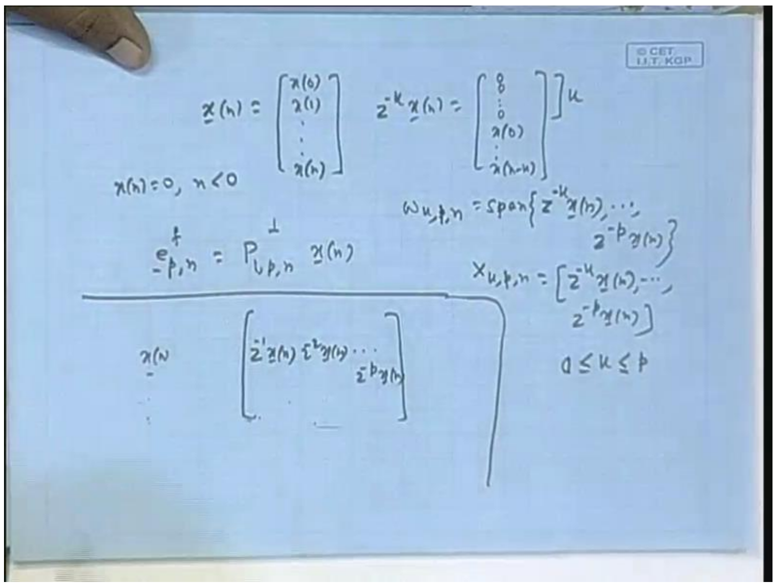
Adaptive Signal Processing

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

Lecture - 31
RLS Lattice Recursions

(Refer Slide Time: 01:10)



We will start with those, start at where we left last time. We defined these data vectors. I will not repeat things today because last time I did too much of repetition. This is my vector Z inverse k x n was 0,0,0 x n minus k and you have got band of k 0 s. I am also assuming x n equal to 0 for n less than 0. This is called pre windowed case. We will be doing, I mean orthogonal projection and all that and the inner product will be what at where exponentially weighted inner product where our factor lambda was introduced; you remember that was done. I will not repeat all those. This quantity, this is called a pth order forward prediction error vector at index n; that is taking data upto index n. That data you tried to project on the space span by past p column vectors; find the error.

So, let me define again W k, p, n as the span of … Also you can define the matrix. If you put the column side by side here, you get the matrix. And obviously k here we are taking k; minimum k is 1; this is at least 1 column. So, Z to the power minus k; Z to the power minus p; k can be minimum 1. In fact, k we can make k equal to 0 also, because that case also you will arise; k equal to 0; k minimum is 0 up to P. The e p, n f is what? P 1 p, n perpendicular x n. What does it mean? You take that as a space W 1,p; 1,p,n. That is the space span by Z inverse x n, Z inverse 2 x n, dot, dot, dot Z inverse P x n. On that you project x n orthogonally and take the error. Just quickly, what does it mean actually?

That just for once I will do this repetition thing; that is you had this thing: Z inverse x Z inverse x n dot dot dot last 1 is Z inverse P x n; these are the columns. You have got x n vector on one hand. You have to find out a set of linear combiner coefficients by which we will combine the columns, take the error, take the square of the error; norm square is not just sum of squares; it is a weighted sum of squares weighted by the factor lambda. That is find the error here. What is the last entry?

x n minus something time x n minus 1 coefficient times x n minus 1. From here x n minus 2; you follow it; this column last entry will be x n minus 1; this column last entry will be x n minus 2; this column last entry will be x n minus P. You are linearly combining them; x n minus that. Similarly, at x n minus 1, difference between x n minus 1 and a combination; linear combination of the past 3 samples. Again, at n minus 2, n minus 3 all that. So, we are finding out those errors squaring up and adding. Normally, it is just sum of square which is a good estimate of that mean square error, prediction error, but here we make it.

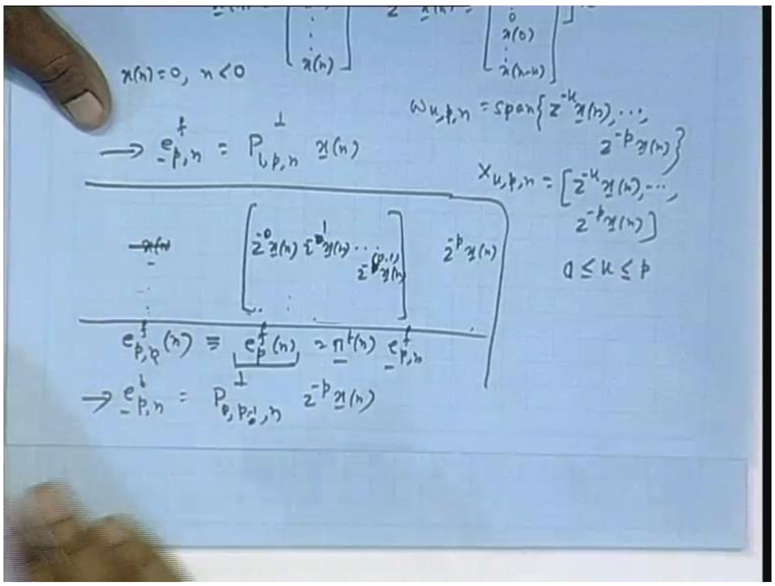
In order to make it adaptive, we make it exponentially weighted; that is last n last error, last component of the error if you call it just square of. Then, last but 1 square of multiplied by lambda, last but 2 error square of multiplied by lambda square, dot, dot, dot, the top most one square of multiplied by lambda to the power n. Add all of them. There is a more general norm expression. When that also for large n gives rise to a good estimate of the mean square error because averaging is taking place mean square error. So, when I do this projection basically these are the steps that are involved.

Here you are not taking just the last entry x n taking at the random variable and combining them in terms of random variable as we did that time previously, whereas random variable is written as a vector; we use the correlation and all that. So, just x n was orthogonally projected on the space span by the random variables x n minus 1 to x n minus P. Here it is not; these are only data samples.

So, first column you know when you say x n it is the data sample for random variable x n minus

1, it is the data sample for a random variable x n minus 2, dot, dot, dot. At index n minus 1,here is data sample for x n. Here is the data sample for I mean one step past I mean the random variable which is you know past by one index, it is passed by two index, like that. Here at various indices you are finding out the error, numerical value, square them up adding with that weighting factor and minimizing that. Those combiner coefficients for large number of rows we will indeed converge on the optimal ones. All this we have seen; this is the meaning of this.

(Refer Slide Time: 07:10)



Another thing is even though we write e p f n, we will be basically interested in the last component of the vector because that is the current component. Current index n, you are combining them. You are using data, past data also so that the sum of square is a better sum of square and you are minimizing that so that you get a better estimate of this combiner coefficient; using them we are combining them.

You get and then take the error. You get a error vector. This is a better estimate of the error vector because you are using the better estimate of the combiner coefficients. But again, in that, I am not any more interested in the past values of that error, forward prediction error; only the current index; so the last component is important. So, I introduce a vector remember p i n; I called it pining vector; p i n was a column vector; all the elements was 0; only last element was 1.

So p i n transpose times any any error vector will be the last component and that we can denote as e p f n n; is it not?

But why use 2 n? We can use our intelligence just to live with 1 n only. This n means what?

We have computed error vector p th order forward prediction error vector by taking data up to nth index; that error vector nth component but since I will be only using the last component, I can as well deal with; that is I will not be using e p f n comma a bracket n minus 2 n minus 3. I will not be using those kinds of things. So, you can make it more compact. Just this I am dropping this index n, but it is all its meaning is there. Whenever I write this, it means I am taking data up to index n, finding out the error vector, then taking the last component that is nothing but pi transpose n e p f n.

Similarly, you have got the backward prediction error vector; that is, first let me write down 0 P P minus 1; sorry in this case, this will start at Z minus 0 1 go up to P minus 1 and instead of this we have got. This vector I have orthogonally projected on the space span by these columns. Last entry will be here what? x n minus P and the last entries of these columns will be what?

x n, x n minus 1, x n minus 2, up to x n minus P plus 1; so P future terms. They are combined; the error is taken. Again at index n minus 1, P future terms are combined, error is taken like that and those errors are squared, summed. These are the weighting factor and then minimized. For large number of samples it will converge on the optimal backward prediction coefficients. So, mathematically this is this: P 0 to P minus 1; 0 to P minus 1 at index n. So, these two components, suppose these are given; these vectors are known; in fact, I will be only bothered about lattice recursion that involves the samples last sample, not the entire vector. But I will derive the recursions first in terms of the vector and then just take the last component. Recursions first will be derived in terms of the total vector and then I am bothered about the last component; just check out the last component; that is how it will be done.

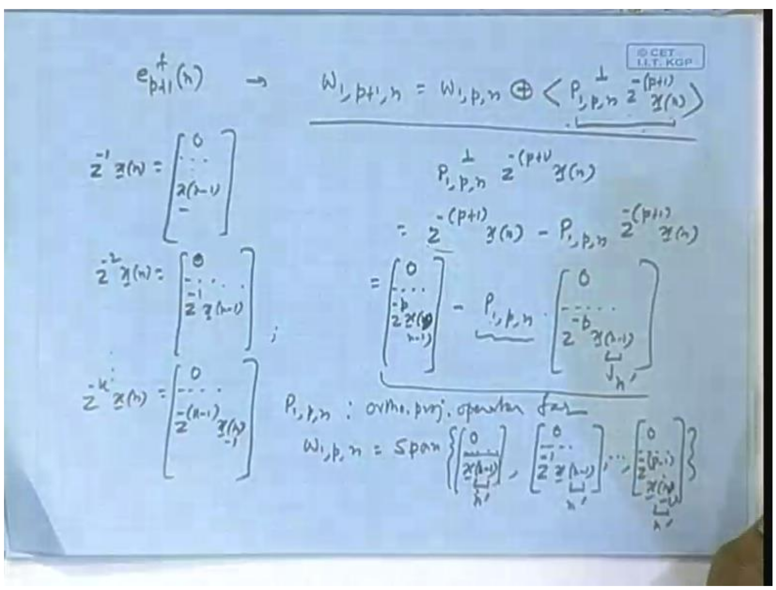
By the way, one thing I mention here. If e p f n you see it is a sequence, error sequence. If I give you e p f n, e p f n minus 1, e p f n minus 2, that is the sequence. This sequence will be meaning what? If e p f n that is how I will be generating the sequence. That means I am taking data up to nth index; then finding out the prediction error vector; then taking the nth component. Next time, I mean if I delay it, the delayed component will be epf n minus 1. Its physical meaning will be what?

I am taking data up to index n minus 1; taking up perdition error vector whose length will be 1 less than the previous one because I am taking data of previous and then taking the last component of that.

Then if I delay it further, e p f n minus 2; that will mean I am taking data that is I am taking that estimation process where I am taking data up to n minus 2 th index; then finding out the projection error vector; then taking its last component. That way, please understand this line of proceeding.

Data is to be taken up to that index and then our estimation is to be part by orthogonal projection, error has to be obtained and then take the last component where we will be generating this sequence: e p f n, e p f n minus 1, e p f n minus 2. I am dropping this index. That is why this confusion might come. This always indicates you are taking data up to which index. Anyway, now you are giving e p f n and e p b n. I want to find out the same for P plus 1th order. Our procedure will be same as we did in the case of stochastic lattice, but everywhere there will be some you know final things which we need to touch accordingly.

(Refer Slide Time: 12:37)



If suppose I want to find out e p plus 1 f n, this I have to find out. for this what is the space? W 1 P plus 1 n; is it not? Here I have W 1 p n Z inverse 1 x n, Z inverse p x n; P plus terms P plus columns; see it that way P plus columns.

Now, I am interested in P plus 1 th order. So, Z inverse 1 up to Z inverse P plus 1; is it not? So, entire that space can be decomposed at W 1 p n direct sum; take the last guy. What is the last guy? Z to the power minus within bracket P plus 1 x n; project that orthogonally on the space W

1 comma p n; take the error. So, that is to start with let us write like that.

Last guy Z to the power minus P plus 1 x n; that I am projecting orthogonally on the space span by what? this pass on this space; is it not? It that was Z. This was a extra guy. This I am projecting orthogonally on this taking the error. So, I am taking the orthogonal projection error operated here for this space. Let us consider this fellow; let us consider this fellow in detail now.

On one hand, this means what? . I will be considering this. Before that, please observe one thing. We know x n; x n definition we know. Z inverse x n, this is what

1, 0 and then x n minus 1 vector I can write; x n x n minus vector starts at x 0, x 1, dot dot dot x n minus 1 and a 0. Z inverse 2 x n is actually 0, 0 then x 0 it goes upto x n minus 2. So, I can write that as 0 and then here Z inverse I can apply. Please see this… So, in general, dot, dot, dot, understood [FL] this is simple. This fact, now you can see here. This means actually the projection error. First let me write it as original vector minus the projection, in that form; original vector is this minus the projection. If you have question, you can ask me. I may be making mistake anywhere; somewhere here. any question?

This is x n minus 1.

Thank you.

x n minus 1. I am writing like this. Now, Z inverse P plus 1 x n means what? Sorry x n minus 1; I am always making that mistake; correct me if I make this mistake. You know I have something in mind while writing something; yes I will do that. Minus again the same vector is here; so I will do the same here also. Same vector is repeated here. Now this corresponds to what? This projection operator corresponds to what? The orthogonal projection operator for which sub space? Sub space span by Z inverse 1 x n to Z inverse p x n.

Remember, P 1, p, n corresponds to what? This is the orthogonal projection operator for W 1, p, n; that is span of what? Z inverse 1 x n first; that I can write as 0; that is one vector. Then, last one how many? Z to the power minus P. These details are you know are not there; X n minus 1, sorry.

So, that means what is this overall projection? This overall projection means this guy is to be projected on this space. So, you have to find out a Set of linear combiners coefficients so that you will linearly combine these columns, take the error between this vector and that, and that error norm square has to be minimized. So, you just square up the individual error terms, add them after weighting them with that; you know weighing them by that by that factor lambda factor and then minimize that. But remember as you combine the first column here by the combiner coefficient, you get 0; subtract from this guy you get 0; so that term when you square up it contributes to nothing.

Are you following me? That contributes to nothing. So what you are then bothered about is to just find out this; I mean these combiner coefficients. This first column first row does not matter. You have to find out the you have to combine them, so that the difference between this guy and the combined output, combined vector norm of that square norm of that difference is 0, minimized.

Now the difference is but that difference vector has always I mean 0 on top, constant, fixed because of this 0 here and 0 is here. So, when you follow them, from the sum of squares what you get? You are as though you are combining these fellows; you are combining these fellows only. These fellows subtracting the combined thing from this fellow and that difference vector you are taking; you are squaring up the error in that each error term in that difference vector, multiply bringing that lambda and then minimizing. But that is what you do.

When do you do that?

Suppose you are standing at index n minus 1. Suppose you are and standing at index n minus 1; n minus 1 is our current index. You can call it even n prime. You call n minus 1 to be n prime. So this is n prime; this is n prime; this is n prime. Forget out that 0 on top; they are not contributing anything. We are finding out optimal that optimal combiner coefficients purely from this lower portion of this column vectors, at this lower portion because you are combining these guys. Now, combining these guys by a set of linear combiner coefficient, subtract that from only this much; that error norm square you minimize you get the optimal phase. By that, you linearly combine you still get back 0, this 0 and this combined 0. This is how they match. Now, when you really are doing this much, you are not considering these and then you are considering these ones. Z inverse p x n prime n prime and you are combining whom?

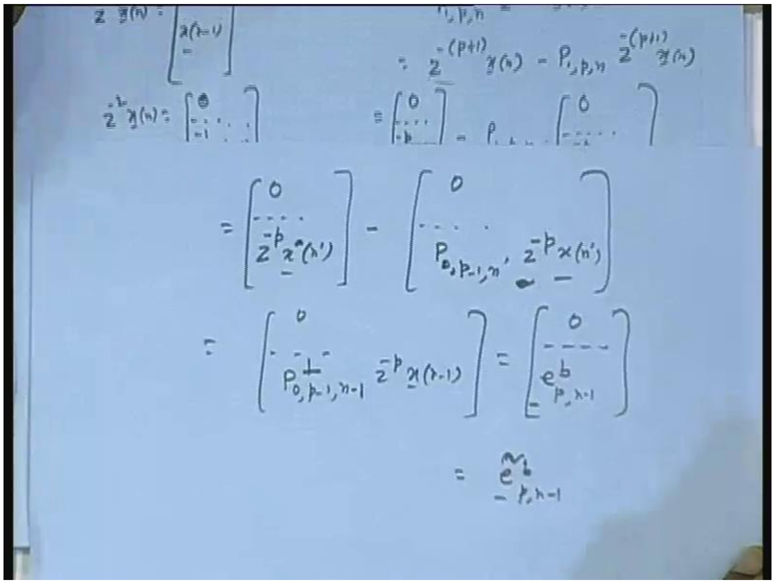
xn prime n n prime is the current index; Z inverse x n prime Z inverse P minus 1 n prime and Z inverse p x n prime. So, pth order backward prediction starting at I mean standing at index n prime; is it not? Are you getting me?

pth order backward prediction because you minimize that error, you are combining this part taking the error and minimizing, you will get that those coefficients which when you put for combining will give rise to backward prediction, pth order backward prediction. It is not as easy as that lattice we did previously. You have to always think about that, that you know optimization involved; projection and computation involved. Is this clear?

You are combining them; you forget about the top, combining them by a set of coefficients taking the difference of that from this; n prime is the current index. Z inverse P xn prime means what? What is the last entry? x n prime minus P.

And what are the entries here? n prime, n prime minus 1 up to n prime minus P plus 1, so on and so forth. So, finding out the errors, squaring up the error, summing them, weighted sum, minimizing; this top fellow you forget. So, that is what you do in that pth order backward prediction; computation just index instead of n its n prime. So, that means what will be this projection? Top will be 0, but this part will give rise to what? pth order backward prediction error vector for index n prime which is n minus 1; that means this thing will give rise to what?

(Refer Slide Time: 24:21)

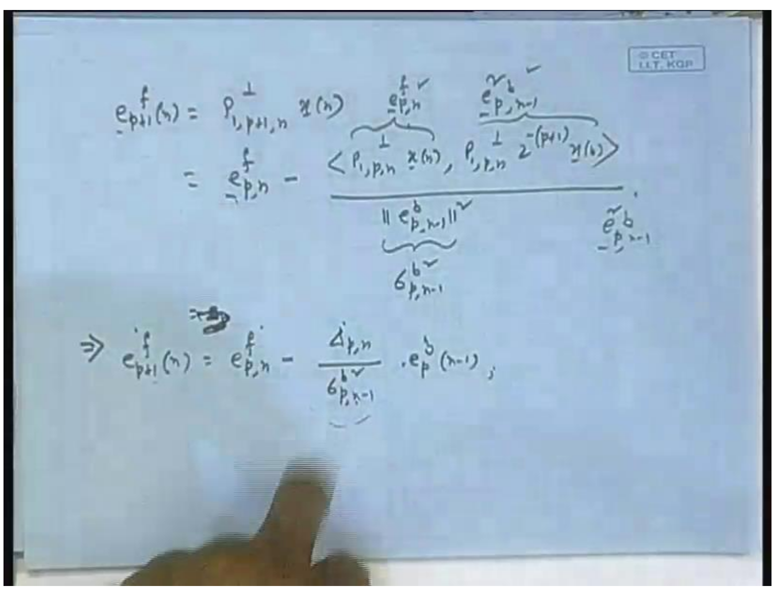


This is as it is you can even call it xn prime, if you want n minus 1 minus this 0 remains as it is because when you combine linearly combine the columns; every column has a 0 on top; so combined output also 0; is it not? That is 0 and before that you get P 0 p P minus 1; thank you very much; you are understanding it; n prime. Now, you subtract 0 minus 0 is 0 and this is this vector minus here also you see this vector minus projection. So, this is the corresponding error. This 0 does not contribute to anything; projecting also you get 0; so 0 minus 0 is 0 and it is Z minus Z to the power minus p x n prime original vector and here is our projection, backward projection. So, that means this becomes simply error.

Now, instead of n prime, I am bringing back n minus 1 and this error is by our definition e P terms P future e into p minus 1. So P backward prediction b index n minus 1. So, that means simply take the pth order backward prediction error at index n minus 1; append a 0 on top. This kind of vectors you know I denote by this way: e tilde b p n minus 1. This means first you remove the tilde, take the vector and then add 1 0 on top; call it. Otherwise every time I have to draw a vector like this. Just to get rid of that hassle, I am just giving it a name. So let us come back here.

We have to find out this P plus 1 th order forward prediction error vector. So, this overall short space was decomposed like this: W 1, P plus 1, n means W 1, p, n and the last guy this fellow has projected orthogonally taken the error but that error is this error is this. So, that means I can now easily derive.

(Refer Slide Time: 27:37)



x n is to be projected orthogonally on this space taking the error; that means project on this taking the error minus projection of these. I am taking error; that is why I am saying; otherwise projection means project here plus project here; then you are computing projection error. So, subtract that from original xn. So, xn minus the projection here; that will give you 1 error vector minus another component as we did in the lattice case. So that will be what? xn minus projection of xn on this. You understand [FL]. I will be projecting x n on this means x n on this past plus here. Subtract that from x n. So, x n minus this projection here means what?

Projection error pth order forward prediction error minus x n and this vector inner product divided by norm square of this vector into this vector . This vector you already found out a name for it; this thing, this fellow case of this kind, remember this.

Now, x n I will not show every step. x n inner product of this over all guy. I can repeat this on x n also. You remember the trick I used; x n inner product of this inner product of this vector. If I am computing this I can repeat this on x n; this operator and this guy. You can also call this that name; if you want you can give it and you can call it that way; this fellow.

And what is this fellow? Forward prediction nth index error e p f n, e p tilde b n minus 1. Remember, it is n minus 1 and tilde; so 0 on top and that backward prediction error vector pth order index n minus 1; that is taking data up to index n minus 1; their inner product divided by norm square of this fellow. When I do norm square of this fellow, let us instead of giving it like this, let us consider this fellow; norm square of this. Norm square means using our weighted inner product definition this entire vector square the last term multiplied by 1, square the last but 1 multiplied by lambda, last but 2 lambda square like that, and last one 0 square multiplied by lambda to the power n, which means nothing.

So, 1 times this fellow its last entry square up; last but one entry of this fellow square up multiplied by lambda, dot dot dot like that. To be simple, the norm square of this fellow (Refer Slide Time: 31:17) norm square of this fellow; this total vector is norm square of this much only; so the 0 into lambda to the power n makes nothing. So when I have norm square of this is simply norm square of this guy. So, simply this and this fellow I will be calling sigma P into this guy; this guy itself inner product by norm square of that into this fellow itself. This overall inner product I will give you the name delta.

Please understand one thing. In the case of lattice, we had, we used stationarity and then we have got a quantity independent of n, and finally by the recursive mechanism we obtain from the correlation; that is how we construct the lattice off line. Here this quantity is purely data dependent; dependent on index n. These are signals. This overall thing is the multiplier. So, lattice filter will be determined by this multiplier coefficient. These are the coefficients. I would say these coefficients are time dependent here. So, we have to have recursive relation so that their values are changed as new and new data come. This is what we did not do in the ordinary lattice case.

That adaptive, gradient adaptive, that is a separate story. You forget that. There are a lot of approximation and some elimination all that; that we are not bothered about. We are doing exact minimization. You know exact least square error minimization.

We would here this quantity is purely depending on n; mind you, I am not making any approximation; no e operator nothing; no stationarity anywhere. This is where norm square by taking data of this vector up to index n; up to index n minus 1. When new data comes I must have the corresponding thing, new norm square. I should have those relations. That is how this, this is a parameter of the filter; some numerator by denominator; numerator also I will call delta P,n depending on n.

In the previous lattice case, stochastic lattice case, everything this inner product was in terms of expectation operator; the entire thing was obtained as in terms of correlations; so offline you construct the lattice once for all; that that is not possible here. I have to have update for this delta. So this quantity is e p and by the way I will not be bothered about this total vector as I told you. I will be taking only the current component.

So, now take the last component. This is giving rise to I should put an index n, then bracket n, but I told you we have the intelligence to understand that you know that there is no need to put n twice. You will know the moment I put n, it will mean I am taking data up to index n. Based on that I am finding out orthogonal projection error and that error vector nth plus component. That is the physical meaning. I am that is I am dropping another index n here; subscript n. So, you take the last component here minus this inner product between this and this, this fellow, and this fellow I am calling it a name; giving a name delta p n.

And here last fellow, you see this last fellow means this vector. Its last component means basically last component of this guy, which is what? Backward prediction error vector where I have taken data upto n minus 1; last component, n minus 1th component of that. So, I should have just n minus 1 after this, but I am dropping this index n minus 1, because I will use only once. But this again means it is an error. How it is obtained?

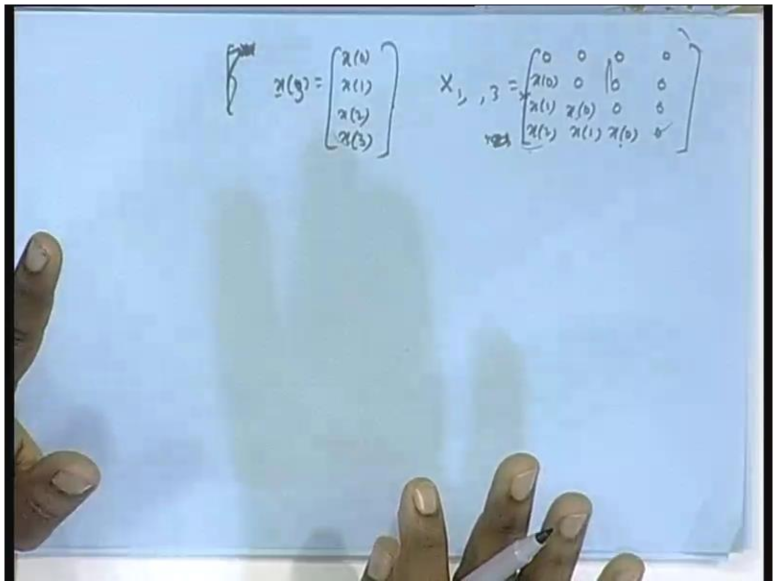
By taking data up to n minus 1; then taking the orthogonal projection, appropriate projection; then taking the projection of vector; then taking the last component. And these quantities are strictly time dependent. At every time index I should have their values. I should add up to ((Refer Time: 35:51)).

I should have recursive relations to obtain this from their past values and I am doing exact minimization; there is no question of approximation no e operator, nothing. Mind you, these are the filtering equation. These are signals only parameter. Let us multiply with this and the parameter needs to be updated in time; time dependent. So, I have to find out exact update relation, exact some recurrence relation n from nth index to n plus 1 th index or at least order update relation; if I know this quantity for pth order, how to go for P plus 1 nth order and all that. One thing there is lot of intricacies here. Again, I come back to this ((Refer Time: 36:36)).

Here you see actually there is a division. It can show up and this may lead to division by 0. How? At sometime of the lattice I have to take care of those steps. Suppose, it so happened that this guy is already contained in the space here, suppose; how I will show all those things you have to be careful. This is content; that means if I take the projection of this guy, this guy I am talking about last guy. This is a Z inverse 1 x n up to Z inverse P plus 1 x n; the last fellow is this Z inverse P plus 1 x n. Suppose this is already content in the this this space. So, if I project it, I will get back this fellow only and the error will be 0.

If this guy is already contained in the space; that means these columns will be linearly dependent because at least one column is contained in the space span by the remaining ones. That is manifesting here in this way; this fellow which is given as which is contained in the space. If you now project it at the ortho projection error will be 0 vector and obviously norm square of that error vector will be 0, and that will manifest here; these are error vectors. So, when you do just go in blindly in this algorithm, you will find out this is standing out to be 0 at sometime. How? When that can that happen, we can see now.

(Refer Slide Time: 38:25)



Consider our matrix. How is our matrix? First one is x n. Suppose I am considering n up to index n equal to 4, so x 0, sorry and I am doing forward prediction; that means x 4 is what? x 0. As an example only I am taking x equal to 4 or x equal to 3 and what is that matrix?

Just a minute. I will put an appropriate figure here. First one is delayed version. Remember x 0. I am assuming x 0 is non zero. Suppose I take up to this, can you see that these columns are linearly independent? If x of 0 is non zero, x 0 is a non zero data, this column, this column, this column, linearly independent because even if you combine these two, you cannot get non zero quantity here. Anyway, if you just this column this column is multiplied by c 1 multiplied by c 2 multiply c 3 add c 1 x 0 plus 0 plus 0 c 1 x 0. If that is 0, c 1 has to be 0. If c 1 0 and c 1 x 1 plus c 2 x 0 plus 0 has to be 0, but c 1 is already 0; so c 2 has to be 0, so on and so forth. This part is no problem. Three terms and this guy is clearly out of it; does not remain in this space because this has x 0 on top; this has 0 s. Even if you combine them you cannot get the non zero here. See you can clearly project it orthogonally on this. You get a projection error which is non zero vector; You have a non zero norm square; all that is fine.

Suppose I have reached this stage where 0 0 0 has come. Up to this was order 1; P equal to 1, P equal to 2, P equal to 3. Suppose I want to go for another at this stage at index n equal to n equal to 3; n equal to 3 at this index I want to go up to another order. So, I will have this.

Now, I want to project this vector on the space span by these 4 columns in that manner, recursive manner. So, I will if I go like that I will say the space span by the 4 columns is the space span by the 3; the direct sum of what? The space spanned by the orthogonal projection error of this vector on the space span by the 3. But this is all zero vector. This is already contained here. So, that error will be 0. Are you following me?

And that is at the error square; the norm square will be 0 and that will immediately benefit; then we will have division by 0 problem. But this is bound to happen because at index, suppose I am doing at an index n equal to 3, if I have done it n equal to 2 you cannot tell me that I cannot go on adding higher orders. As long as your index does not grow sufficiently, that will be terrible; that is not algorithms can work. Even at index 3, I can go up to order 10, order 11, order 12, order 14 order 20 and all 0 0 0, 0 0 0, 0 0 0 columns will come up,

Suppose I want to go maximum up to order 20, so that I have to do for n equal to 0 also, 1 also, 2 also, 3 also, for all n. Remember I told you the number of rows must be greater than or equal to the number of columns; the original condition. But in this data structure, the moment you see you have even this equality because of the 0 s presence, you have this problem coming in. This comes only with respect to backward prediction. Suppose you have got this space, you had to do forward prediction of this ((Refer Time: 42:31)).

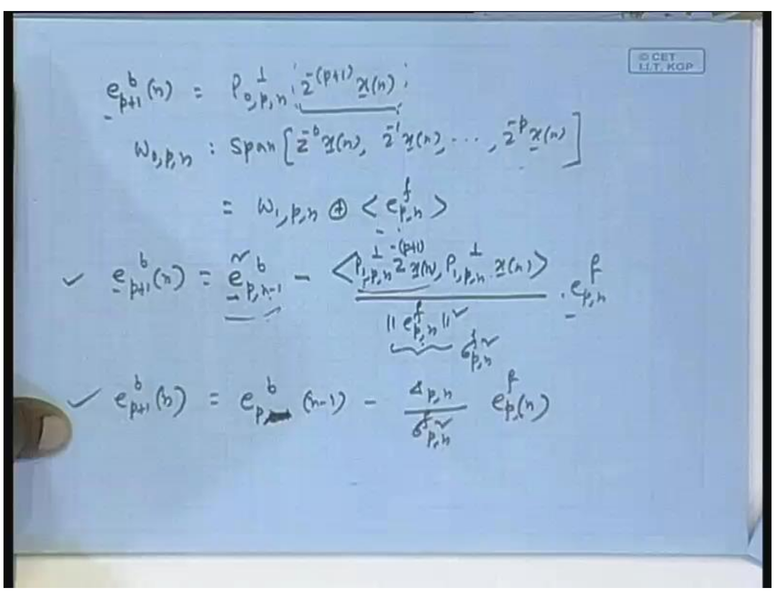
Suppose you have to do pth order, you have to do backward prediction; current vector, forget about this current vector. This column, this column, this column, this column, fourth order; like here also it was fourth order. This was to be projected on space span by the 4 columns. I had obtained prediction for third order as I was going for fourth order. Here also, suppose I want to do fourth order backward prediction current vector, 1 delay, 2 delay, 3 delay, 4 delays. So, this guy has to be projected orthogonally on the space span by this fellow and these 3 fellows. So, there again you can decompose the entire space as what? Space span by this 3 and take this project orthogonally on the space span by this 3; take the error.

There I do not have any problem because it is happily lying outside the space span by 3 always. Because it is non zero, this vector does not lie in the space span by them. So, there I will not have the problem that error vector is a 0 vector is norm square is 0 and division by 0 occurs. This occurs only in the manifesting it is dependent only in the backward prediction error norm. This is bound to happen normally. This is for those indices when numbers of order is larger than index number. There is number of rows is less where the all 0 columns will come up. Are you following me?

That means my algorithm in the end has to have those mechanisms also by which division by 0 is automatically restricted without effecting the algorithm. I am just telling. You know it is dropping back; it is not nowhere near you know. I mean this adaptive, this stochastic lattices are much more trickier, much more challenging. They are the e operator and those are absolute things. You know ideal things correlation and all; they saved us. Here is purely data dependent. There is no question of safety using some you know abstract e notion. This is purely an exact competition.

So, remember this. This you take with a pinch of a salt that this division is possible, dividing only by this is possible; that is we were doing a update only when this is non zero. So, this is for forward prediction error updating; the same thing we can do for the backward prediction. You can quickly do that and mind you I am doing it, like in the case of stochastic lattice I did what? I am doing it up to first order that is from e 1 f n I am going to e 2 f n then e 3 f n and all that, but what is e 0 f n or e 0 b n? 0 th order; that will be obtained separately. Remember there also I started at e 1 f, e 1 n f, e 2 n f, e 3 n f like that. Your starting was e 1 n f or e 1 n b, but I have to get them from e 0 n f, e 0 n b. What is e 0 n f? e 0 means we will directly see; that is what I did there also.

(Refer Slide Time: 45:36)



Now comes this thing - backward prediction thing. This you have to find out and this equal to what? P 0 P plus space here is W 0 p n will be span of Z minus 0 x n is x n. This is same as what? Take the space span of this part that is W 1 p n and this guy Z inverse 0 means nothing; x n. x n projector of this space taken; the error that is obviously e p n f. This is very simple here. This part, span of this part is W 1, P, n and then other part is what? Orthogonally composition was span of what? x n. Z inverse 0 means nothing; x n projected on this sub space span by this fellow's error. So, that is e p f n. We have just now seen first P samples, first P vectors. So, that means what is this quantity?

This guy is to be projected orthogonally and this projected orthogonally on this on this subtracted from this (Refer Slide Time: 47:16). This guy, this fellow has to be projected on this and on this and then that has to be subtracted from this fellow only. If you take this and project on this, we have seen it here, we have seen it there. You will get this component Z to the power minus P plus 1 x n is this because 0 and Z square minus P x n minus 1 which I call n prime, and 1 to P, 1 to P you know is what? Z inverse 1 x n was what? 0; x n minus 1 which is x n prime, dot, dot, dot; this we did here; this projection we did here; you will get this component. When you project this guy on this fellow and take the error minus this guy; inner product with this divided by norm square of this into this ((Refer Time: 48:58)).

But this is what? e p f n in terms of projection is what? P 1, P, n f x n. That operator I can work on that also. After all e p f n is what? e p f n is P 1, P, n x n. This is e p f n. So, this operator I can work on this fellow also. I can repeat divided by norm square of this and once again this is this quantity (Refer Slide Time: 59:56); this quantity and this quantity are same; is it not? P 1 p n this e p this; so these are e p f n. This inner product or in the other recursion what inner product you have got, those two are same; this inner product and this inner product, they are same.

So, the real data a ,b inner product of b a with b or b with a, they are same. Here they are the two terms where e p f n and e p b n minus 1 tilde; here it is e p b n minus 1 tilde. Here this guy e p f n inner product is same. This quantity I will name as sigma p n square. See if I do not take the last component, the last component here, I drop this index.

Here I do not have any problem with this division. I told you. Because of the data structure here this error vector will always have will be non zero error. If you remember the data structure is for the x 0, x 1 up to x n is x n; then Z inverse x n, Z inverse 2 x n, and all that. When you project x n on the space span by these fellows, x n is never converging the space span by those fellows;

those delayed vectors. So, error vector will be always non zero and norm square. Problem was coming in the backward prediction case; the division by 0 thing was coming up. These two are things.

Student: ((Refer Time: 51:57))

Here sigma f; here I forgot this is sigma f here. And you can also update these two; order update. See these are the signals. Signals are no consequence to me. Basically the filter is constructed by these parameters and the parameters are time dependent; so every time I have to update them as new and new data come. These fellows can be order updated. If you know them for p th order, you can go for P plus 1 nth order; the same way as we did earlier but then at the very initial stage, when there is no previous order, that time they have to be updated in a time recursive manner. That is when new data come, at the very first stage, with a new data new norm square will be evaluated and that will be order updated.

Are you following me or not following me? From p th order these two norm square will be updated. You get P plus 1th order value at a same index n. Then you by updated they will be using that updated relation I will derive. You will get the same values for P plus 2 th order, so and so. But you then go back to what? You come to the very first stage of the lattice - 0 th order. That time I cannot go any further stage backward. So, that time those quantities will be what? Will be updated updated with what?

As new data comes and then as you move from n to n plus 1 th index or may be n minus 1 th to nth index and new data has come. They will be using that. We will be updating the norm square at that stage. So, at nth stage norm square, at the very first stage or update and then you update for the same index from the 0 th stage you obtain from the first stage, from first to second, second to third by a recursive formulae; that is order recursion. But at the very first stage it has to be time recursion; from n minus 1 th to nth. As the new data has come, you have to modify the norm square in the 0 th stage using the new data information. Once you update for that stage then propagate it order recursively.

Unfortunately the same cannot be done for the top delta p, n. You cannot get from delta p,n to delta P plus 1 n to delta P plus 2 n. There is a problem. It cannot be done. It has to be time updated and it is far more complicated and more challenging.

How to obtain delta p n plus 1 from delta p n? These are all beautiful linear algorithm; we will do that; I donot think I have time today. So, I will conclude. Thank you very much. It will take at least 2 days; 2 more days to complete the recursions.

Thanks.