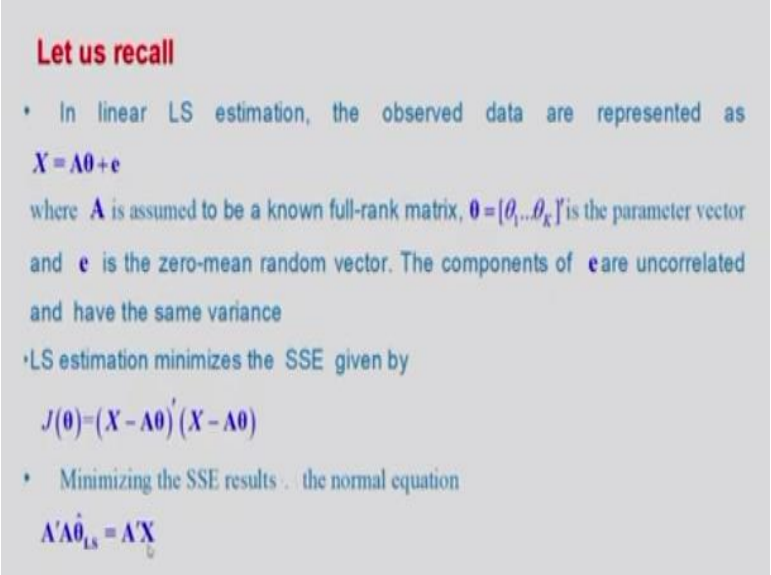
Statistical Signal Processing
Prof. Prabin Kumar Bora
Department of Electronics and Electrical Engineering
Indian Institute of Technology – Guwahati

Lecture 32
Recursive Least Squares (RLS) Adaptive Filter

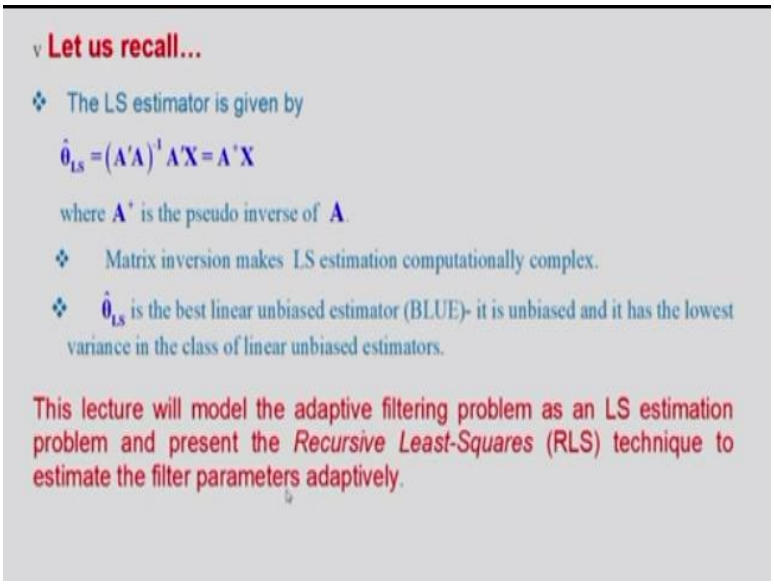
(Refer Slide Time: 00:49)



Hello students. Welcome to this lecture on Recursive Least Squares, RLS Adaptive Filter. Let us recall in linear LS estimation the observed data are represented as X is equal to A theta plus e vector, where X is the data vector A is assumed to be known full rank matrix and theta is the parameter vector. So theta is the linear combination of parameter vector plus some noise vector and e is the zero-mean random vector.

The components of e are uncorrelated and have the same variance. So e is a random vector and its component has the same variance and zero mean and because of this random noise, this data is random. LS estimation minimizes this sum square error given by this cost function that is J theta is equal to X – A theta transpose into X – A theta and minimizing this sum square error cost function, we get the normal equation that is given by A transpose A into theta hat LS that is equal to A transpose X.

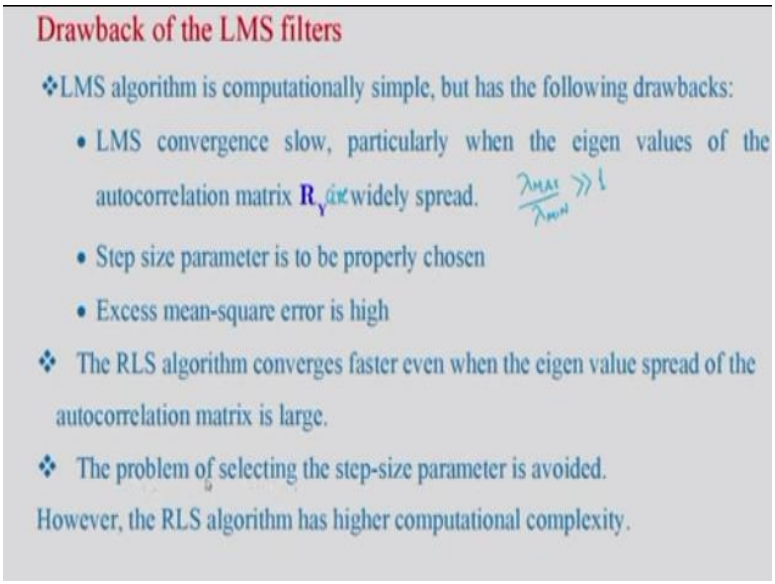
(Refer Slide Time: 02:17)



Now the LS estimator is given by this expression theta hat LS is equal to A transpose A inverse into A transpose X and this quantity A transpose A inverse into A transpose is denoted by A plus and is called the pseudo-inverse A plus is this pseudo-inverse. Therefore, theta hat LS is equal to pseudo inverse A into X. Now, we see that there is a matrix inversion. This matrix inversion makes LS estimation computationally complex.

We also note that theta hat LS is the best linear unbiased estimator BLUE. It is unbiased and it has the lowest variance in the class of linear unbiased estimators. So we will consider a class of linear unbiased estimators. In that class, theta hat LS has the lowest variance. This lecture will model the adaptive filtering problem as an LS estimation problem and present the recursive least square technique to estimate the filter parameters adaptively.

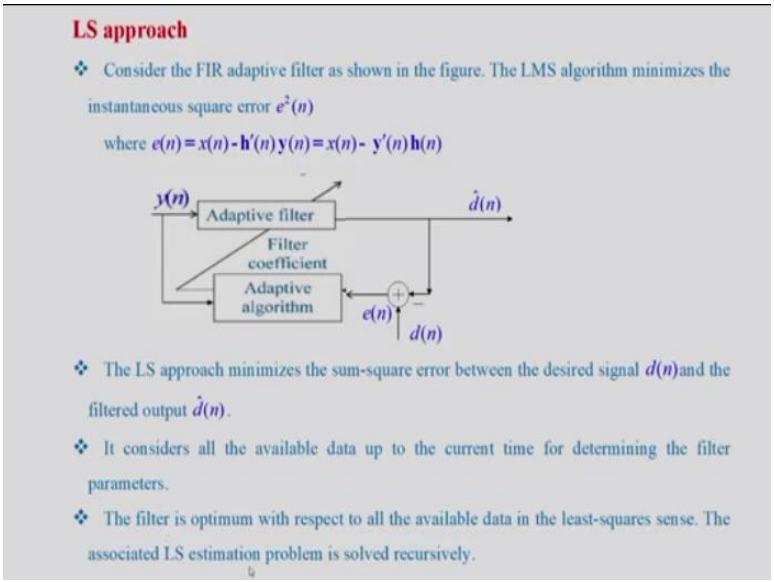
(Refer Slide Time: 03:54)



We will start with the drawback of the LMS filter. LMS algorithm is computationally very simple, but it has the following drawbacks. The major drawback is LMS convergence is very slow, particularly when the eigenvalues of the autocorrelation matrix Ry are widely spread, if lambda max divided by lambda mean is much greater than 1. Then convergence is slow. Step size parameter mu is to be properly chosen.

So that is one important factor, otherwise the algorithm may diverge. Excess mean square is high, particularly if step length is high, then this excess mean square error will be high. The RLS algorithm converges faster even when the eigenvalue spread of the autocorrelation matrix is large. So this is one important advantage of RLS algorithm. The problem of selecting the step size parameter is avoided in RLS algorithm. However, as we will see later, the RLS algorithm has higher computational complexity.

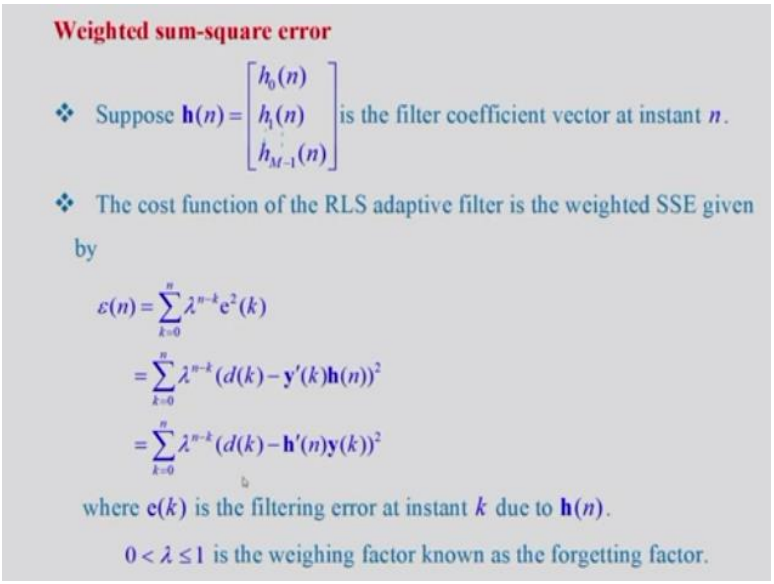
(Refer Slide Time: 05:40)



Let us see the FIR filter structure for adaptive filter implementation. So here yn is the input signal. This is the FIR adaptive filter. Filter output is d hat n and it is compared with a desired signal dn and dn – d hat n is the error. So that error is used to adaptively update the filter coefficient and those filter coefficients are used to filter this signal. We know that the LS approach minimizes the sum square error, SSE between the desired signal dn and the filtered signal d hat n, because here error is between the desired signal and the filtered signal.

Therefore, LS approach will minimize this sum square of this error. It considers all the available data up to the current time for determining the filter parameters. This is the advantage of RLS algorithm that it considers all the available data up to the current time for determining the filter parameters. The filter is optimum with respect to all the available data in the least-squares sense. The associated LS estimation problem is solved recursively. Now this filter will be optimum with respect to all the previous data. That is one important aspect and this estimation problem is solved recursively.

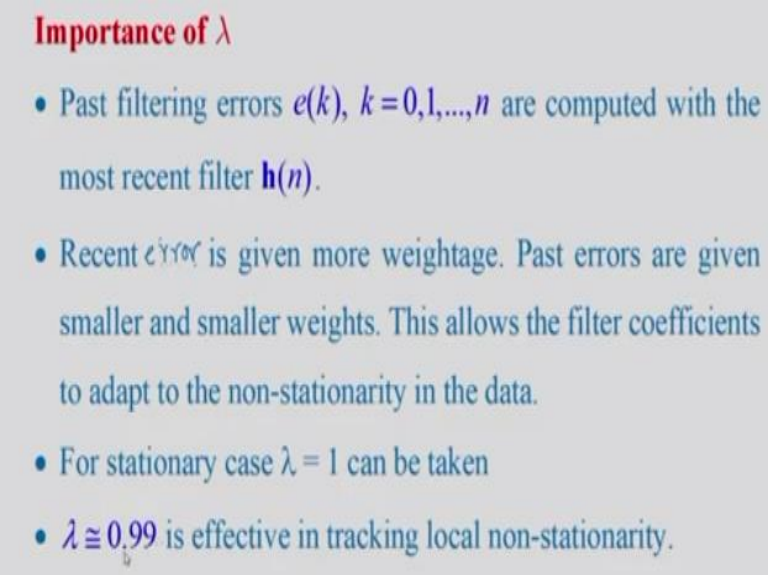
(Refer Slide Time: 07:39)



Suppose hn that is equal to this filter coefficient vector given by h0n h1n upto hm – 1. The cost function of RLS adaptive filter is the weighted sum square error. So here the error scores will be weighted. This is given by this; that is epsilon error is the sum squared error. This is equal to summation lambda to the power n – k into e square k, k going from 0 to n, so up to instant n from k is equal to 0 up to instant n, all the error squares are weighted and summed up.

And this ek, we can write as dk – yk vector transpose into hn vector and which can be also written as dk – hn vector transpose into yk vector. Now here ek is the filtering error at instant k due to hn. So here you see that filter at instant n, that is hn that filter is used to filter all the previous data yk, k from 0 to n. So that way ek is the filtering error at instant k due to filter hn. This we have to note and this quantity lambda which lies between 0 and 1 is the weighing factor known as the forgetting factor. So this factor is known as the forgetting factor.

(Refer Slide Time: 09:34)



Now let us see the importance of the forgetting factor lambda, no depth past filtering errors ek, k going from 0 to n are computed with the most recent filter hn. This is one important observation we have already made. Recent error is given more weightage; past errors are given smaller and smaller weights. How this is so? Because this is lambda to the power n – k, when k is suppose equal to n, then lambda will be equal to 1.

And when k is 0 suppose, then it will be lambda to the power n and this will be a very small quantity, because lambda is a number less than. That way recent error is given more weightage; past errors are given smaller and smaller weights. This allows the filter coefficients to adapt to the non-stationarity in the data. If data is non-stationary, then the recent error will reflect that non stationarity and since it is given more weightage, therefore we will get better estimation. For stationary case, lambda = 1 can be taken. Lambda approximately equal to 0.99 is effective in tracking local non-stationarity. So it should be less than 1 and we can take something around 0.99.

(Refer Slide Time: 11:22)

Normal equation

- The LS estimation problem is

Minimize $\varepsilon(n) = \sum_{k=0}^{n} \lambda^{n-k} \left(d(k) - \mathbf{y}'(k)\ \mathbf{h}(n)\right)^2$ with respect to $\mathbf{h}(n)$

- The minimum is given by $\frac{\partial \varepsilon(n)}{\partial \mathbf{h}(n)} = \mathbf{0}$

$$\Rightarrow 2\sum_{k=0}^{n} \lambda^{n-k} \left(d(k) - \mathbf{y}'(k)\ \mathbf{h}(n)\right)\mathbf{y}(k) = 0$$

$$\Rightarrow 2\sum_{k=0}^{n} \lambda^{n-k} \left(d(k)\ \mathbf{y}(k) - \mathbf{y}(k)\ \mathbf{y}'(k)\ \mathbf{h}(n)\right) = 0$$

$$\therefore \left(\sum_{k=0}^{n} \lambda^{n-k} \mathbf{y}(k)\mathbf{y}'(k)\right)\mathbf{h}(n) = \sum_{k=0}^{n} \lambda^{n-k} d(k)\mathbf{y}(k)$$

which is the *normal equation* for the LS estimation.

Now the LS estimation problem is given by minimize; this is sum square error up to instant n. This is given by summation lambda to the power n – k into dk – yk transpose into hn whole square, k going from 0 to n. This is the sum square error. This sum square error we have to minimize with respect to hn vector. So each component of hn we have to use for minimization. Now the minimum is given by this condition del en del hn vector is equal to 0 vector.

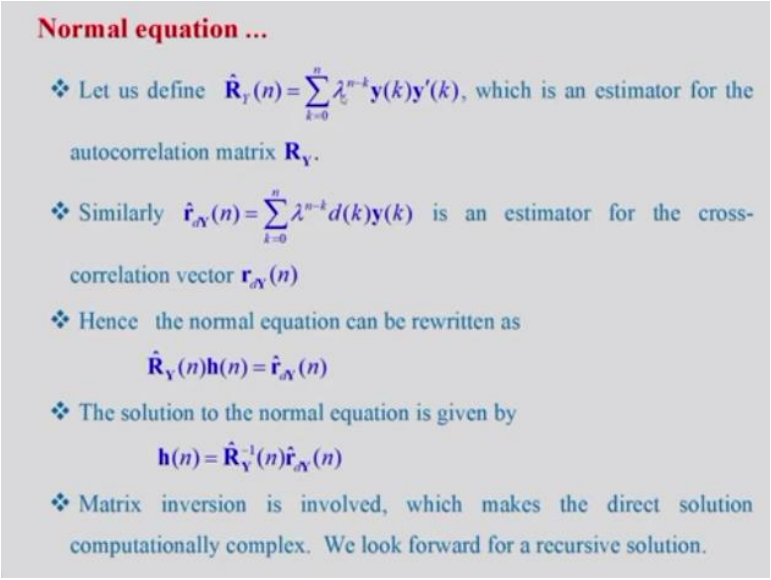
So here differentiation with respect to hn vector and we can carry out the differentiation with respect to hn vector. We will get because of this square 2 will come and then this part will be; they are as dk - yk transpose into hn and then because this quantity also has term hn. If we differentiate with respect to hn, we will get yk here. Therefore, the minimum condition will be now given by 2 times summation lambda to the power n - k into dk – yk vector transpose into hn vector into yk, k going from 0 to n, that must be equal to 0.

Now this yk, we can take inside and we note that this quantity is a scalar. So yk we can bring here. Similarly, this quantity is also a scalar; yk we can bring here. This yk is a vector. Therefore, we will get, 2 will be there. Anyway 2 can be cancelled. Summation y to the power n – k into dk into yk vector - yk vector into yk vector transpose into hn, k going from 0 to n is equal to 0.

So now bringing this part to the right hand side, we will get summation lambda to the power n - k into yk into yk transpose, k going from 0 to n into hn is equal to summation lambda to the

power n - k into dk that is a scalar into yk vector, k going from 0 to n. So we have a matrix equation now and this equation is known as the normal equation for the LS estimation. Now this quantity, this is similar to the estimators for autocorrelation matrix.

(Refer Slide Time: 14:42)

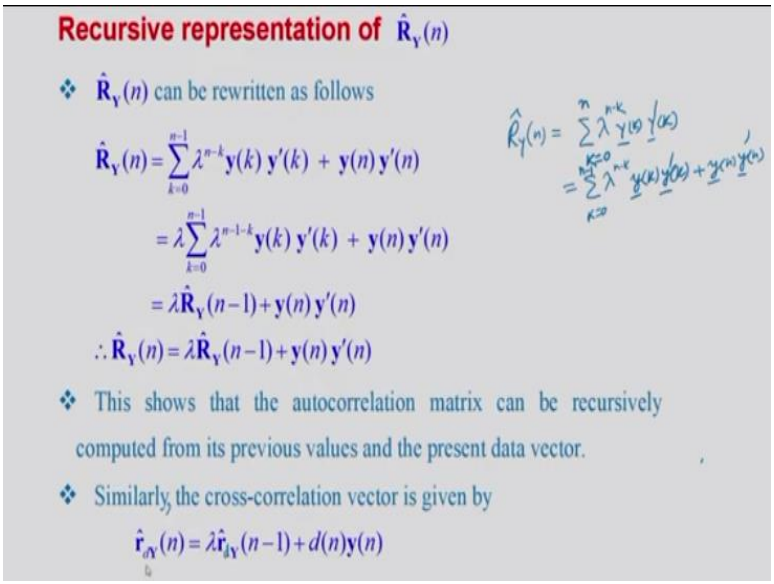


So we will denote Ry hat n equal to summation lambda to the power n – k into yk into yk transpose, k going from 0 to n. So this is a matrix. This is the estimator for the autocorrelation matrix Ry. Similarly, we can define this quantity summation k going from 0 to n lambda to the power n - k into dk into yk vector. So that way, this will measure the correlation between dk and component of yk vector and then you are summing up from k is equal to 0 to n.

So that way, this is also an estimator for the cross correlation vector Rdyn. So we can now write Ry hat n for this expression and rdy hat n for this vector. So hence the normal equation can be rewritten as Ry hat n matrix into hn is equal to rdy hat n vector. This matrix is invertible. Therefore, the solution of the normal equation is given by hn is equal to Ry hat n inverse into rdy hat n vector. So that way here, the filter coefficients are to be obtained by matrix inversion.

We have to invert the estimated autocorrelation matrix. Here matrix inversion is involved, which makes the direct solution computationally complex, because finding inverse of a big matrix is computationally very complex. We look forward for a recursive solution.

(Refer Slide Time: 16:55)



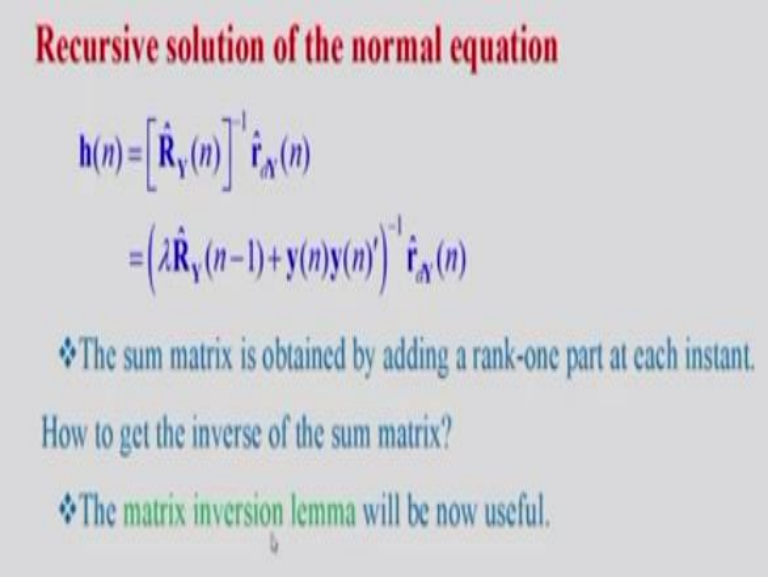
First, we will see the recursive representation of Ry hat n. Ry hat n can be rewritten as Ry hat n is equal to lambda to the power n – k yk vector into yk transpose, k going from 0 to n. Now, if we separate out the term involving yn, so when we put k is equal to n, then that term will be lambda to the power 0 is 1. So if we take the first n – 1 terms, then we will have summation lambda to the power n - k yk into yk transpose, k going from 0 to n – 1.

And last term when k is equal to n that will be given by lambda to the power n – n that is lambda to the power 0 is 1. So, last term will be simply yn into yn transpose. So therefore, Ry hat n can be rewritten as this summation; summation lambda to the power n - k into yk vector into yk transpose, k going from 0 to n – 1 plus yn into yn transpose. Therefore, this term we can write as lambda into summation lambda to the power n – 1 – k yk vector into yk transpose, k going from 0 to n – 1 + yn into yn transpose.

Now this quantity is nothing but this is the estimator for the autocorrelation function at n – 1, because it is from k is equal to 0 to n – 1. Therefore, this part is Ry hat n n – 1. Therefore, we get lambda times Ry hat n – 1 plus this term, that is yn into yn transpose. Therefore, we have established that the autocorrelation estimator at instant n is equal to lambda times autocorrelation estimator at n – 1 + yn into yn transpose.

So that way autocorrelation function can be incremented. This shows that the autocorrelation matrix can be recursively computed from previous values and the present data vector, because yn is the present data vector. So yn multiplied by yn transpose, this part we are adding to the previous estimator. Similarly, the cross correlation vector is given by rdy hat n. This can be recursively obtained lambda times rdy hat n – 1 + dn into yn vector. So this way we can update this cross correlation function.

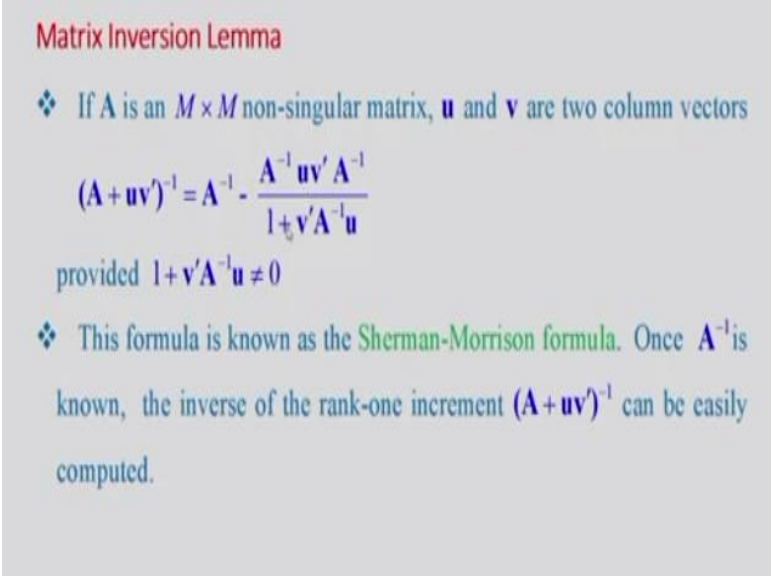
(Refer Slide Time: 20:37)



Now we will see how we can get the recursive solution of the normal equation. We have hn is equal to Ry hat n inverse into rdy hat n and this we can write as, because this term we can write recursively, so that is lambda times Ry hat n – 1 + yn into yn transpose whole inverse into Rdy hat n. This sum matrix is obtained by adding a rank one part. This part is the rank one part, because this is only one column is there.

That column you are taking transpose and you are getting the matrix. This is a rank one matrix. So the sum matrix is obtained by adding a rank one part at each instant. Instant n, we are adding this rank one part. How to get the inverse of this sum matrix that is a problem. The matrix inversion lemma will be now useful. There is a powerful result what is known as the matrix inversion lemma. We will see that now.

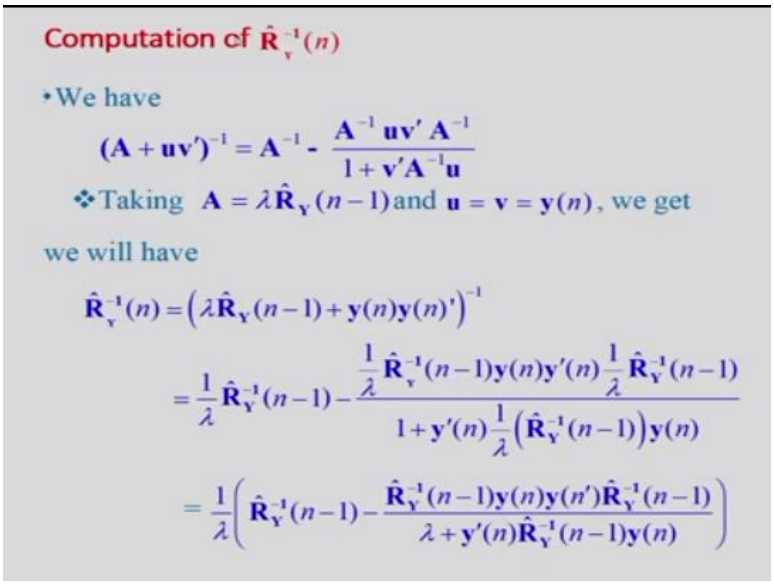
(Refer Slide Time: 21:53)



If A is an M by M non-singular matrix, so that inverse exists, u and v are two column vectors, then inverse of A plus uv transpose is equal to A inverse - A inverse uv transpose A inverse whole divided by 1 + v transpose A inverse into u, provided this denominator term 1 + B transpose A inverse into u is not equal to 0. In this way, we can get the inverse of this sum in terms of the inverse of A and the matrix u and v itself.

So there is no other inversion, except A inverse and if we know in A inverse, then this can be computed algebraically like this. This formula is known as the Sherman-Morrison formula. Once A inverse is known, the inversion of the rank one increment A + uv transpose inverse can be easily computed. This is the advantage of the matrix inversion lemma. So suppose if we know A inverse, then inverse of the rank one increment, so A + uv transpose that can be easily obtained by this formula.

(Refer Slide Time: 23:39)

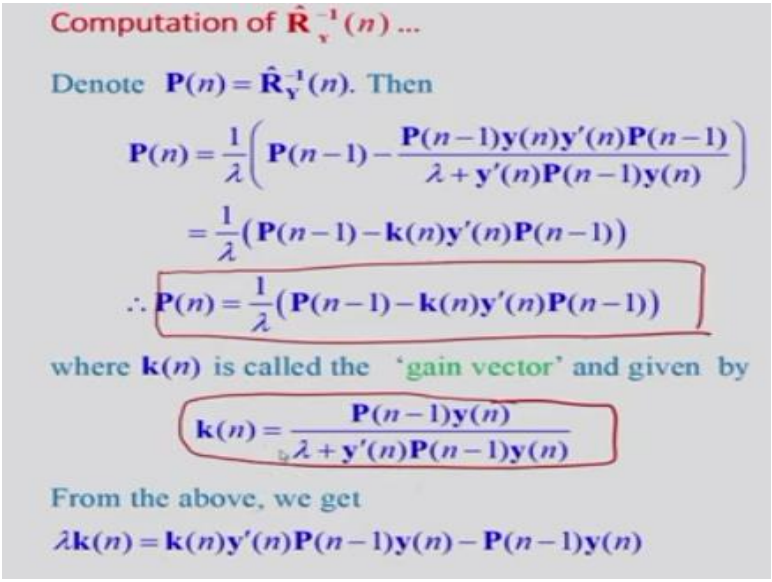


Now, let us see how to find out the inverse of the autocorrelation matrix. So we have this result. Matrix inversion lemma taking A is equal to suppose lambda Ry hat n – 1 and u = v = yn vector. So what we will get then? The inverse of Ry hat n is given by the inverse of lambda Ry hat n – 1 + yn into yn transpose and now using these results A is equal to lambda times Ry hat n – 1 and u = v = yn, we will get this inversion, will be equal to by using this formula 1 by lambda times Ry hat inverse n – 1 minus, now numerator this inverse of this quantity lambda into Ry hat n – 1.

That is 1 by lambda into Ry hat inverse n – 1, then u and v are equal yn into yn transpose into again inverse that is 1 by lambda into Ry hat inverse n – 1 divided by now denominator term 1 is there 1 + v is yn, therefore this will be yn transpose, 1 by lambda into Ry hat inverse at n – 1 into yn. We can take 1 by lambda outside. So we will get 1 by lambda into Ry hat inverse n – 1 – Ry hat inverse n – 1 into yn into yn transpose into Ry hat inverse n – 1.

Divided by denominator will be, because lambda if we multiply by lambda this one will be lambda, so lambda + yn transpose into Ry hat inverse n – 1 into yn. So this is the expression for Ry hat inverse at instant n and which is obtained from the inverse at instant n – 1 and it is updated using the current data.

(Refer Slide Time: 26:27)

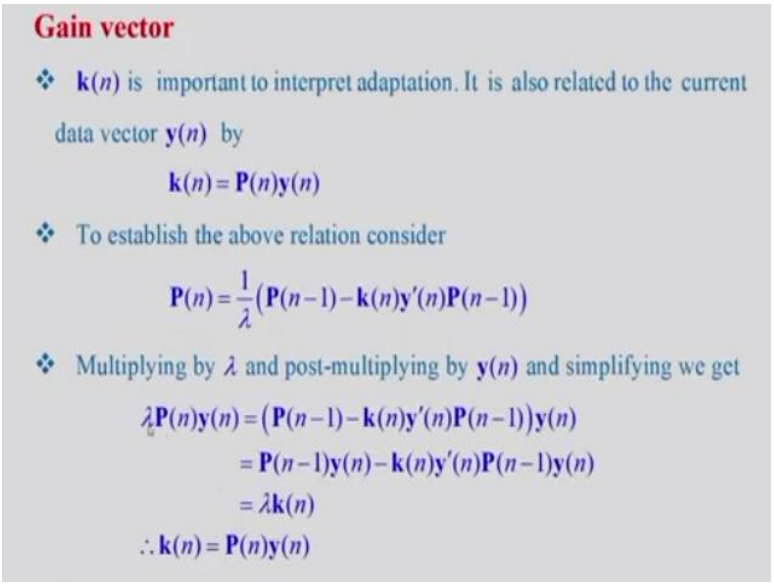


Now, we will denote this term Ry hat inverse n that is inverse of the autocorrelation estimator as Pn matrix. This is the standard notation Pn is equal to the inverse of the estimated autocorrelation matrix. Pn is equal to Ry hat inverse and then from this, this Pn will be equal to 1 by lambda into Pn – 1 minus, here also Pn - 1 into yn yn transpose into Pn – 1 divided by lambda + yn transpose into Pn – 1 into yn. So that way, we write Pn is equal to 1 by lambda times Pn - 1 - Pn - 1 into yn into yn transpose into Pn - 1 divided by lambda + yn transpose into Pn - 1 into yn.

Now, this part Pn - 1 into yn divided by lambda + yn transpose into Pn – 1 into yn, this part we will denote by a vector that is gain vector kn. Therefore, we can let Pn = 1 by lambda into Pn – 1 – kn that is the gain vector multiplied by yn transpose into Pn – 1. So this is the expression for recursive estimation of Pn. Pn = 1 by lambda times Pn – 1 – kn into y transpose n into P n – 1. This kn is called the gain vector and given by kn = Pn – 1 into yn divided by lambda + y transpose n into Pn – 1 into yn. This is the way we define the gain vector.

So this is the updating for the inverse of the autocorrelation matrix and this is the updating for the gain vector and from this result kn = Pn – 1 into yn divided by lambda + y transpose n into Pn – 1 into yn, we can easily show that, because we can multiply lambda into kn and that will be equal to kn into y transpose n this one kn into y transpose n into Pn - 1 into yn minus this part Pn minus into yn. This result is also useful for us.

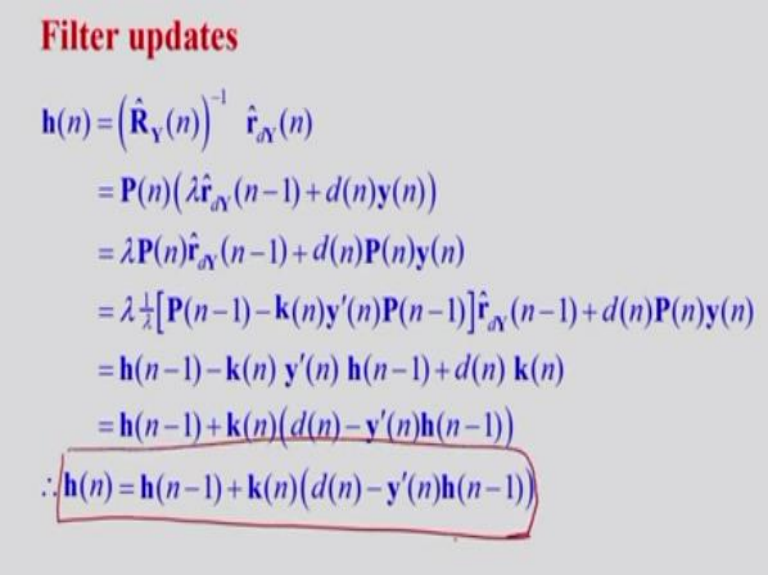
(Refer Slide Time: 29:43)



The gain vector kn is important to interpret adaptation. It is also related to the current data vector yn by this relationship kn = Pn times yn where Pn is the inversion of the autocorrelation matrix. This result we can establish. We start with the expression for Pn. This is the expression for Pn; that we are considering Pn is equal to 1 by lambda times Pn – 1 - kn into yn transpose into Pn – 1 and multiplying by lambda and post multiplying by yn, because this is a matrix.

We are post multiplying by yn and simplifying we get lambda times Pn into yn is equal to Pn – 1 – kn into yn transpose into Pn – 1 into yn, because we are post multiplying by yn. So that way, this will be equal to Pn - 1 into yn. This is Pn – 1 into yn minus kn into yn transpose into Pn – 1 into yn. Now this result we know. We have shown that this result is equal to lambda times kn. So therefore, we write here lambda times Pn into yn is equal to lambda times kn. Therefore, kn will be equal to Pn into yn. So this result we will be using.

(Refer Slide Time: 31:30)



Now, let us see how to get the update of the filter. So hn is equal to Ry hat n inverse into rdy hat n. Now this one, we can write as Pn, because this is our notation Pn into rdy hat n – 1 + dn into yn. So that way, we will get this expression is equal to lambda times Pn into rdy hat n – 1 + dn into Pn into yn. Now we know the expression for Pn that expression we will use.

So this expression, we will be using. So lambda into 1 by lambda times Pn - 1 – kn into yn transpose into Pn – 1 into this R hat dy n – 1 plus again dn yn into Pn. So that we can write as dn into Pn into yn and therefore now, this quantity Pn – 1 into rdy hat n – 1 that will be A sub n – 1, because this is the inverse of the autocorrelation matrix multiplied by the cross correlation function that will be h of n – 1 and this is kn and similarly this is P n – 1 into rdy hat n – 1.

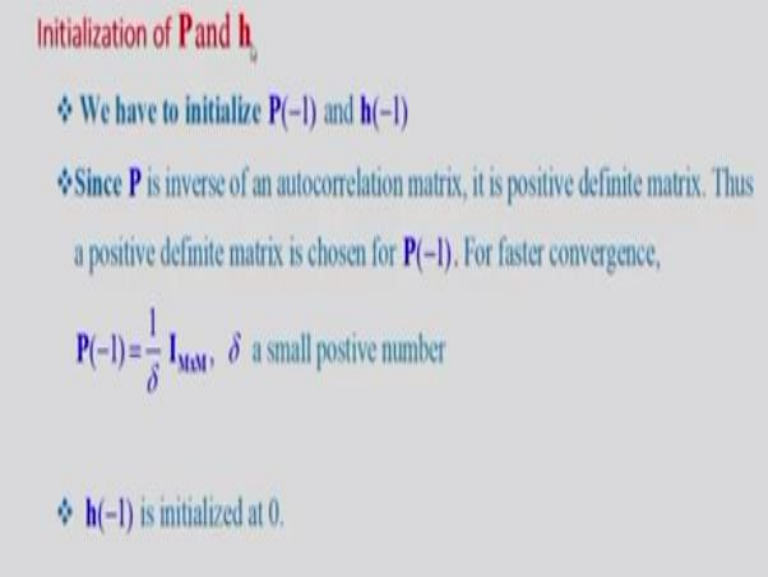
That will be again hn – 1 and this part Pn into yn that we have established that it is equal to kn. So here, we have established that Pn into yn is equal to kn. Now we can take kn common here from this expression and this expression; therefore, we will get h of n – 1 + kn times, this is dn - y hat n into A sub n – 1. Now this is the desired signal and this is y hat n into hn – 1 that is the filtered output, because of the filter hn – 1 and the difference will be the error.

This error is scaled by gain kn and then added to the earlier estimate of hn. So that way, hn is equal to hn – 1 plus kn times some prediction error. So this interpretation is helpful. Later on, we will see similar results for (()) (34:46) filter. Therefore, this is like prediction error, because this

you are getting by filtering with previous filter coefficients. So that way this is the predicted output and then this is the desired output.

That difference is the prediction error that is scaled by kn and you are adding to hn – 1 to get hn. So that way we have also the recursive relationship for hn. This is the recursive relationship for hn. So we have 3 recursion relation now for hn and for kn and Pn.

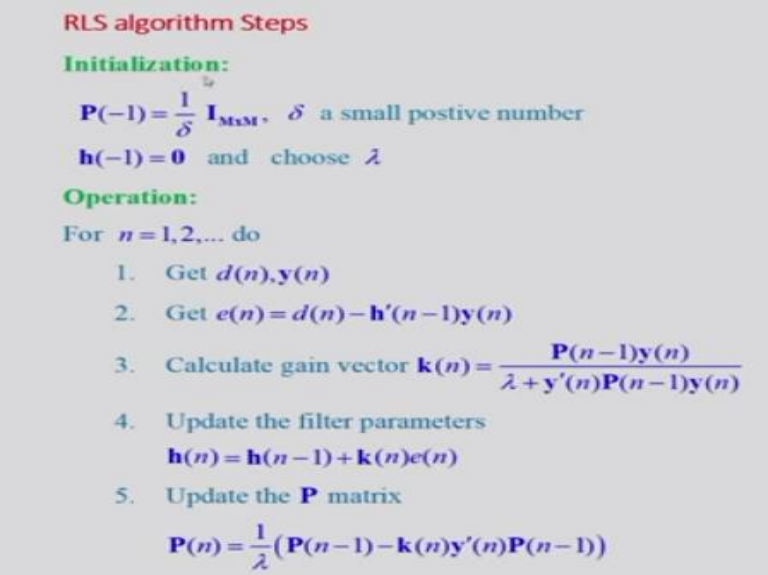
(Refer Slide Time: 35:35)



So these recursive relations are there, but we have to initialize Pnh, P matrix and h vector. We have to initialize P of minus 1, because at n = 0, we will need P of minus 1 and h of minus 1. Since P is inverse of an autocorrelation matrix, it is positive definite matrix. Thus, a positive definite matrix is chosen for P of minus 1. For faster convergence, we write P of minus 1 is equal to 1 by delta into I M by M, where this is the identity matrix divided by delta.

If we choose delta is small positive number, then 1 by delta will be large. Therefore, this will be a diagonal matrix with large diagonal component. This is required for faster convergence that we will see again later and h of minus 1 is initialized at 0.

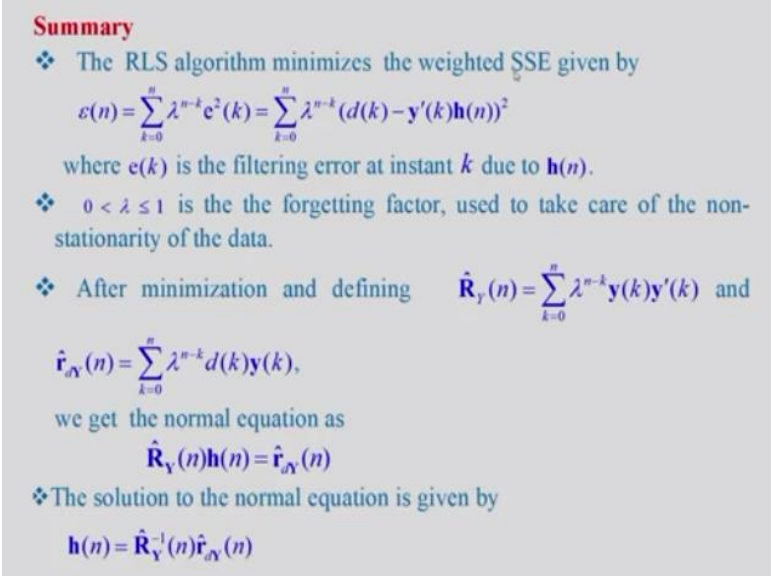
(Refer Slide Time: 36:36)



Therefore, the RLS algorithm can be presented in the following steps. First is initialization, that inverse of the autocorrelation matrix that is initialized at 1 by delta times IMM, then h of minus 1 is equal to 0 and we have to choose proper value of lambda, that is the forgetting vector. Now operations are for n = 0, 1, etc. What we will do? Get dn and yn; this is the desired signal. This is the input data, then find out the error en = dn – h transpose n – 1 into yn.

Calculate the gain vector kn that is this expression P of n minus 1 into yn divided by lambda plus yn transpose into Pn – 1 into yn. Then update the filter parameters hn is equal to hn – 1 + kn into en and then update the P matrix by this relation, PN is equal to 1 by lambda times Pn – 1 – kn vector into yn transpose into Pn – 1. So this way this RLS algorithm can be implemented.

(Refer Slide Time: 38:12)

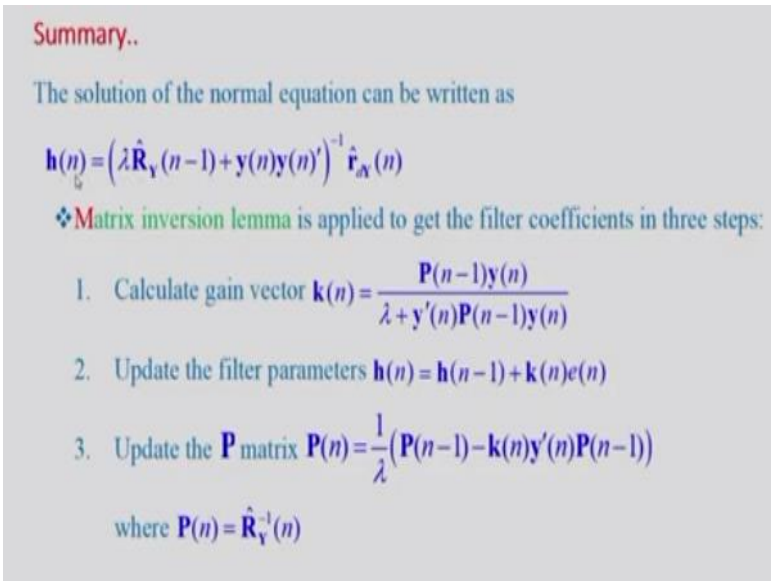


Let us summarize the lecture. The RLS algorithm minimizes the weighted sum square error is equal to summation lambda to the power n - k into e square k, k going from 0 to n, which is equal to summation lambda to the power n – k into dk - yk transpose into hn whole square k going from 0 to n, where ek is the filtering error at instant k due to filter at instant and this is very important observation, because you have to use the filter at instant n to filter all the previous data.

Zero less than lambda less than equal to 1 is the forgetting factor used to take care of non-stationarity, for non-stationarity lambda is less than 1 usually 0.99 like that and for stationarity we can take lambda equal to 1. After minimization and defining this quantity Ry hat n is equal to this. This is the weighted sum of this matrix and this is the estimator for the autocorrelation matrix. Similarly, this part rdy hat n is defined by this quantity.

This is the estimator for the cross correlation function. So we get the normal equation Ry hat n into hn is equal to rdy hat n. We get the normal equation as on this autocorrelation matrix into hn is equal to cross correlation vector. The solution to the normal equation, this is the normal equation, is given by hn is equal to because we have to invert this ry hat inverse n into rdy hat n vector. So that way, we can find out hn.

(Refer Slide Time: 40:29)



This solution of the normal equation can be written as hn because using the recursive relation for Ry hat n, hn can be written as lambda times Ry hat n - 1 + yn into yn transpose whole inverse into rdy hat n. So that way we can write hn and now we can do the matrix inversion lemma is applied to get the filter coefficient in three steps. So what are these three steps: calculate the gain vector kn given by Pn – 1 into yn divided by lambda + yn transpose into Pn – 1 into yn.

This is the gain vector. Update the filter parameters hn = h of n - 1 plus gain vector multiplied by en, where en is the difference between dn and the filtered output. That we have shown earlier. Then we have to update the P matrix by Pn is equal to 1 by lambda times Pn - 1 – kn vector into yn transpose into Pn – 1 and in these steps Pn is equal to inverse of the estimated autocorrelation matrix. Thus, we have derived the RLS algorithm. In the next lecture, we will discuss some important properties of RLS adaptive filters and compare those properties with those of LMS filters. Thank you.