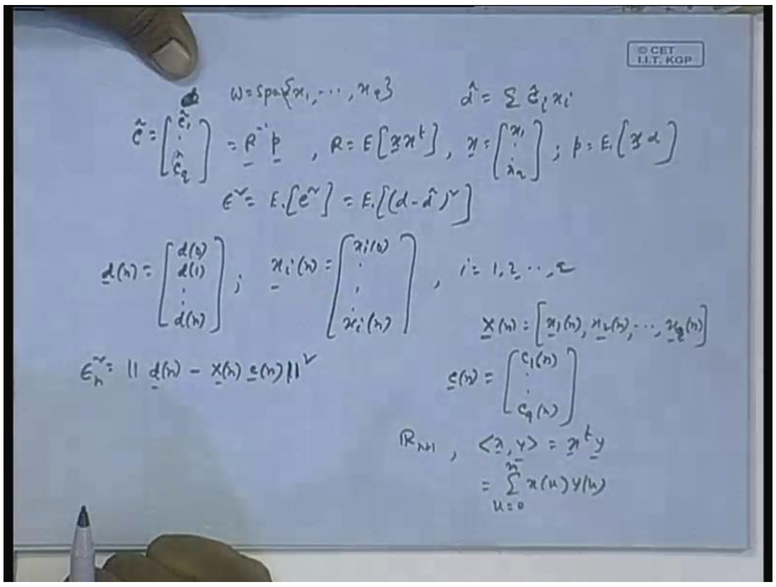
# Adaptive Signal Processing

**Prof. M. Chakraborty**
**Department of Electrical & Electronic Communication Engineering**
**Indian Institute of Technology, Kharagpur**

## Lecture - 29
## RLS Approach to Adaptive Filters

(Refer Slide Time: 01:04)



$W = \text{span}\{x_1, \cdots, x_q\}$ $\quad \hat{d} = \sum \hat{c}_i x_i$

$\hat{\underline{c}} = \begin{bmatrix} \hat{c}_1 \\ \vdots \\ \hat{c}_q \end{bmatrix} = \underline{R}^{-1}\underline{p}$, $R = E[\underline{x}\underline{x}^t]$, $\underline{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_q \end{bmatrix}$; $p = E(\underline{x} d)$

$\epsilon^2 = E[e^2] = E[(d-\hat{d})^2]$

$\underline{d}(n) = \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(n) \end{bmatrix}$; $\underline{x}_i(n) = \begin{bmatrix} x_i(0) \\ \vdots \\ x_i(n) \end{bmatrix}$, $i = 1, 2, \cdots, q$

$\underline{X}(n) = [\underline{x}_1(n), \underline{x}_2(n), \cdots, \underline{x}_q(n)]$

$\epsilon_n^2 = \| \underline{d}(n) - \underline{X}(n)\underline{c}(n) \|^2$

$\underline{c}(n) = \begin{bmatrix} c_1(n) \\ \vdots \\ c_q(n) \end{bmatrix}$

$\mathbb{R}_{n+1}$, $\langle \underline{x}, \underline{y} \rangle = \underline{x}^t \underline{y}$
$= \sum_{k=0}^{n} x(k) y(k)$

Case study let us continue. So, suppose we have a random variable d, and there is a set of random variables x 1 dot dot dot say x q, and we construct the space span by them; you call that w as the span of this, okay. What we did yesterday is that suppose you want to project d orthogonally the space w. That means you want to find out d cap as sum c cap i x i, where c cap i is the best estimate; that is if you really orthogonally project the corresponding combiner coefficients I denote by c cap I, so that the error between d and d cap that is having minimum variance.

Here in this space let us have a vector space of random variables that the inner product was a correlation between random variables. So, the norm was the variance, norm square is the variance and all that. So, if you take the difference between d and d cap; that error will have minimum variance, and that error will be orthogonal with each of these x 1 to x q; that is the correlation will be 0 and all that, okay. That is what we found; that will give you the optimum coefficients, but that optimum coefficient will lead the auto code is the correlation values between x 1 to x q and the cross correlation between the external fellow d with each of x 1 to x q.

Because that optimum if you really have c vector c cap vector as c 1 cap dot dot dot c q cap. This was R inverse p where R was p x; all are real. So, x has transposed; x is x 1 dot dot x q, and p is the cross correlation vector, right; this is what we know, this you know. Then I said that suppose we do not know the correlation value. So, they change from time to time. So, you have to make it adaptive. First step was to go for steepest decision that time, okay. Worst position exists if you replace this gradient and gradient it needs r and p, but something which I mean in the beginning only I said, suppose you do not have that they change with time.

Then I went into the approximation replaced R by wild estimate x into x transpose for a particular vector x. So, only p s just x into d and I got into Levinson Algorithm, but I paid the price there that convergence is not absolute; it is just a convergence in mean. I will be taking more time because as suggested in search technique and that too it is a very approximate search technique. Then I said that the other approach will be what? That this error is equal to epsilon square as e square, where this is d minus d cap square.

This c i caps for minimizing these; that is how they become optimal, okay. Then the other approach was that that let me find out an approximate value of this numerical value, how? I take data samples of d and also for each of the variables x 1 to x cube and form these data vectors d n as see you can start at any point; I am starting at n equal to zero d n. And for any x i n vector x 1 x i 0 dot dot dot x i n; that is I collected data at time equal to 0, collected data at time equal to 1, so on and so forth, i x i n where i could be 1, 2 up to q, 1, 2 up to q, okay.

And you form the matrix x n as first column is x 1 n, second column is x 2 n; thought, it is repetition, I need to rewrite this, because I will be using this x q n extensively today. Then what I said? That if you minimize this if you take this vector x n into a vector c n where c n consists of some linear combiner coefficients, but since I am using data up to n x index I am putting an index at c 1 to c q n. So, x n into c n means what linearly combining these columns by these coefficients, is it not?

So, if you take the vector space span by this column vectors, then x n into c n will be a linear combination of those basis vectors. So, some vector belonging to the column space of x n, we call it column space, d n minus that is the error. And if you take the non-square of the error, that will be what? For each entry for the first entry, take the errors, square it up; for the second entry, take the error, square it up. Add all those I mean and all the terms; you remember how I defined the inner product case yesterday, no? That random in vector space r m plus 1, where inner product x and y was x transpose y or y transpose x which was same as x k, y k; all are real; so no k 0 to n, okay.

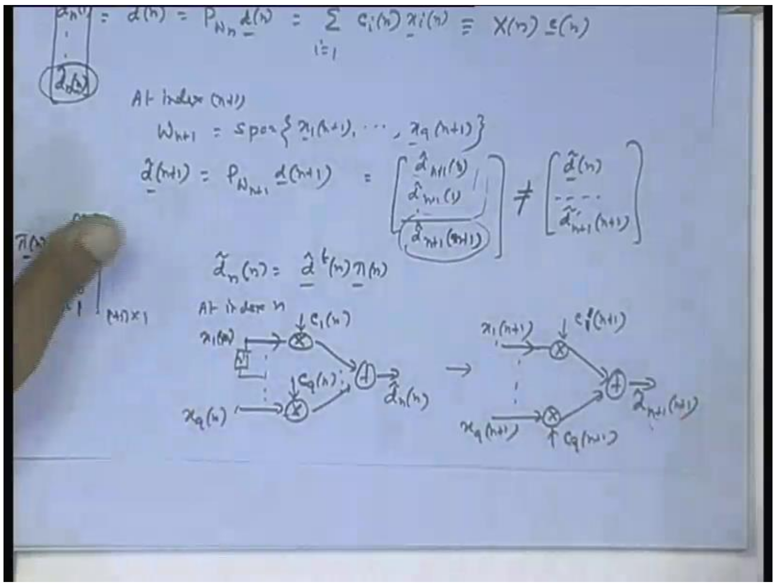
So, variance is k 0 to n x square k; that means x square 0, x square 1, x square 2, like that. Similarly, if I take the error taking variance; that means this error at 0 with index; square up the error at one th index, square it up. So, add all of them; that is the way to add. That if you divide by the total number of terms n plus 1 that will be a good estimator epsilon square; if I minimize that quantity to with respect to the c's, then effectively I will be minimizing these, because if this and this non-square if I call it epsilon n square, you know this norm square means what? Take this error vector; take the zero th component, square up; take the first component, square up; take the second component, square up; add all of them.

That is the norm square; that is the sum of the squares of each error; sum of square of errors, but that will be that of course divided by n plus 1 for everything. We will get very good estimator of epsilon square if I have got sufficient number of terms. See if you take this to be very close to this and minimizing this means almost minimizing this. So, the coefficients that we will get will be very close to the optimal ones. And in fact in practice, you really do not need up to infinite number of samples; just 100, 150 or 200, 250 is good enough.

Whatever value we have for this that it almost like epsilon square, and as a reality if you minimize that which is to the coefficients, we minimize that; you will essentially get the optimal coefficients c caps. And that is why if the convergence and all will be absolute and will be fast also; there is no search technique involved and all that and will be exact convergent, okay. So, that means what I have to do? I have to project d n orthogonally on the column space of this matrix; that is the space span by these columns.

Okay, find out the linear combiner coefficients; for n sufficiently large, there will be most equal to this coefficients, because if you take this square and divide by n plus 1, minimize or do not divide by n plus 1 and minimize in either case, because your minimization is with respect to the coefficients in vector; you will get the same result; that is a quadratic function, is it not? So, that is what we did yesterday.

(Refer Slide Time: 08:40)



$$\hat{d}(n) = P_{W_n}\underline{d}(n) = \sum_{i=1} c_i(n)\,\underline{x}_i(n) \equiv X(n)\,\underline{c}(n)$$

At index $(n+1)$

$$W_{n+1} = \text{span}\{\underline{x}_1(n+1), \cdots, \underline{x}_q(n+1)\}$$

$$\underline{\hat{d}}(n+1) = P_{W_{n+1}}\underline{d}(n+1) = \begin{bmatrix} \hat{d}_{n+1}(0) \\ \hat{d}_{n+1}(1) \\ \vdots \\ \hat{d}_{n+1}(n+1) \end{bmatrix} \neq \begin{bmatrix} \underline{\hat{d}}(n) \\ \cdots \\ \hat{d}_{n+1}(n+1) \end{bmatrix}$$

$$\hat{d}_n(n) = \underline{\hat{a}}^t(n)\,\underline{\pi}(n)$$

That is we considered the space W n, remember, W n span of I already defined what is x 1 n. It starts at n equal to 0, x 1 0, x 1 1 dot dot dot up to x 1 n, n plus 1 length dot dot dot x q n. I want to find out this vector d cap n estimate of d which is the orthogonal projection of d n on this space. So, I can consider P w n is orthogonal projection acquired for this space. That working on d n; that is what I want to find out as a linear combination of these elements; that is as this is what I want to find out or equivalently x n c n, this is what you have to find out, okay .

Now see one thing. Suppose now at index n, i was this. At this d cap n is important to see this notation; d cap n if I have to write it term wise, you should write like this. You will see this is of i very soon d cap n 0; they have to have the subscript. D cap n 1 you will see very soon; that is why I am doing this elaborately. Why? Because suppose you are now going in time index n plus 1 that is W n plus 1. This is at n at index n and now at index n plus 1 and, obviously, at index n plus 1 my new coefficients will not be c i n, but I will be calling them c i n plus 1, because they will be obtained by taking data up to the index n plus 1.

And, obviously, there will be better estimate than better estimate of the optimal coefficients than c i n’s because I took sum more data in the averaging process, is it not?

Say as time progresses, they will be better and better; in that there is no doubt, because you are doing better and better averaging. Epsilon n square will become closer and still closer estimate of epsilon square as n becomes larger and still larger. As a result, these coefficients will be better and better estimate of the original optimal c i's, c i caps, okay; that is fine.

But suppose that index n plus 1 you want to do the same thing to obtain the better estimate c i cap or sorry c i n plus 1, say, c i n plus 1; that means what you have to do? That time your this entire thing will be span of all vectors of length n plus 2 now, and you want to find out d cap n plus 1 as P w n plus 1 d n plus 1. But you understand that this vector even though, its length is n plus 2, and this is length n plus 1; the zero th entry here will not be same as the zero th entry here. Please see this; this is crucial.

First entry here will not be first entry here. How? Because here I took these columns from 0 to n, linearly combine, took the error, minimize that epsilon n square. I got some solution of c 1 into c q n and then combined them I got these. Next time my sum of square x expression is different, because another term is coming last row having x 1 n plus 1, x 2 n plus 1, x q n plus 1. So, total sum of square expressions is defined now, epsilon n plus 1 square, not epsilon n square. So, epsilon n plus 1 square is minimized with respect to the coefficients to be you are likely to get new set of coefficients.

If you combine this column by the new set of coefficient, the resulting vector will be all together different from what we had, you understand; that is why I am using this notation this. This time it will be d cap n plus 1 0, are you getting me? Please see that; this is crucial d cap n plus 1 1 dot dot dot d cap n plus 1 n plus 1. So, this element, this is not equal to d cap n and the last entry is this clear? This is not that this part, this part is same as this part, because the combiner coefficients have changed. Earlier you took this space; data was from 0 to n for every vector.

You combine them, count the error, minimize the norm square of the error, norm square of the error as per the epsilon n square; you got one solution for the coefficients. Combine these columns by those coefficients; you will one particular d cap n, this is this. Next time we got longer vectors; new data has come up. Again you combine with a new set of coefficients, take the error; take the error norm square. We call it epsilon n plus 1 square, minimize it with respect to the coefficients. Mathematically, you are not guaranteed to get the same set of coefficients as before. It is a new epsilon n square 1 function, is it not? One extra some square term has come up. So, more or less, it has to be minimized, okay.

So, that mean, in general, you get a new set of coefficients; by that if you combine them, naturally you will not get these elements in the first positions, first n plus 1 positions of this vector. You will get totally new value terms, okay. Please understand this as usual, because this is very important here in the development of this Levinson Algorithm and all. Another thing is I will be always interested in finding out projection; projection is like a filter output. Only at the current index; that means for this vector if by anything x is the current in this, this component is the vector estimate.

At any n plus one th index, I got a better estimate in the form of d cap n plus 1, but the total vector is not required, because this is already passed; this is already passed, I cannot go back there. I am standing at index n plus 1; I want to find out that filter output at n plus 1 at index only. So, my quantity of interest will be the last quantity always, are you getting me? I am interested; please see the inner mechanism here. I am interested in finding out filter output; filter output means combiner output. Straight line of filtering because this x 1 n; it is not x n vector, x n minus 1 vector and all that, x 1 n, x 2 n up to x q n.

So, I am combining them real time vector I am calling combiner, but my combiner is giving me. I am trying to find out the combined output at each index of time; at nth index, what the optimum combiner is doing? It is doing this projection operation firstly. Theoretically, what is happening? It is linearly combining the columns, finding out the error, minimizing the norm square of the error. Norm square of the error is called epsilon n square, minimizing that; you are getting an optimal set. Putting that as a combiner, you are combining only the last entries of each vector and getting this guy, because that is my current index.

First, this 0 to n minus 1 is in the entry, no consequence now, because that is past. I am only interested in finding out the estimate at current index, okay. Even though this terms are now better than they were earlier, because better combiner coefficients, more accurate combiner coefficients, more closer coefficients; coefficients were more closer to the optimal ones have come up, and therefore this would have been better, okay. But still

I am not bothered about; I am only bothered about the current one, okay. I define a vector called pining vector pi n 0 0 0 0 0 and only 1 n plus 1 into 1.

What is pi n? N plus 1 into 1; that means, I will be interested in this coefficients d cap n n which is transpose n pi n, very simple; last element we picked them, but this pi n is very important. See the structure of pi n, it is 0 and then 1; it just picks up the last guy. Have you understood this? That by resulting vector will be with the better coefficient resulting vector will be the d cap n plus 1 is a better estimate of d n, where this one will be giving a better estimate; this fellow will be a better estimate of d 0. This will be a better estimate of d 1, because I got better value I mean better estimate of those combiner coefficients; they are more closer to the optimal ones than earlier.

But I am always interested in finding out the combined output in the current index only, because whatever past index cannot be revived any more is it not? Am bringing to real time, okay; I give the d vector. So, that is what. What we will do in the recursive business? That you are be giving the c vector c coefficients c 1 n, c 2 n, dot dot dot c q n for nth index. And new set of data d n plus 1, x 1 n plus 1 dot dot dot x q n plus 1. Using them by simple computation without doing all these elaborate concept, just simple recursion approach; you get this c 1 n plus 1, c 2 n plus 1, this is the entire business.

And exactly without any error without bringing any statistics any operator anywhere; just using algebra, okay; that is the thing that is the entire approach. And you understand this journey will take us to the optimal ones c vector n, then c vector n plus 1, then c vector n plus 2 as more and more data is used in computing epsilon n square. It will be a better and better estimate of original epsilon square, and therefore, this coefficient sequence will march to, finally, the optimal ones exactly not it mean unlike LMS, okay.

And no searching business will march very fast. So, this is what it is. So, that means entire thing you can say at index n. Situation is like this. Suppose this is say x 1 n, you are multiplying by c 1 n dot dot dot x q n multiplying by c q n adding dot dot dot adding, what you are getting is the n th index data; x 1 n, x q n, what you are getting is this last fellow, and what coefficients? Coefficients obtained by using data up to nth index only; those coefficients are called c 1 n, c 2 n, c 3 n, like that.

Those have been put here, you understand; please see the figure. This is the situation I am writing, and this guy is the latest guy d cap n n. This n indicates I have taken data up to nth index; based on that, found now the combiner coefficients, then combining the stuffs using them and the current index n getting this. This is the last guy. Then at n plus 1 nth index, this values change into the better estimate; that is the changing mechanism is the recursive as per algorithm we will be deriving, but what it will be doing then again? That time the new data will be x 1 n plus 1.

I am doing all this elaborately so that there is no confusion prevailing you know, and I am using the latest this c 1 sorry not c 1, and this is your this guy; d cap I have taken data up to n plus 1 nth index. Based on this, found out the coefficients and the current estimate, because I am combining the last elements of each of the vector by those coefficients, and I am getting this. So, this will be the mechanism. They will be updated; this coefficient is updated by a recursive algorithm which will derive into this, but always it will be acting on the current data only. So, the current estimate, okay.

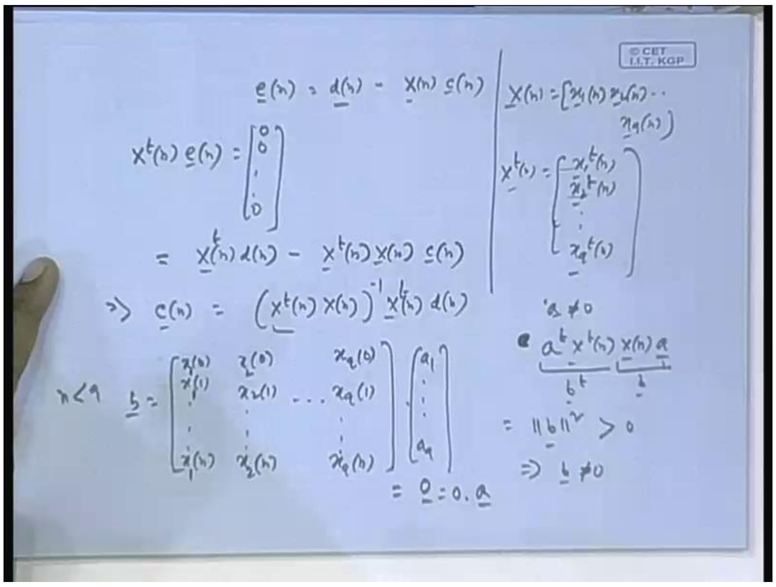
Current estimate based on what I have taken? Data up to nth index from the entire vector d cap, then the last component. Here again I have taken data up to n plus 1; I have computed entire vector. So, those are also could be obtainable; better estimates of d 0, better estimates of d 1 than obtained here, here. But in practice, I am not bothered about them, because I cannot be non-causal. So, I can only use a new set of combiner coefficients to work on the current set of that and get me this. This is the schema thing mainly.

It will come in an indirect way, because I will be doing again linear prediction of all that, but basic option is this; you understood the estimation mechanism here. Here it is more general if it is a filter x 1 n, then there would be a delay here, then that x 2 n there will be a delay here, x 3 n dot dot dot, and finally, a delay and this, same here. Then that becomes a fire filter; coefficients will be adopted by that recursive list for algorithm to the new set, okay. So, you understood the philosophy involved here.

I have to do that inner model because it is not easy to conceive, okay, and this pi n definition; you do not forget. It is a pointing vector, fix up the last guy. If it is pi n plus 1, it will go up to n plus 1, n plus 1; I mean n plus 2 th position will be 1 and okay. There is one notion; I am just discussing various notions, okay. Another thing which I was doing it was end of yesterday’s class and i was just running through actually. So, I have to redo that. That suppose I now want to decide this to explain the framework actually the basic dynamics of the mechanism of the estimation, okay.

But for any particular index n only, I want to find out this coefficient c n. What should be c n so that this vector is in the orthogonal projection, so that the error is having the minimum norm square usual in the given norm square sets in this r n plus 1 space, and therefore, that is orthogonal to each of this x 1 n to x q n again using the inner product for the r n plus 1 vector space, okay. So, that was this.

(Refer Slide Time: 23:58)



© CET I.I.T. KGP

$$\underline{e}(n) = \underline{d}(n) - \underline{X}(n)\,\underline{c}(n)$$

$$X^t(n)\,\underline{e}(n) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$= \underline{X}^t(n)\,d(n) - \underline{X}^t(n)\,\underline{X}(n)\,\underline{c}(n)$$

$$\Rightarrow \underline{c}(n) = \left(\underline{X}^t(n)\,X(n)\right)^{-1}\underline{X}^t(n)\,d(n)$$

$$n<q \quad \underline{b} = \begin{bmatrix} x_1(0) & x_2(0) & & x_q(0) \\ x_1(1) & x_2(1) & \dots & x_q(1) \\ \vdots & \vdots & & \vdots \\ x_1(n) & x_2(n) & & x_q(n) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_q \end{bmatrix} = \underline{0} = 0.\underline{a}$$

$$\underline{X}(n) = [\underline{x}_1(n)\,\underline{x}_2(n) \cdots \underline{x}_q(n)]$$

$$\underline{X}^t(n) = \begin{bmatrix} \underline{x}_1^t(n) \\ \underline{x}_2^t(n) \\ \vdots \\ \underline{x}_q^t(n) \end{bmatrix}$$

$$\underline{a} \neq 0$$

$$\underbrace{\underline{a}^t \underline{X}^t(n)}_{\underline{b}^t}\,\underbrace{\underline{X}(n)\,\underline{a}}_{\underline{b}} = \|\underline{b}\|^2 > 0 \Rightarrow \underline{b} \neq 0$$

That if i do I know that if I have e n vector error vector that was original d n vector minus x n matrix into c n vector; we forget this is x n matrix, is it not; x 1 all this column vector are present; you are multiplying by c n vector. So, these combiners are these coefficients are linear combing the columns that is the projection. There is a projection original error vector, okay. You have to minimize this norm square of this error vector, is it not, and that will give raised to and from that you will find out c n and that with those optimal c n's this error vector will be the this will be individual projection; this will be projection error, and this error will be orthogonal to each component each column of x n, because columns of x n are linearly combined x 1 n, x 2 n, up to x q n, right.

So, that means if I do x transpose n e n, what is x transpose n and e n? X transpose n means what was the x n? X n was this. X 1 n I am repeating here, x 2 n dot dot dot x q n; that was your x n, column, column, column, okay. Now x transpose means just they become rows; this column will become row. So, it will become x transpose n. Second column will become second row x 2 transpose n; it is a simple thing, right. You can visualize it easily. First column becomes first row, second column becomes second row, last column becomes last row.

So, x transpose n times e n means this row times e n, then the inner product between e n and x 1 n vector; that is 0, because e n is orthogonal to x 1 n, then x 2 transpose n e n. That also is 0, because e n is orthogonal to x 2 n dot dot dot dot dot, is it not? This will become 0, 0, dot dot dot dot 0, but e n you substitute by this x n d n minus; that means what is c n? Provided the inverse exists; okay, provided this inverse exist. When does the inverse exist? Let us see this first; this is very important.

First look at this matrix x transpose x. This is the Hermitian matrix very easily, because if you take the transpose, you get this back x transpose x again Hermitian matrix. No conjugation here, because assume nothing is complex; it is a Hermitian matrix. Further, I said not only Hermitian, it is a non-negative definite matrix as such, because any non zero; see this c and this c are not same, or if you want, you can call a. A non 0 vector, any a transpose x transpose n x n a; this will always be non-negative. It is because x n a if you call it b after all this is a column vector matrix into column. So, column vector, then this is b transpose; this is b transpose, is it not; so, we can transpose it.

So, this is equal to. So, this cannot be 0 which means it is non-negative definite, but non-negative definite means Hermitian. Firstly, Hermitian means Eigen values are all real. Non-negative definite means Eigen values are not only real, they are greater than equal to zero, but it is equal to 0; determinant will be zero. So, not invariable, but if it is positive definite, Eigen values should be all positive; so, this must be strictly greater than 0; this must be greater than 0. When will that be? Norm square is greater than 0 if the vector is nonzero implies b must be nonzero; that is you should not be able to find any a nonzero a, so that x n a is a 0 vector; please see this.

Are you following me? This is norm square of b; this is always greater than equal to zero, but I want it to be greater than 0, because if it is greater than 0; definitely I am guaranteed, you can say why are you bothered about sufficient condition. If it is greater than 0, fine; I will answer to that later also, but if it is greater than 0, I am through, because then all Eigen values will be strictly greater than 0. Non-negative different means greater than equal to zero, but if it is greater than 0, Eigen values will be that we have proved.

For positive definite matrix, all Eigen values are strictly positive, then there is no problem. No Eigen value is 0; determinant will be positive will be nonzero, is it not. So, that will be positive. But these norms square greater than equal to 0 means you know the definition norm square. Norm square is always greater than equal to 0, and it is 0 only if the vector is 0 vector. That means if it is greater than zero means this b fellow should never be equal to 0 vector. Instead of this, this is greater than 0.

That means you should not be able to find out any nonzero a, so that x n is a 0 vector. Then you are guaranteed to that. Then the question comes suppose I do not find? If I just prove that I cannot find a nonzero a so that x and a is a zero vector, then it is fine; we are through. But it is like a sufficient condition, are you following me what I did here? X n into a must not be 0 vector, then we are through; that is what I am saying. If it is not if x n into a for nonzero a is not a zero vector, then you are guaranteed to have this positive definite it is.

So, it is sufficient but it is also necessary rather. That is suppose it is not that also you can see. Anyway x n into a, what should be x n? So, therefore, any nonzero a, you cannot find the nonzero a vector, so that x n a is a zero vector. What is x n? You have to find, hold on. These times our vector a a 1 dot dot dot a q and not all the a 1 to a 2 is 0, because a is a nonzero vector; that is primary. Even for any such a, where a is not a zero vector. When can I say that this product will definitely be a nonzero vector? If the columns are linearly independent, then if you combine them if you do this product means a1 times first column, a 2 times second column dot dot dot plus a 2 times last column. You equate that to 0 vector.

Then the only solution would be a 1 to a q 0, but that is ruled out. So, that means if you do not have that satisfied that a 1 to a q, you do not allow to be 0 altogether, then if they are linearly independent, then linear combination cannot be a 0 vector. So, that means this column should be linearly independent, because there should not be any linear relation between them. Can there be any linear relation between them? See the difficulty in RLS. If suppose number of row here n is less than q; q is the number of columns, you have to start at n equal to 0, then equal to 1, then equal 2 dot dot dot dot at some point only at n equal to q only we will cross q, then only we will go ahead, is it not?

Before you reach n equal to q, you have number of rows less than number of columns, dimension of the column space is I mean n; I mean less than q. Suppose we are here, number of row is less than q, but dimension is what? Dimension is n only, is it not? Length n plus 1 vector, so dimensional is n. So, n plus 1; suppose n plus 1 is less than q. So, you have got dimension of the space vector space; you are taking the vectors of length n plus 1; dimensional is n plus 1, but that is less than q. So, you have got more elements here than the dimension.

So, obviously, that cannot be a linearly independent set in a vector space of dimension, say, p, you cannot have p plus 1 linearly independent vector. If you have p plus 1, there has to be dependence relation, is it not; otherwise, it violates the thing, no. This we have discussed at length in a vector space of say dimension p, we cannot have p plus 1 or p plus 2 linearly independent vectors. Remember, I did that beta in terms of gamma and gamma in terms of beta while proving the dimension thing, remember no, that alpha 1, then i constant alpha 2, then alpha 3 dot dot dot and that process start meets at, say, n.

And next time we start at beta, beta 1, beta 2 at m; m cannot be greater than n, n cannot be greater than m; we have done it or not, is it not? So, that means if you number of row is less than q; obviously, some columns will be expressible as a linear combination of the other. I can choose a 1 to a q accordingly, so that total matrix into that vector product is 0, okay. You understand and immediately, what happens actually if this product is zero. If it is nonzero, I have told you here if it is nonzero if they are linear independent and a is nonzero, this is, obviously, nonzero and then we are through.

Question is if it is 0, then what happens you understand. Initially I wanted this, but late this b vector be nonzero vector. Question is how can I ensure that given a nonzero a, then I worked out this elaborate matrix into vector. I said that if these columns are linearly independent given a nonzero a vector if they are linearly independent, then linear combiner; you are just combining them. It cannot be zero, because if they are linearly independent, the only way I mean the combined thing becomes 0 is by choosing this each a 1 to a q equal to zero, but that we are ruling out; that means the combined output cannot be 0, okay.

So, there I said that if they are really linearly independent, I am through, because if this is nonzero vector; obviously, norm of b square greater than 0, it is strictly positive definite. All Eigen values are positive, I am through; these are then nonzero, okay. So, then I am through, but if they are not linearly independent, and therefore, if we do with nonzero a, I can make this product 0, then am I a loser? That is the question. If it is not if x into a vector that is b vector is nonzero, I begin it; understand the mathematics. If x into this vector which is called b; if b is guaranteed to be nonzero, then I gain, because, obviously, norm square of b is greater than 0 positive definite all Eigen value is positive determinant nonzero. So, I am through, everything fine.

But suppose on the other hand, they are not linearly independent, they are dependent, because of that phenomenon that number of row is less than q plus 1 I mean less than q, okay. And therefore, even with nonzero a I get a 0 b vector; in that case, do I lose? Answer is I do, because if you can find out a nonzero a vector, so that this matrix type a is 0, 0 no, because I am trying to see the case when B equal to 0; if they are dependent, then only it is possible.

If this into A is 0, then my point is then I am a definite loser, how, that is the question. If it is strictly not 0 like if they are linearly independent, combined, definitely it is not a 0 vector, because a is nonzero. And then if it is nonzero vector, this is total is b. If b is nonzero vector, suppose it is non zero; obviously, norm square of b positive, positive definite I am through I am a definite gainer. On the other hand, if it is really 0; that is if they are dependent, you can find out nonzero a, so that even if you multiply the two, you get a 0 vector, then what happens? Am I a definite loser, answer is yes?

Because what is this equation? This equation suggests that a vector is an Eigen vector with 0 Eigen value, which means amongst various Eigen values of this matrix that is one with zero. So, I am a definite loser. So, this is not good; this is not allowed. Are you understanding the mathematics my mathematical logic. If it is strictly not equal to 0, fine, but is that really necessary? Question is even if it is 0, can I still survive? Answer is no. The moment if it is equal to 0, it means Eigen value is 0, one of the Eigen value. Eigen value 0 means determinant 0, okay. When you do math, think like a mathematician; I mean use mathematician logic, okay.

So, that means I now make the assumption that I am standing at a index n which is much higher much larger than q. At that time only I am doing all these things, and you can say that, okay, fine. If you want to do it recursively, you have to start at n equal to 0, n equal to 1, n equal to dot dot dot n equal to q minus 5, n equal to q minus 4, n equal to q minus 3; what you will do at that time? Recursively that time number of row is less than number of column; Ii will do that I will manage it somehow. All those things will be built in on recursively algorithms which you never dealt with in LMS case.

Because we are doing pure linear level now we will be doing, okay; see these intricacies, fine. So, under that assumption that is in data positive definite matrix that is I am standing at a index n which is higher than q n plus 1 number of rows; n plus 1 should be at least equal to q or greater than q. N plus 1 equal to q means n should be q minus 1. So, I am standing at an index n which is either q minus 1 or higher, and there is no other linear relationship.

Even if suppose number of row is much higher, you have to add that additional thing also that there is no other linear relationship by some extra condition extra equation linking the columns. Then of course, everything is finished; I cannot do anything, okay. So, under that case, this is inverse; this inverse exists, this part. So, what was our equation? We tried to find out c n such that x n c n d n minus that that; that error norm square is minimized. X n internal could be such that number of row is much higher than number of columns.

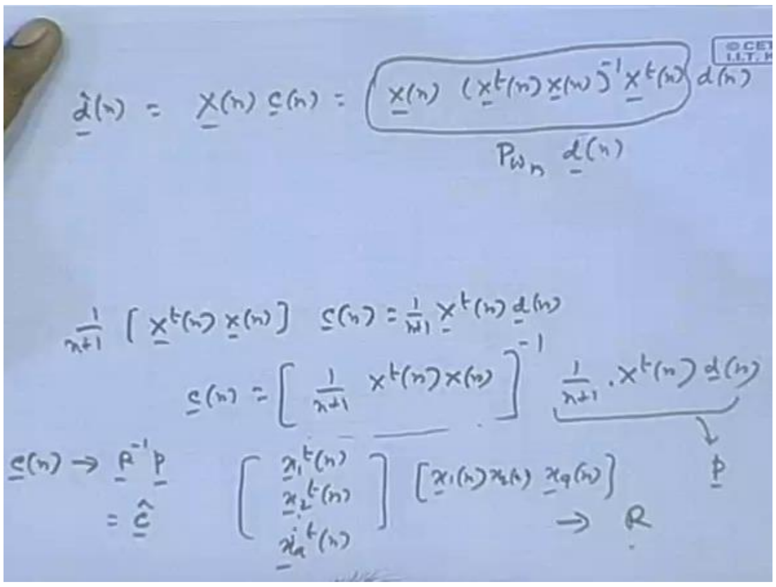
If d n is already present in the column space of x n; that is in the space spend by the columns, then you will get a linear combiner so that x n into c n exactly equal to d n, and error is 0; otherwise if d n is outside the column space of this matrix, then you will get that combiner coefficient which will give minimum error norm. This is a matrix working on d n is called least square inverse or more popularly pseudo inverse or the matrix x n. When this inverse does not exists, then they go one step ahead. They invoke something called SVD singular value decomposition and get this done, okay.

You will come across these terms in I mean if you have any further studies in communication control signal processing, we will come across SVD; we rule architectures for SVD values it carries and all too composite pseudo inverse. So, you know just see this pseudo inverse thing; what they need? All these always which I do that suppose you have given me data up to d n. I have to find out this pseudo inverse, work on d n vector, get me this coefficient. Next time this matrices are different, one more row has come up.

So, new pseudo inverse should come up, you understand. It is very difficult to relate that new pseudo inverse to the original one not so easily. So, new pseudo inverse times the new d n vector will give you the new coefficient. There are recursive algorithms for pseudo inverse this updating this matrix only. That you get me a matrix x n, found out its pseudo inverse; now an extra row has come how will that be pseudo inverse be obtained recursively by some simple computation from the old pseudo inverse. So, you do inverse updating.

There are styles of that also in this Euler's context and then beautiful architectures for doing that. I am still not I am still under periphery of analysis. I am trying to develop the motivation develop inside mind it. I have not started deriving things; I have not yet touched upon the RLS algorithm, how to derive the algorithms, okay, anyway, one more thing.

(Refer Slide Time: 43:57)



$$\hat{\underline{d}}(n) = \underline{X}(n)\,\underline{c}(n) = \underbrace{\boxed{\underline{X}(n)\,(\underline{X}^t(n)\underline{X}(n))^{-1}\underline{X}^t(n)}}_{P_{W_n}}\,\underbrace{d(n)}_{\underline{d}(n)}$$

$$\frac{1}{n+1}\left[\underline{X}^t(n)\,\underline{X}(n)\right]\underline{c}(n) = \frac{1}{n+1}\underline{X}^t(n)\,\underline{d}(n)$$

$$\underline{c}(n) = \left[\frac{1}{n+1}X^t(n)X(n)\right]^{-1}\underbrace{\frac{1}{n+1}\cdot X^t(n)\,\underline{d}(n)}_{\downarrow\ \underline{p}}$$

$$\underline{c}(n) \rightarrow \underline{R}^{-1}\underline{p} = \hat{\underline{c}}$$

$$\begin{bmatrix} \underline{x}_1^t(n) \\ \underline{x}_2^t(n) \\ \vdots \\ \underline{x}_n^t(n) \end{bmatrix}\left[\underline{x}_1(n)\,x_2(n)\ \underline{x}_q(n)\right] \rightarrow R$$

So, this equation I equated to 0; this is equal to 0 or I had that c n is equal to x transpose n d n; this is what I was doing yesterday. Suppose, I divide this by 1 by n plus 1, this also by n plus 1; in that case, same c n can also be written as like this 1 by n plus 1, okay. What is this matrix? For large n, you can see easily x transpose n x n; x transpose n x n means x 1 transpose n, x 2 transpose n, this and then x 1 transpose x 1. You know this matrix matrix multiplication; first element will be x 1 into x 1 divide by 1 by n plus 1.

So, it will be estimate of the variance of x 1, then x 1 transpose x 2 n. It will be you are multiplying the two, averaging by n plus 1. So, it will be a correlation; good estimate of the correlation between x 1 and x 2 dot dot dot between x 1 and x q, then again x 2 transpose x 1. X 1 transpose x 2, x 2 transpose x 1; they are same, divide by n plus 1 again will be a good estimate of the same correlation x 1 and x 2. So, this will give approach R matrix for large n. This fellow will approach P matrix P vector easily no x transpose n d n.

So, you see x 1 transpose n d n means transportation between x 1 and d; you are doing the averaging also, is it not? So, you understand this c n tends to what? Finally, R inverse P which is C cap for large n. Convergence is exact and absolute, no convergence in mean and also no search technique, no approximate search technique, no approximation used anywhere will be faster. One more thing we found out c n. So, what is d cap n? D cap n was x n into c n, is it not? By the formula, you are combining x n the columns by these and c n value we have already found out, the pseudo inverse thing comes here.

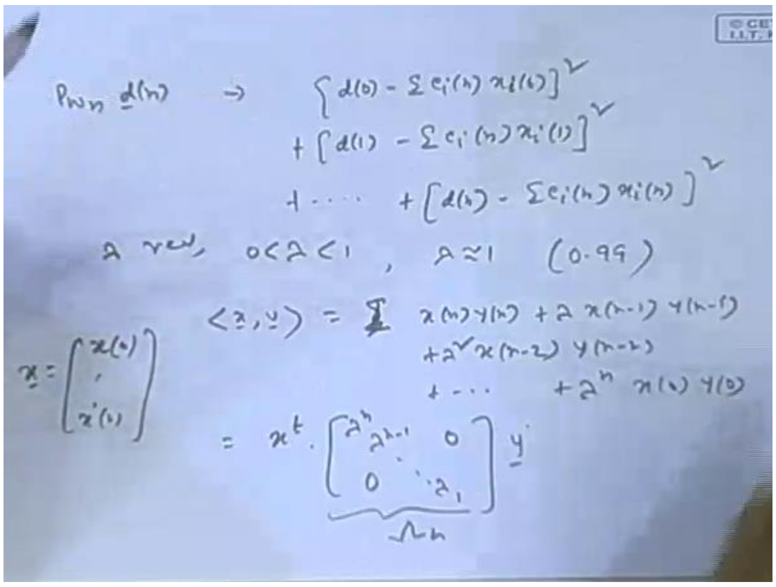
This part is a matrix working on d n; you call that matrix P something that I will complete the notation later P d n; it is come out by identity, d cap n; how do you find the identity? A projecting d n on search plus d cap n; how could be it identity? It is not identity; these are rectangular matrices. What is this? It takes any vector d n, gives it projection on the column space of x n, okay, and it is a linear operator. It is no, matrix times a vector; this is a linear operator.

Instead of d n if you have d 1 n plus d 2 n P into d 1 plus d 2 n means P d 1 plus P d; it is a linear operator. This is the orthogonal projection operator that takes these are mathematical form of that orthogonal position operator that takes a vector d n, unless it has no abstract. Now it has a specific form, because I have got a very specific vector free structure column vectors unlike it is used to be just symbol P operator and all that, no specific form.

But I am now in a specific example of vector space R n plus 1; this is the orthogonal projection thing. Okay, it will be. Okay if x a square, they are linearly independent see it is fine, but x n is in general; n is increasing, no, yeah, okay. One more thing; I am forgetting the things. So, this much you have understood; orthogonal projection, pseudo inverse and all that and also the c n sequence will approach c cap very fast as you go from n to plus 1 to n plus 2 and all that. Now suppose up to index n equal to n naught from n equal to 0 to some index n equal to n naught system input statistics or d n statistics, they are fine; they are constant.

At n equal to n naught, suddenly there is a jump; the statistics changes. R value matrix changes P vector changes. The point is if you are immediately in the vicinity of n 0, n naught plus 1, n naught plus 2, you are sending around that zone. Then what is the sum of square?

(Refer Slide Time: 49:47)



$$P_{W_n} \underline{d}(n) \rightarrow [d(0) - \sum c_i(n) x_i(0)]^2 + [d(1) - \sum c_i(n) x_i(1)]^2 + \cdots + [d(n) - \sum c_i(n) x_i(n)]^2$$

$\lambda$ real, $0 < \lambda < 1$, $\lambda \approx 1$ $(0.99)$

$$\langle \underline{x}, \underline{y} \rangle = \sum x(n) y(n) + \lambda x(n-1) y(n-1) + \lambda^2 x(n-2) y(n-2) + \cdots + \lambda^n x(0) y(0)$$

$$= x^t \cdot \underbrace{\begin{bmatrix} \lambda^n & & 0 \\ & \lambda^{n-1} & \\ & \ddots & \\ 0 & & \lambda, 1 \end{bmatrix}}_{\Lambda_n} \underline{y}$$

$$\underline{x} = \begin{bmatrix} x(n) \\ \vdots \\ x(0) \end{bmatrix}$$

Suppose you are minimizing this, no, d n you are projecting on this W n, is it not? You are doing this projection. For that, what we are doing? You are combining these d 0 minus c i n x i 0 square, then d 1 minus c i n x i 1 square dot dot; this is what you are minimizing. Suppose, you had n is very close to n naught just n naught plus 1, n naught plus 2, something like that. That time this sum of square will not be a good estimate of x valued n square, because the statistics itself has changed, is it not? Statistics has changed.

Earlier, most input statistics were same; d was a random variable; x 1, x 2, x q were random variables. You are trying to find those best c caps c 1 cap, c 2 cap, dot dot dot. So, you took some samples from the sum of square of error and minimized it; for large number of data, they will approach the optimal one. But suppose after some index, the statistics changes, then what ideally I should do? I should try to discard in this summation contribution from index from 0 to n naught or n naught minus 1.

And have more contribution from n naught, n naught plus 1, n naught plus 2, n naught plus 3 like that. At this part, then I will minimize with just to the coefficients, I will get the new set of optimal filter once, is it not? But I cannot detect suddenly that this has happened at n naught. So, I should have some mechanism to forget the remote past and concentrate it more on the current future. There are two approaches; one is called sliding window where into this sum of square you do not take. Here it is infinite wing you know as n increases, it can be 100; it can be 1 million as it goes on with n.

This expose this is growing winds; it is called growing window, you understand this growing window? N equal to 0 we started then n equal to 1; as time moves, the total number of terms goes on increasing and increasing, it is called growing window. We can either make it sliding window. That may be from n equal to 0 to 100; we form a sum of square minimizing. The next time n equal 1 to 101 from the new sum of square, minimize it. So, that way you are using a window only on a finite duration. Okay, even if there is some change of statistics after sliding the window for a while, you will be able to ignore all the past and concentrate only on the current in the sum of square formation.

That will lead to more complicate algorithms there are algorithms. I will be concentrating on what is called exponentially weighted growing wind; it is a growing window, but exponentially weighted. I will be introducing a forgetting factor. Lambda real lambda less than 1 greater than 0, but at the same time, lambda very close to 1 like you know 0.99 you know something, and then I will have the inner product x y modified like this. Current term x n y n, then lambda times x n minus 1, y minus 1, then lambda square times x n minus 2 y n minus 2 dot dot dot dot lambda n times x 0 y 0, okay.

Recent one multiply by 1; next 1 immediate past lambda, next to that lambda square. So, since, lambda is less than 1, but very close to 1, say, 0.99, you understand. This will be suppressed most because lambda to the power n you know is becoming less and less. So, only the current few one fully more dominant in this inner product current few ones will be more dominant; this is not product, you can make it variance. So, the variance term was though only the current samples are more important; it could be error variance. So, in the error variance, you will find an error variance there.

Error variance will be more localized; more focused on the current error samples than the ones at n equal to 0, n equal to 1, n equal 2, okay. First you verify that this indeed is a inner product; it will satisfy all the properties. You can also write it like x transpose then a matrix x is this. Similarly, for y; x transpose lambda n lambda n minus 1 dot dot dot lambda 1 0 0 y, this entire thing. X transpose means x 0 to x n y 0 to y n. So, you understand x 0, y 0 will be multiplied by lambda n; x 1, y 1 will be multiplied by lambda to the power n minus 1, simple and add it. This matrix gives that lambda n.

I will be using this and reformulate, rewrite those expressions of pseudo inverse and all using this tomorrow; are you getting me, clear? If you use this in an inner product, say, and make it variance at that variance of the error, because after all while computing the projection, we will be finding the error vector and minimize variance. In that variance, if you want to forget the contributions of remote past n is equal to 0, n equal to 1 and concentrate mostly on the current ones.

Then these waiting terms will take care of that. In the sum of square, dominating part will be coming more from current ones x n y n or may be lambda times x n minus 1 y n minus 1. If it is variance x n square, then lambda into x n minus 1 square, lambda square into x n minus 2 square, may be few more terms; other terms will be sufficiently suppressed, okay. So, lambda will be part of the algorithm. So, that is all for today.

Thank you very much.