by delta times i. So, this is given to me ok.

Let me call it so this is assume this to be given let me call it u n. So, I will calculate it once because here in the denominator also this factor comes. So, why calculate twice and use waste computational resource. So, calculate is only once for the numerator and use it in computing the denominator and also lambda inverse lambda inverse if you multiply numerator and denominator by lambda we have got lambda cancelling out it is lambda plus this thing. So, you calculate u n as I told you we are starting at n equal to 0 to final final can be any point of your choice.



Let me write it more clearly to some point that n equal to 0 this inverse which we just

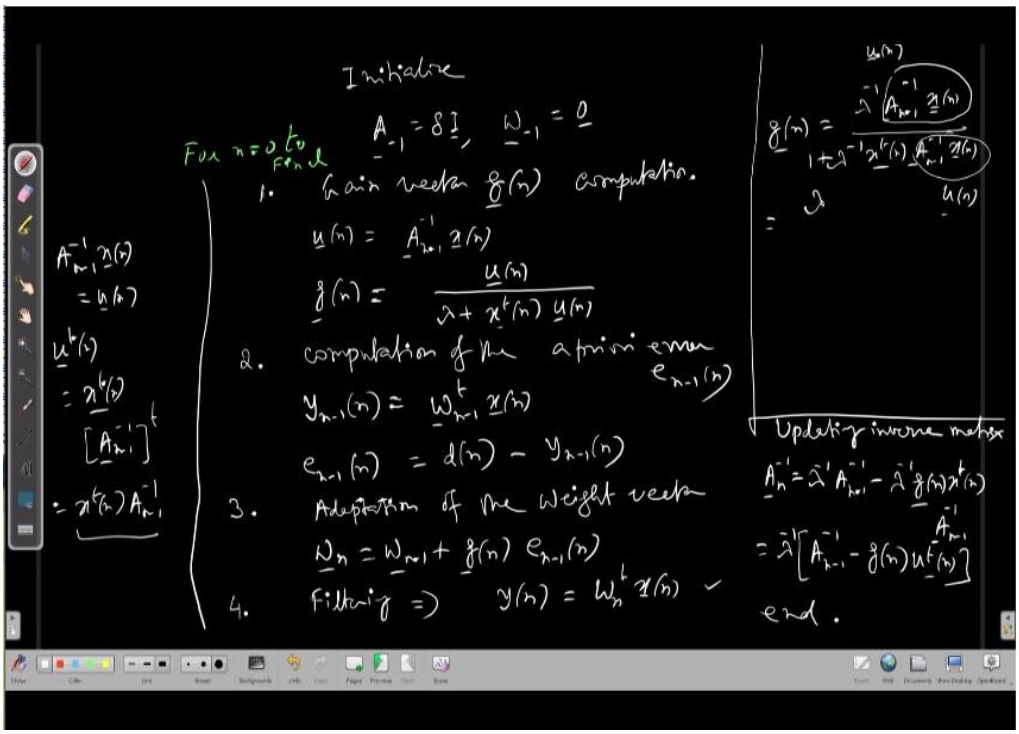
because a minus 1 inverse if I know a minus 1 and all the diagonal entries are positive. So, a minus 1 inverse is nothing, but 1 by delta times i. So, this I call u n using this u n I write g n as. So, I am assuming this to be known a n minus 1 inverse given at the nth stage. So, therefore, when I go for the n plus 1 stage I will have to make sure that a n inverse is also available ok and that is how we will proceed.

So, g n is u n vector divided by a scalar number and scalar number is lambda plus x transpose n u n this part is u n x transpose n u n all right. Then number 2 computation of the of the a priori by the way or maybe I 2 computation of the a priori error e n minus 1 n. So, for that remember we find out this output where we use the current data vector, but filter coefficient vector we use which is obtained by taking data up to the previous clock up to n minus 1 ok. So, that was some at the nth clock I am still using the current data vector only, we are using of the filter vector of the previous index ok. And there is this error is nothing, but current desired response minus this output.

Number 3 adaptation of the weight vector simply we derived the formula it was from w n minus 1 plus g n number 4. Now, this is the correct weight vector. So, filtering this is a common if I have filtering y n this I got the filter weight vector for the current cycle. So, I just use it to filter the current data vector and get the output ok. This is my actual this is a conventional style conventional you know filter output this is what we will take at filter output this was previously y n minus 1 n was just to compute this a priori error.

But now last stage is very important and that is updating inverse matrix that is I assume that a nth clock a n minus 1 inverse is available. Therefore, when I move to the next clock a n plus 1 that time I should make sure a n inverse is available. But we had already obtained the formula a n inverse what was that a n inverse was lambda inverse a n minus 1 inverse minus lambda inverse g n x transpose a n a n minus 1 inverse right this is what take lambda inverse common this is known to us a n minus 1 inverse g n. Now, this quantity x transpose n a n minus 1 inverse we have seen a a n minus 1 here we have seen inverse x n is u n. So, what is u transpose n x transpose n a n minus 1 inverse transpose.

Now, remember a n minus 1 is hermitian and we had proved earlier part in the of this lecture series that if a matrix is hermitian its inverse also is hermitian assuming inverse exists inverse also hermitian. So, therefore, its transpose is itself. So, it will be x transpose n inverse and that is what I have here x transpose n a n minus inverse. So, it is g n into u transpose n.



So, that is the end of this algorithm and also that marks the end of this course I hope through this course I could impart sufficient knowledge starting from you know probability random variable properties of correlation matrices hermitian form positive definite form optimal filtering adaptive filtering and lot of derivations which are useful in many other contexts and applications of adaptive filters.

And you know these are not only used in the context of adaptive filter adaptive filter was a pretext was a was a tool around which I could develop this theory ok, but all these are very useful in several other areas like communications for that matter wireless

communications, image processing, video processing, neural networks, deep learning and all that. So, I am sure you will be able to use this knowledge if you do not forget you will be able to use this knowledge in many other contexts as well. So, on that hope I do see all the best I conclude my lectures here. Thank you very much.