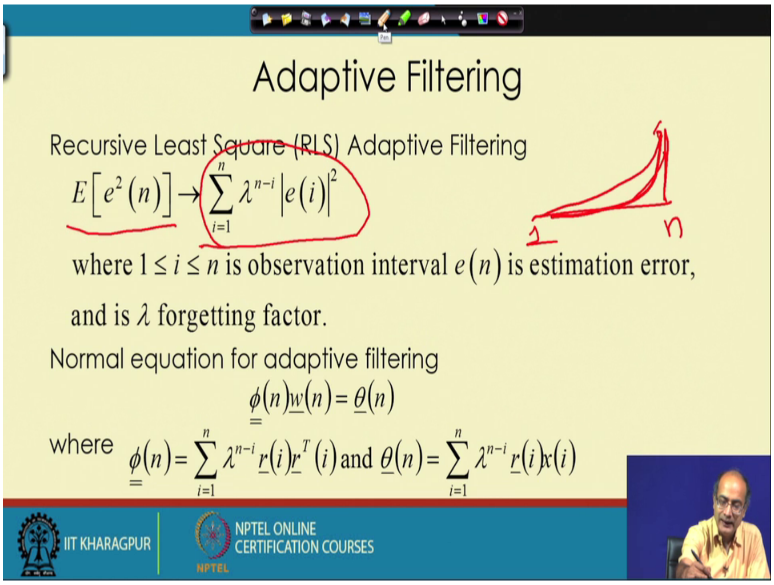
Biomedical Signal Processing
Prof. Sudipta Mukhopadhyay
Department of Electrical And Electronics Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 16
Artifact Removal (Contd.)

We will start the, this again adaptive filter we will go for the recursive least square adaptive algorithm. And here we have a small change compare to the LMS algorithm again the starting point is the same.

(Refer Slide Time: 00:28)



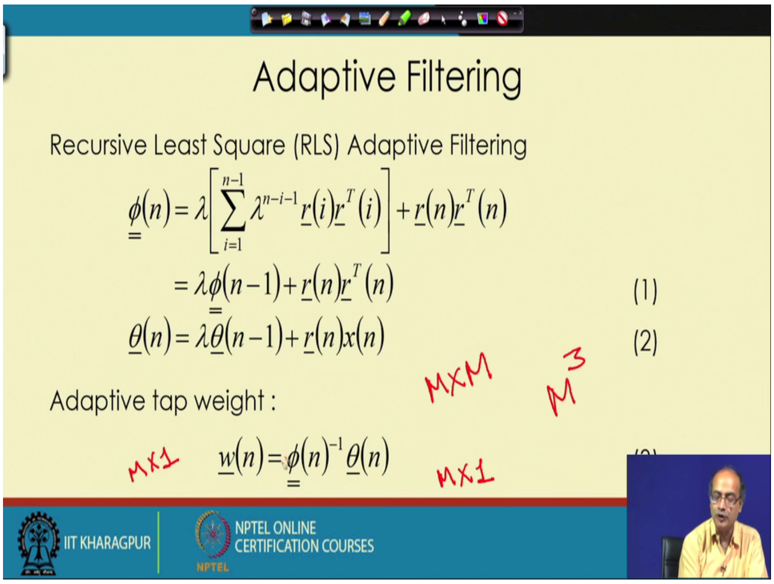
We look at that how we deal with the expected value of e square n. So, in this case instead of entirely dropping the expected operator we give rise to that the noise, what we called as noise in the output what we called as that with better noise. We are taking actually that this summation over the time we are taking a summation over the time for the last that n inputs, but in this case we are taking a time window lambda is the forgetting factor, now how it is working that say if this is the instant 1 and now we are at instant n the value of lambda here if it is less than 1, it would be here and as we are going back over the window then we are having power of n it will reduce like this.

So, in other word what we are suggesting that we though we are doing averaging we are giving the maximum weight to the error which is the present error and to the nearby part as it goes further and further the influence of it would be reducing, so that is the that main assumption of the RLS algorithms. So, now why it is actually we are taking it in this way that because we are giving more importance to the recent value of n it will help us to that adapt to the change in the situation, the smaller the value of lambda this 1 will become this one move actually more and more fast towards the 0 and thereby the changes will be more, but if we take lambda is very small then there is some penalty towards this.

The penalty is that the averaging also would be less and that means again it will give actually more with better noise. And like the previous case we have e n is the estimation error so we are having the new term introduced here that is the forgetting factor lambda, now the next part or the rest of the RLS algorithm is towards the fact that we need to have an update equation of the tap weight as we have seen it there.

So, let us move further that what we get here, that we get a normal equation phi n that is a matrix multiplied by the weight with the w n equal to theta n where this phi n it is a autocorrelation matrix of the reference input into the forgetting factor when we are summing it up we are having this forgetting factor, same way the theta n it is the cross between the vector of the reference input and the observed present value of I and the that the last n minus 1 values also and again the forgetting factor is come into play.

(Refer Slide Time: 05:31)

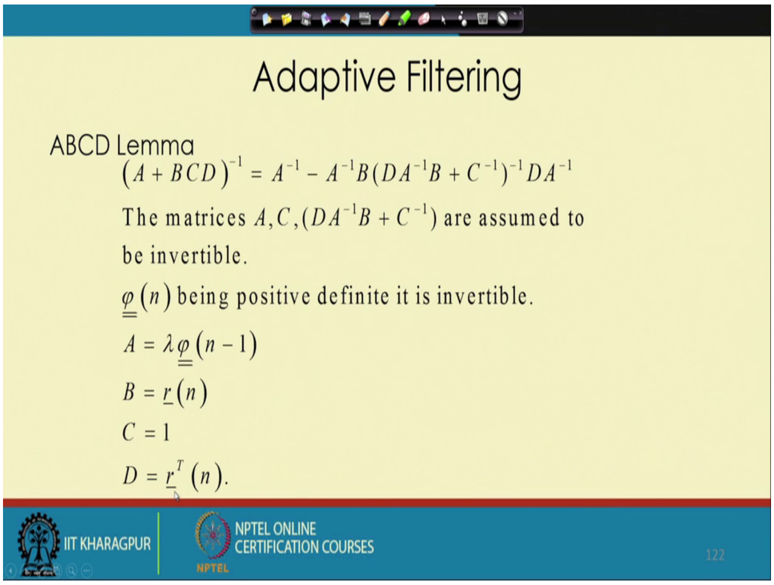


If it is a stationary process in that case we can take lambda equal to 1 and then as you see it will becomes same as our that the wiener filter. Now, let us look at that how we can compute these values, you see that this phi n can be written in this way that we can take it out the most recent contribution that the vector r n and transpose the vector r n that the present values, so if we take it out we are left with this term multiplied by lambda and this term is nothing but the previous value of phi that is the matrix phi n minus 1, so we get a that update equation of phi where if you know that that the previous value if you gets compute these parts and we take the multiplication with the lambda we can update the phi n. In the very similar way we can get an update equation for theta n where we can compute it from theta n minus 1 and the COS between the reference input vector and the observed present value x n.

So, what we get that we reduce the burden in computing that means we need not have to start from the scratch we can reduce that part of the computation and for the tap weight w n at each instance we can compute it by multiplying the inverse of phi n the inverse of this matrix into the vector theta n. Now here let us look at that what is the dimension of all these quantity see for w n yeah w n it is m by 1 dimensional vector theta is again m by 1 dimensional vector phi n is having dimension m by m and we know that the complexity of that matrix inversion is pretty high.

It is the order of complexity is actually m cube where m is the dimension of the matrix ok, so this is a (Refer Time: 08:32) an amount of computation what we need, because we have already seen that if we are taking a signal sample that 10 kilo hertz we have only 1 millisecond time to update the tap weight w n. So, this much amount of computation would be very challenging to do and for that we need to have a better formulation of these to make it for any practical use.

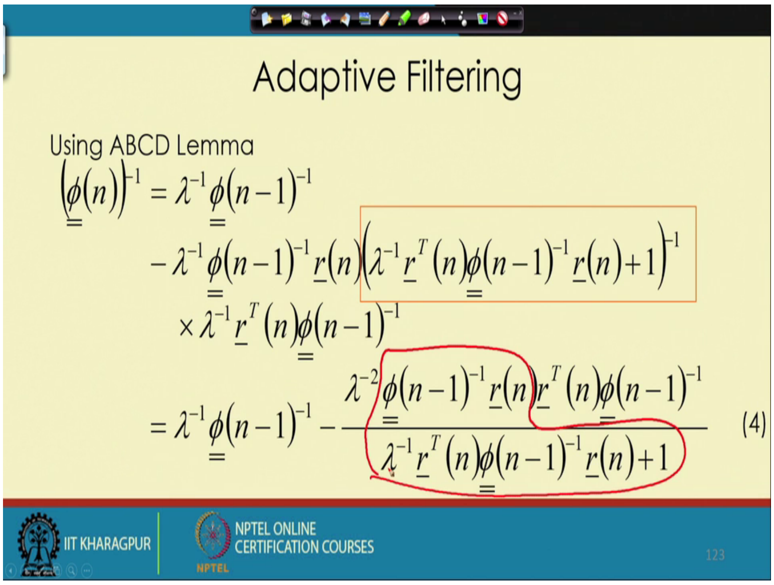
(Refer Slide Time: 09:23)



So, let us see that how this computation can be reduced and rest of the LMS algorithm would be a search for that in reducing that complexity. So, in an attempted that first we introduced one lemma called A B C D lemma, in this A B C D lemma the goal is that the matrix what we have to invert then in this case that phi n say if we can divided into few constituents in this form that if we can represent it by A plus B C D in this form, where we assume that A is invertible C is invertible then D A inverse B plus C inverse this is also invertible, in that case we can actually write the inverse of this in terms of the inverse of A inverse and C inverse and this quantity and how that can help, now if we can break it into a smaller parts which are easier to take invert then it can reduce our computation in the inversion.

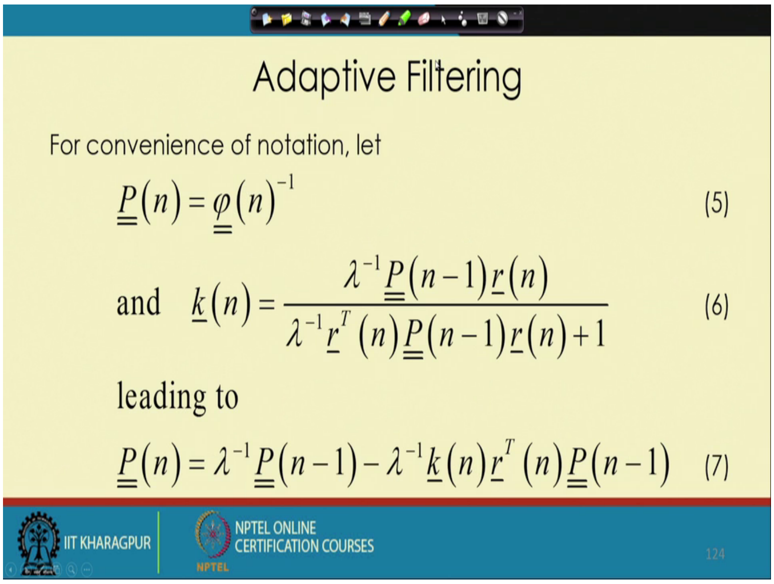
To be more precise that we recall that we have some update equation for phi with the help of phi n minus 1 plus A new term so here that A for the inversion of phi n we are taking A equal to lambda into phi n minus 1 and as we have computed the previous weight suddenly we must be having A inverse now for a B we have taken it r n, C is taken 1, so inversion gives the same value and D is again r T n. So, in that way actually we are actually writing phi n in this form, so once we can write it now we will try to make use of the A B C D lemma and represent phi n inverse in terms of this inverse of A inverse of C and the other terms.

(Refer Slide Time: 12:15)



So, as we write it in that form what we notice that here we get a part that what we have highlighted here these part the dimensionality is 1, so this is actually give you just a scaling and we are taking the inverse of the whole thing so we can write it in a will be better form that we can write it that that the these part is A that inverse minus this is the part and the part which is giving the scaling we have written as a denominator.

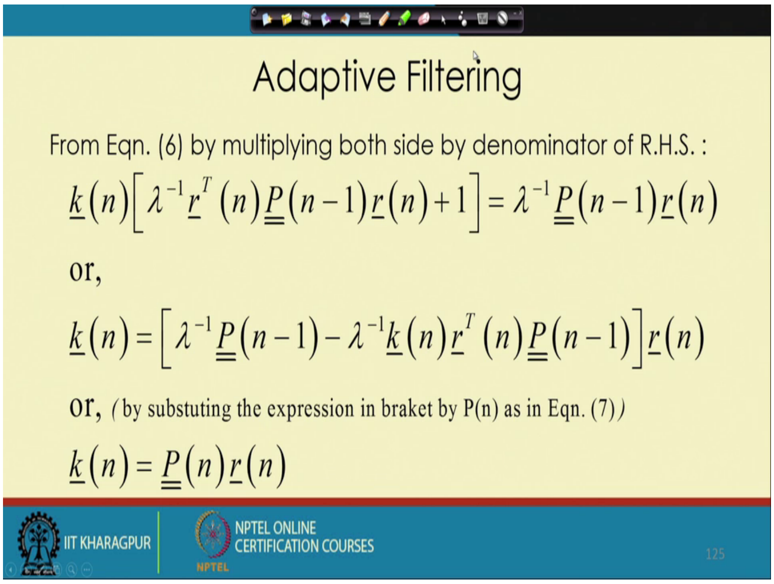
(Refer Slide Time: 13:28)



Now, we look for further simplification of it to reduce the computation further and to do that we would like to actually make the notation a little more simple and instead of phi n inverse we give it a new name called P n here, ok so the P n we have chosen again it is a matrix it is nothing but phi n inverse and another thing is introduced that is the vector k n and how the vector k n is chosen, let us look at the previous stage that we have this part this denominator and we have this part ok.

So, taking this part together we are taking that k n and direct assumption now the phi inverse of the new value of P n we can write in terms of that P n minus 1 and k n with the remaining terms that the transpose of r n and P n minus 1, so it suddenly that that looks actually greater it becomes more readable an expression.

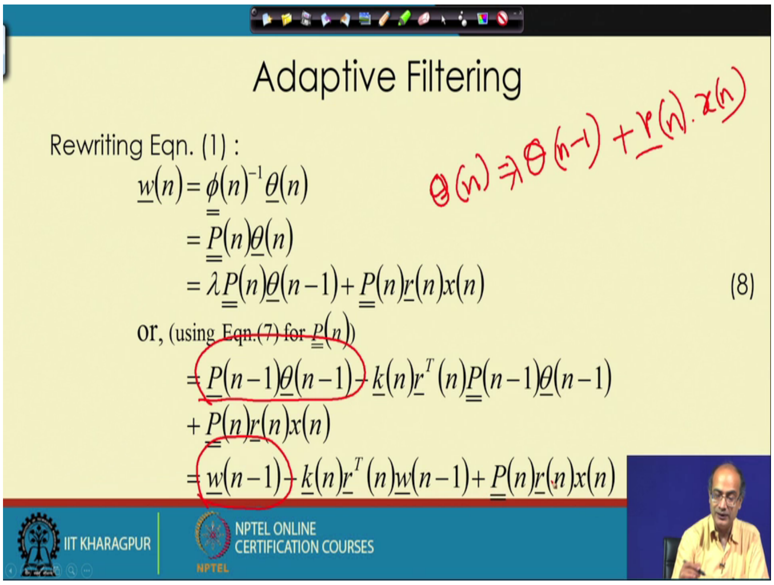
(Refer Slide Time: 15:20)



So, now by multiplying both the side by the denominator of the right hand side we try to actually simplify this and we get this new equation with k n and here we get two terms out of that that when you multiply with 1 we get k n we keep that in the right hand side and the product of the k n and this the first part of this term or the first term that we take in the right hand side and as this one has r n bit at the end this r one has they have bit at the end, we take that vector r n in as in common and we can write it in this way and what we realize that these part what we are looking at this is a same as the expression of P n in the equation 7 ok.

So, let us look at that that this was the expression of P n so using that replacing that expression with P n we get a very compact form we get k n is nothing but P n into r n where P n is a matrix r n is also a vector so that give rise to the vector k n.

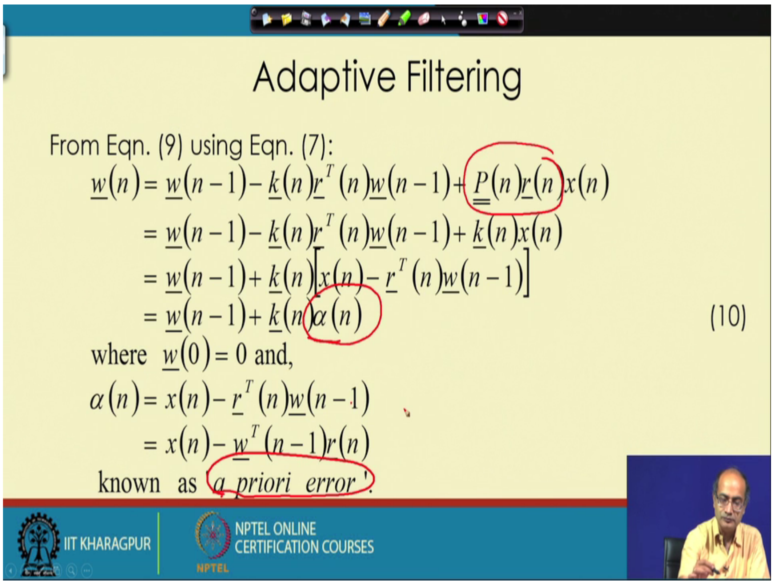
(Refer Slide Time: 17:04)



Now, let us go back that how we can compute the that the new tap weight that is w n. So, w n we know the expression is the inverse of the matrix phi n into that vector theta n so phi n inverse is nothing but P n into theta n and now if we actually write theta n expand theta n in terms of theta n minus 1 and the component that r n into x n that here we can actually if we go back that we will get this expression that theta n is nothing, but that theta n minus 1 lambda times plus r n into x n.

So, using that we have expanded it in this way and now in the next step what we do we actually replace P n, the P n in terms of P n minus 1 and the other term with k n so we put that, so we have now three terms and out of that what we get that the first part that this 1, this 1 looking at this equation w n that we get it is nothing but w n minus 1 and we have two more terms here involving w n minus 1 which we know that is the good part of it what are the things we do not know that is P n we to compute that we have we know the value P n minus 1 do not know k n these are the say the things we need to update to get w n.

(Refer Slide Time: 19:54)

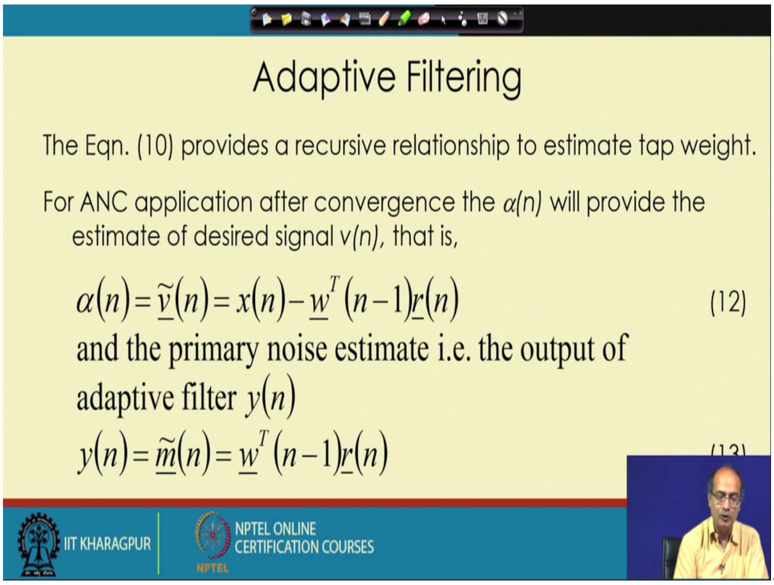


So, we look further for simplification and in the next step we start from their value here oh we are in the last page and from that expression what we do, we get the last part that k n equal to the matrix P n into r n so we are replacing this part with k n. Now, as the second and the third term both are having k n the forth line we are taking them k n in common and we are writing this term.

And if we look at this term that it is very similar to the error equation that the only change that in case of error equation that w instead of n minus 1 it would have been w n in this case that for these term we are giving a we are replacing with a new name that is called alpha, now if we at the beginning start with that initial with 0 then we can write that alpha is nothing but that x n minus the dot product of the reference signal and the previous tap weight that means that tap weight at the previous instant ok.

So, we can change their order because that this is the dot product and because we are using the tap weight of the previous instance it is called a priori error instead of errors we give a name for it that we call as a priori error. And here it would be interesting to note that if the process convergent that means there is no change from n minus 1 or there is hardly any change from n minus 1 to n for the tap weights then this is a priori error would be same as the error e n ok so in case of convergence that is what we would get.

(Refer Slide Time: 22:32)



Now, taking the previous equation that we would like to have a recursive equation to estimate the tap weight and for this particular application of adaptive noise canceller in short ANC that after the convergent the alpha n it will be same as the e n, so that is what we note and the equation what we have seen in the previous one that this is our update equation what we get.

Now out of that actually what are the thing what is the thing that is difficult to estimate or we need to take refer to compute that is only k n because w n minus 1 is known computation of that because w n minus 1 is known computation of alpha n is also trivial the only combustion thing is the k n that we have to compute to get the estimate or the new estimate that is w n, so with this we complete the part of RLS filtering.

Thank you.