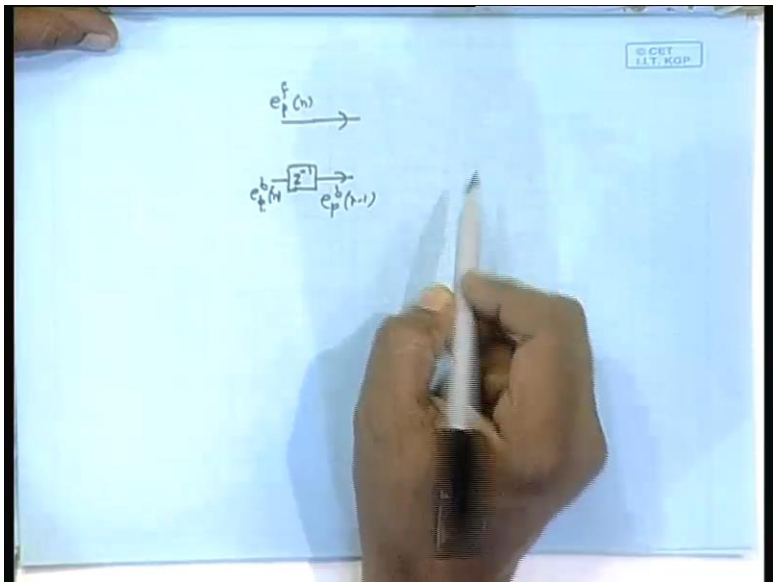
Adaptive Signal Processing
Prof. M. Chakraborty
Department of Electrical and Electronic Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 33
RLS Lattice Algorithm

Important thing of updating that inner product delta that is something we have to take up, that is the most challenging thing.

(Refer Slide Time: 01:13)



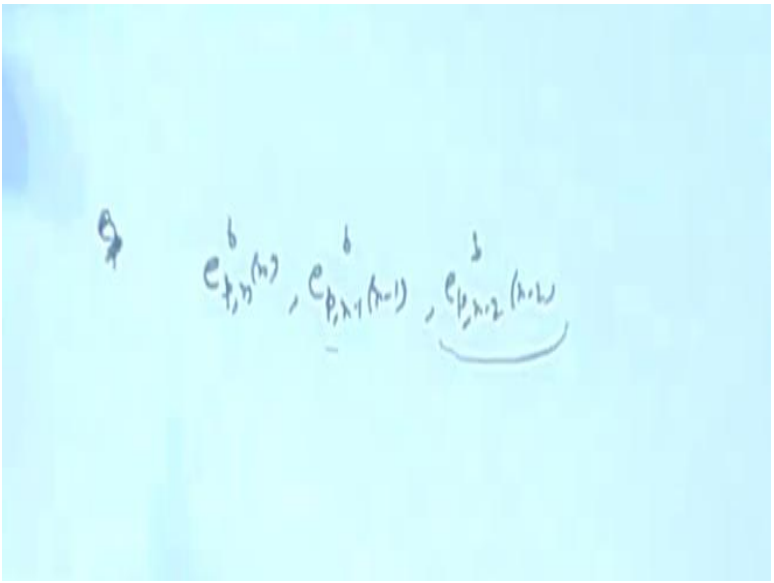
What we have obtained so far, we obtained the lattice; the structure was like this; if I write e p f n actually, you remember there was a subscript (p, n) actually, but then I dropped; ideally the subscript should be there, it means that you take data upto nth index; based on that do the list size minimization that is orthogonal projection, find out the error and that error's last component, but since it is still n only here I am dropping that n, assuming that we do understand the implication.

These 2 signals, this one and this one, that is more important here; this point is e p b n minus 1, but remember, when I write e p b n minus 1, it means not that there is a vector whose nth component is this and n minus 1 th component is this, not at all. This means, I took data upto nth index, found out that e square of minimization, that found out optimal combiner coefficients, combined the columns of that matrix involved, and then found out the error; that error vector, last component.

Here, it is not the last but 1 component of that vector; that vector itself is different because I am taking data upto n minus 1 th index. Then I am doing e square minimization, I get a new different set of combiner coefficients; by them mean if I combine the columns of that data matrix; I get a new, I mean different orthogonal projection, then take the error, last component of that error vector. Do you understand the difference?

It is not that, you know, I mean there is 1 vector whose n th component is this and n minus 1 th component is this. So, I am dropping the subscript, but if I really put the subscript here, e p, n then bracket n, and e p, n minus 1 then bracket n minus 1, then it, I mean, clearly indicates that. Just for saving space and all that I dropped the index n, but then you do not get confused, you understand the real meaning of these two. Why not? I give you, I, this is, keeping this in mind I made one statement; I think in last but, last to last class.

(Refer Slide Time: 03:29)

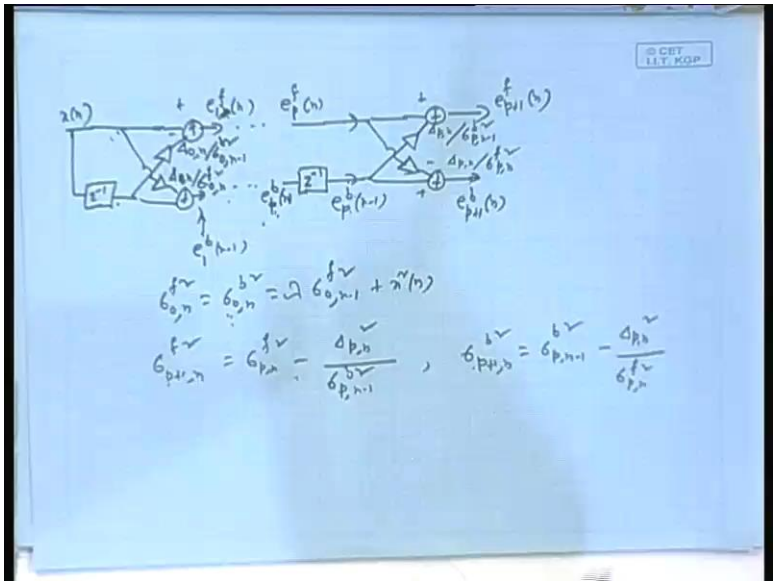


That I will make a sequence this way; this is my sequence – e p n b n, I will, I am not dropping the subscript for the time being which I leave in the previous page; then e p n minus 1, n minus 1, b; e p n minus 2, n minus 2, b; this is my sequence. These are, such a sequence with n minus 2 th element is this, n minus 1 th element is this, nth element is this; this is my sequence. More than you are trying to understand, this is, please understand that; this is the sequence.

Where these values are obtained from what? I took data upto n minus 2 th index, orthogonal projection vector, last component of that vector. Here I took data upto n minus 1 th index, found out orthogonal projection, those 2 projections are not same; here you get a better set of combiner coefficients. Again, find the error, n minus 1 th component, that is the last element of that.

This is my sequence. If this is the sequence, obviously, this is the value of that sequence. This is the element of that sequence at the current index; this was one at the previous index. So, if I take the sequence and then delay it, definitely this will be delayed and you will get this because this is my sequence, mind you; you understand this; this is my sequence.

(Refer Slide Time: 04:58)



And obviously, because from here by the computation, now how do I assure, you can still ask me that how to ensure that that is the sequence generated, fine; I will, by this lattice computation I will generate here e p b, sorry, e p plus 1 b n; and that is based on what, data upto nth index. So, e p b n minus 1 by taking data upto n minus 1 th index, then e p b, e p, sorry, let me complete it, then.

So, you see, this guy and look at this guy, this will be used again in the previous, next stage, is it not? But here what I have done, like here, data upto nth index was considered, here also data upto nth index that will be delayed. That is, if I start with e p b, that is if suppose you give me this thing anything, I can use data up to nth index, found out the

error vector to the last component, then by the computation I am generating the same thing just for p plus 1 th order by the same thing.

So, if I, I am delaying it here by my previously I am getting that 1 step delayed element of that sequence, here also I will get the same. Important thing is this coefficient; coefficients are delta p n by, sigma p n minus 1 square, right, b square, is it not? This one, these you have to remember; e p, I mean that is forward prediction error, then the previous one forward prediction minus this side, and the inner product is common, delta will common, but norm square will be from this guy.

And here it is, these 2 coefficients, dot, dot, dot; and in the first stage, you remember, I did the first stage separately, first stage, because I told that suppose when I derive the lattice I told that I will consider at least p equal to 1 and then go for p equal to 2 p equal to 3, but what happens to p equal to 0 and all that, you know, that I did separately; you remember this I did. And then the rest of that only I defined what is e 0 n f, what is e 0 n b; this your x n; like this, ok.

So, here you get the sequence e 1 n f, sorry, you can put this way; here you get, e 1 b n minus 1 like that; did you understand? Now, you see, so lattice recursion is, I am not writing the equation for this separately; e p plus 1 f n is this minus this times these, and so on and so forth. So, once you know the parameters at any index, be all the coefficients are time dependant, unlike that lattice, here all the coefficients are time dependant.

And, at each index at n when you find out the coefficients then you use them to filter; x n, this e p b f n these are only signals; these are not parameters. Once you know the multipliers and all, at that index of time, whatever signals be used, 1 filter in them, you get various signals. And, how the parameters are treated? Parameters are treated like this; this is the 0 th stage, sigma 0 n f square and sigma 0 n b square, both are same; and that was lambda times plus current 1, is it not?

So, using, at the moment anything index, current index comes and new data has come, you find out this new norm square. Delta is a problem, but suppose by some hook or crook we know all the datas for the time being suppose, delta are supposed are known to us at each index, somehow, then we only need to know this sigma 0 n square and this is sigma 0 b, but n minus 1 square. So, this was obtained in the previous stage and stored.

And, the current one which stored for the usage at n plus 1 th index, current one for this, ok; so using that you find out these ones. Then, we have, then how to obtain the sigma's

for the subsequent stages? See, it becomes adaptive because of, because data comes in, so data determines the 2 norm squares, 2 sigma's here. And, from this sigma squares, in general, recursively I obtain the sigma squares for the subsequent stages by that those recursive equations, sigma. These we have obtained, is it not? And, please correct me if I am making any mistake; this may be sigma p b n minus 1.

So, mind you, this is how it is adaptive because data comes, data changes the norm square at these and those norm squares are used order recursively at the same index to find out the norm square, this variances, for the subsequent stages; using them new, the coefficients are changed; using those values the corresponding predictions errors are computed; in turn those predictions errors will be used to update deltas, that we will see.

Deltas will be updated; it is not possible to order update delta. The mathematical is not allowed; so it is not possible. So, what we will be doing is, for every stage we will be time updating delta that is from the previous value of delta we need some storage there; we need to store the previous index value of delta that will be used to time update, that is to get the current delta p n from delta p n minus 1, but that will be also using the 2 errors available, 2 product the forward prediction and the backward prediction errors available, those all also will be used.

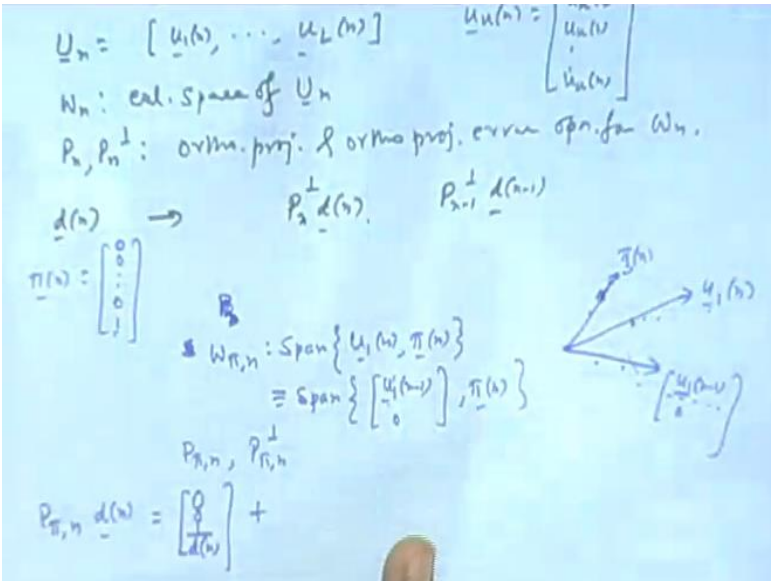
So, then how the adaptation will go on, that at a 0 th stage; so you consider 0 th stage; x n comes, so the 2 variances change. And, deltas will be what? They will be time updated from their previous value plus, the 2 prediction errors. And then by order recursion you go to the next stage, you get the new variances, and there you time update the corresponding delta, but that will be using previous value of delta plus, these 2 prediction errors which are computed at this stage, so on and so forth.

So, it is adaptive, you understand; but, adaptive is exact computation; and as more and more time passes, basically this will become better and better estimate of the actual optimal forward and backward prediction errors because underlining mechanism is that list square minimization. More and more time passes means more and more data comes, and you know the underlying mathematics, you will get better and better estimation, but it is purely adaptive, no approximation anywhere.

You understood that adaptation phenomenon, is it not? Whereas in each index n, I am adapting the coefficients; variances are updated order recursively and using the 2 prediction errors available with us; and the first value of the delta for the particular stage

by some formula we will be calculating current value of delta. So, now our main task remains as to, how to compute the current value, how to time update delta; let us spend some time on that. So, to do that we set aside the lattice thing we come to some general result now. I was just discussing yesterday, but the time was short, so I could not explain properly. So, let us go to some definitions.

(Refer Slide Time: 14:14)



Suppose, we have got some vectors, u 1 n, dot, dot, say, u l n number of vectors, you form a matrix U n; here, any vector here, u k n, is of this kind; u k 0 th index, u k 1 th index, dot, dot, u k current index, normal. W n, column space; there is a space span by these columns; space span by the columns of this matrix, this column vector; there is a column I call it usually column space and that is W n.

P n, P n perpendicular, you understand, they are orthogonal projection and projection error operators, ops, for W n; this space W n, fine; d n is an external vector. Here, I want to find out this estimate, this projection; it could be either P n d n or P n perpendicular d n, because when you have P n d n, e p n perpendicular d n is no problem, you subtract from d n.

So, maybe I write directly this; I want to realize somehow to; mind you, the 2 lengths are different because when I say p n minus 1 perpendicular means, I have got columns like u 1 n minus 1, u 2 n minus 1, u l n minus 1, the last row of this matrix is absent; d n vector also not; the last element is absent, is only d n minus 1 vector; there I do a projection, and here I get a projection.

Of course, its length is 1 higher than here. But even where there the first n minus 1 n 2 is not same in the row because the 2 projections are different, you may define of combiner coefficients. Because here d n and these columns, they have got 1 extra data present, is it not? But, I want to somehow relate this with this, you understand. This is time recursion, n minus 1 to n th, but it is not easy; it is not direct anyway, no way it is direct, it is not easy.

To understand this, suppose I consider this equation. And, you remember, pi n vector, in case you have forgotten, 0 0 0, last component is 1. Now, suppose I have got a situation like this, I have just only one vector u 1 n, suppose I have got only vector 2 u, just for illustration purpose, u n has only 1 column vector. I will generalize it to this case, but suppose I have got only 1 column vector which is u 1 n; and this guy supposed to u 1 n minus 1 and then 0.

There is u 1 n minus 1 vector, and then just have 1 0 here; you understand that pi n is orthogonal to this because pi n has 1 in the last component, this guy has 0. And, this has all 0s here, this has data; you take the inner product, even if you have lambda factor and all, it will be 0, you understand, is it not? 1 into 0, and this data into 0. So, they are orthogonal. So, your pi goes this way, pi n.

Mind you, right now I am taking only this case where u n is this. So, here P n means and P n perpendicular, they are all related to what, the projection and projection error operator corresponding to the space span by only 1 guy. You understand; I will be using that notation P n, P n perpendicular here, but here the space is spanned by only 1 value, just for this example; it will you give some clarity, so this graphical treatment, then I will generalize, come to generalized case; fine, these 2 are orthogonal.

Now, suppose I consider, but there all length n, I consider another vector that is your u 1 n. It is a 2 dimensional plane, but does not, from that you do not come to the conclusion that since in this figure it appears, this is lying in the plane containing pi n and these 2, and they are orthogonal, this is always expressed as a linear combination because this is actually in dimensional space.

These 2 can be pointing in 2 different directions and this fellow could be pointing in other direction. I mean this fellow may not be in the plane containing this guy and this guy in general, not here, but in general. So, do not get deceived by this factor, you know, that I am drawing on a 2 d plane, so definitely this is appears to be the same plane which

contains this guy and this guy; plane means space, in general not; so in vector pointing in 3 different directions. So, 1 is not in the plane containing the other 2, the space counting the other 2 in general, but here it will not be, I explained that yesterday.

My claim is, if you consider u 1 n and pi n, these 2 fellow; or to start with, this if you consider pi n and this guy, and the space span by this 2 guys; or using real life geometry the plane containing this 2 guys; in vector space, in solid the space, spanned by this 2 in real solid geometry called the plane containing these 2 both will be same thing you can always see, in that plane any vector will be a linear combination of these 2 vectors.

My claim is that contains this guy, first claim; that will be simple. What is u 1 n? U 1 n means this part is common, last part, this guy gives me 0, but this guy gives me 1. So, I take the last component of this vector u 1 n, the scalar element, u 1 n; multiply that by, multiply pi n by that, so 1 into u 1 n that plus this guy together will give me this vector. So, this is linear, this is a linear combination of these 2.

Obviously, this is containing the plane containing these 2, this containing the space spanned by these 2; that means if I consider these 2 lines, if I consider these 2 and consider the space spanned by these 2 fellows, that is a subset, that must be contained in the space spanned by this 2 fellows because this fellows is always some linear combination of these 2 guys. So, space spanned by these 2 means what? Space containing linear combination of these 2 guys.

Any linear combination of these 2 guys, you think; you will replace this by a combination of these 2. So, obviously, that will be what? Finally, a linear combination of this fellow and this fellow, is it not? So, space spanned by pi n and u 1 n, that is containing the space spanned by pi n and this guy, and reverse also it is true.

If you now consider this fellow, this is also a linear combination of these 2 guys. Because u 1 n, pi n, this I multiply by minus of last the component of this guy. So, minus of last component of this guy will multiply by 1; now if you added that 2, last 1 will cancel now, you will get 0, is it not? If you consider pi n, multiplied by the minus of the last component of this vector; so that will multiply the 1 here, and then you add that two.

So, last component and minus of the 2, will be canceled, leaving the 0 here; top 1 will be like this because this is 0s, will not change anything. So, by the same logic then space contained, space spanned by these 2 fellows, is containing the space spanned by these 2 fellows. That means, if a is content in b, and b is content in a, then a and b are same. So,

space spanned by pi n and u 1 n, and space spanned by pi n and these fellows, they are the same.

But here this 2 are orthogonal. So, if I have any third party vector say d n, and project it orthogonally on the space, on the space because there is only 1 space here now; either they are same as that on that space, then that will be what? If I view the space as the one spanned by pi n fellow and this fellow which are orthogonal, then the projections is very easy to compute; that will be what? Projections along this component and projections along this component.

Projections along this component will be what? Last guy; d n inner product with pi n divided by norm square of pi n into pi n, is it not? D n inner product with pi n minus will give rise to what last one. And then norm square of pi n is 1; pi n, norm square is 1; lambdas will effect here; this fellow is free of lambda; norm square is 1. So, last component by 1 into pi n, so that last component multiply the 1.

So, that means, can I write this way? P n, you are understanding one thing; now I am not making use of any, you know, physical geometry here. I am still using vector space logic only. My logic, I am giving a pictorial description, but my logic is purely using the examples of vector space. If I am here, if I do not know the figure, I can write directly, because I said by linear combination approach there is one space is contained in the other and vice versa; I did not use any real life geometry for analogy and all that.

My treatment is still purely based on vector space. So, with this figure only was used to have some clarity that is all, visible clarity that is it. So, p n, not p n, so first I write span of, if I call it w pi n, span of in this case u 1 n and pi n vector, then this is equivalent to span of this 2 fellows, that we proved just now. This is what we proved here, is it not? At this space I am calling w pi, n; this space is w pi, n; this space is w pi, n.

The corresponding projects, the total, here I consider only the w n, mind you; here I am appended another guy pi n, that is the only difference between these two. Here it was purely used, now pi n has come in, that is why I am calling w pi, n; it was only purely w n. And here the corresponding projections operators are p pi n, p pi n perpendicular, fine; do you understand their meaning?

Then you see I can easily write, if I take the third guy, third party external vector d n projected on this space, I use this thing because pi n and this are orthogonal; that will be what? Projection on this and projection on this. Projection on pi n will be what? D n,

inner product with pi n by norm square of pi n into pi n; norm square of pi n is 1; norm square is 1; others are 0; this one 1 square 1. No lambda on that, 1. So, forget about that denominator; d n vector inner product with pi n will be the last component of d n into pi n. So, d n into, that last component into, 0 0 0 0 1. So, essentially a vector, and where all are 0, and last one is the last component will be d n, that is d n.

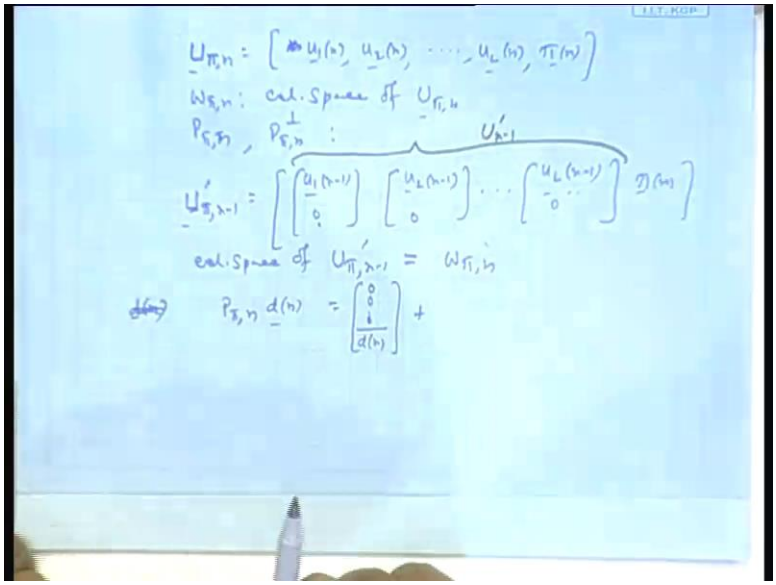
So, one vector will be 0s and then d n. There is a projection on pi n. And, projection on this will be what? D n inner product with, instead of taking the inner product let us, let us not use that formula here because I will be generalizing the case where not only u 1 is project, so many are project. So, there inner product thing will not vary; inner project works only with the 1 vectors on which I am projecting. So, I can say inner product between the 2 by norm square of that.

We are projecting d n on this vector. So, essentially you will be finding a linear combiner coefficient, only one in this case; that times this vector subtracted from d n error vector, norm square of that error will be minimized. But, that norm square will be what? Last component of d n; last component of d n is not 0. I am doing it here, but I wanted to do it in a general case.

If you permit me because I am just not able to take the temptation of going to this general case where I mean, you know, you have all the vectors present. Because just a minute, it is very silly to just deal with that becomes very simple case actually. So, upto this I used this thing; that, in this figure we had found out that the 2 spaces are same, w pi n. You can see this way, and any external vector can be projected using this treatment where it is simply a summation of these 2 projections.

This we did for this case; now I will generalize where it is not only 1 vector, I will generalize this statement where all vectors are present; I will just write down this equations. And then I redo this projections for the general case whose special case will be this where only 1 column vector is projected because that is more, that is greater actually, you know.

(Refer Slide Time: 29:23)



So now, once in this figure I will be now generalizing; here now I have only e 1 fellow; I will be considering. this vector, u pi n, as, u 1 n, u 2 n, dot, dot, dot, u l n, as before and pi n append it; you understand this what I did, about only one vector was projected; I am just giving a name. And the corresponding space spanned by w pi n cal space of u pi n, and this projection operators.

So, you understand, I am not doing anything new, but I am just taking so many vectors at a time. I mean I wanted to do the entire thing by one go. That not do the sum this treatment for 1 vector and again redo the same thing when the general case comes that is the rigid you know, it was to come to the general case directly. P pi n perpendicular, you now these 2 operators, I am not, I do not want to write their meanings.

P pi, n is the orthogonal projection operators for this space; p pi n is perpendicular is the corresponding error operator, you understand this, right. And, I define this vector say u pi prime n minus 1, that is what you now. U 1 n minus 1 0, u 2 n minus 1 0, dot, dot, dot u l n minus 1 0, and pi n. So, you understand I am not doing anything new; earlier I did the same with only 1 vector.

Earlier on this u to u l were not present. I consider u 1 n, appended with pi n this; and this fellow with this. As I said this space spanned by the 2 vectors here and here they are same, that is what I did that time. I am now a little more general. Is this seen clearly? Earlier, u 2 to u l were not present. I considered u 1 n and pi n, and other case I

considered u 1 n minus 1 0 and pi n, that is what I did in this figure. One is these 2, one is these 2.

Now instead of only 1 vector I am considering all of them. So, here all of them is having this 0 fellow. Once again by the same logic you can see the space spanned by this column, column space you can see, column space of, is same as the same w pi n. I proved it here for the simple case where it is 2 vectors or only 1 u 1, same logic can be used here.

Firstly, you consider this. I have considered this. First I show that column space of this column, this matrix is contained in the column space of this matrix, and then I will show the reverse. Column space of this matrix; first consider pi n, pi n present here, present here. Column space of this matrix is this, but the space where any vector is typically a linear combination of these fellows.

Pi n present, present. And, any other vector if you pick up say, u k n, that is obtainable as what, a linear combination of pi n and the vectors here with the u k n minus 1and 0 by that logic. Suppose, you are consider this same thing, u 1 n minus 0, it is very simple u 1 n minus 0 this vector pi n; you can combine the 2 to get u 1 n; I have done it already. Multiply this pi n fellow with the last component of u 1 n. And, add that with this; you will get back that; same for all the columns. And, pi n also is projected.

So, any linear combination of this fellow is a linear, is ultimately a linear combination of this fellow. That means, column space here is contained here, but is a column space here and vice versa also. Column space of this matrix is contained in the column space of this matrix because pi n, pi n, pi n, then any vector here will be say, you take u 1 n minus as 0, that is a linear combination of u 1 n and pi n; just take the negative of the last component, multiply pi n by that, add that 2; last component will cancel and give you 0.

Very simple; just repeat, go on repeating that logic on all of them; that means, the 2 column space are same. So, I am doing that now; you see, general case that d n is a external guy. I want to find out orthogonal projection of d n on this space. This space can be viewed as a column space of this matrix or this. But here advantage is pi n is orthogonal to each of these fellows.

They are not mutually orthogonal. See that is the little difference. Then these fellows are not mutually orthogonal, that is the reason I wanted to be general. That when many vectors are present they are not mutually orthogonal that fact was coming up here

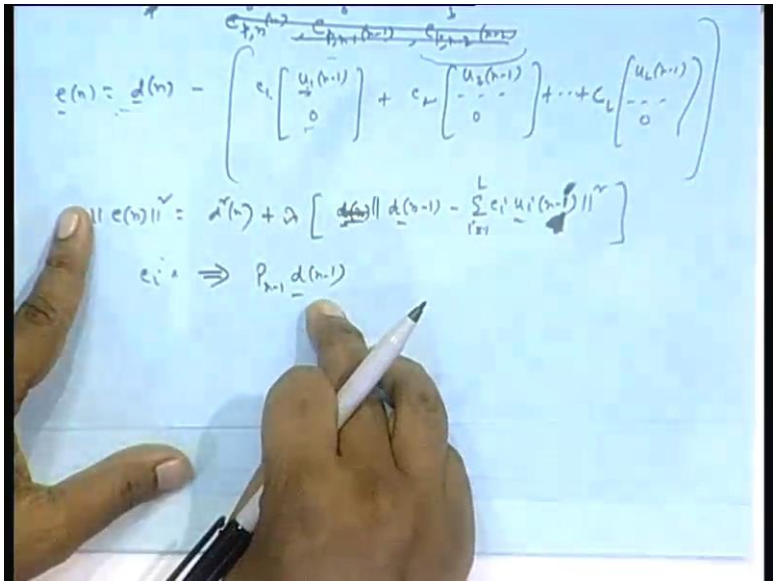
because I just chose 1 fellow. So, that is why I wanted to switch over to this general case. Yeah, pi n is orthogonal to them; that means, over all projections is what, projections on space spanned by this side plus the projections on pi n, is it not?

So, P pi n d n is what, projection on pi n that we have already seen what it is, in a previous page. That is irrespective of whether there 1 column vector or many, projection of d n on pi n we have already seen; inner product with pi n divided by norm square of pi n into pi n, that was giving rise to 0 0 0, and the last component; we have seen that. Please do not ask me to repeat this. This you have seen d n on pi n, we all know what it is.

Now, comes this bigger issue where I stopped and kept the general case, d n to be orthogonally projected on the space spanned by this part, this matrix, this particulars you call as matrix u prime n minus 1, pi dropped from here; u prime n minus 1 means all these vector are there with the 0 as effect. So, that means, now I have to project d n orthogonally on the column space of this matrix. This pi n I have taken care of here. I have to project d n on the space spanned by this column, that the column space of this matrix.

Please see this carefully now. Let us go to the first principal form doing orthogonal projections. What do, how do you do? We find out a set of linear combiner coefficients by which you combine the columns then take a error, take the difference of that combined thing from the d n fellow. And, take norm square of that, minimize that with respect to the coefficients.

(Refer Slide Time: 37:14)



If you do that, just for rough work I am using this part; if you do that, you are suppose linearly combining this columns, so on one hand you have got d n fellow and other hand you have got this columns c 1 into u 1 n minus 1 0 plus, say c 2 into u 2 n minus 1 0 plus, dot, dot, dot, plus say c what, c l into u l n minus 1 0, is it not? At the end you take the error d n minus this side, combined thing, and that error vector norm square is to minimize with respect to c 1 to c n, that is what we do.

Once we obtain the optimal set, combine them, these were projections; and subtract, this is the error; that is a simple thing. Now, in that you find the error vector. The last component will be what? Only the thing coming from here; here, all are 0s. So, the error, you will not take this, find the error and call it say e n; norm square of e n will be what? First take the last guy; last guy is squared up; no lambda there; you remember, norm square in last guy is 1, previous to that lambda, previous to that lambda square. So, last guy is d square n.

Now, I will write compactly using the intelligence. First just imagine the last but one component; c 1 times last component of u 1 n minus 1, e 2 c 2 times last component of u 2 n minus 1, dot, dot, dot, here. Subtract that from d n minus 1 that error square, multiply by lambda. Previous one also, multiply by lambda square. You take lambda common. Then what you get, these, is it not simply this.

You have not, you have missed out some classes I think; you have missed out classes or you have not been following. What was the definition of inner product that I gave? Norm

square is a special case of inner product. I gave inner product with a lambda or not, forgetting factors so that contribution of real first is forgotten. I am using that definitions, general definitions of inner product; you are also confused about lambda, is it not? Recheck your notes.

So, this is, please see, d n minus 1 minus, summation c i u i n minus i, i equal to 1 to l; this you are minimizing; this is the thing you have to see always inside, you know. You are taking the error. Last guy, just square up. It is independent of, because of the presence of 0 last guy does not have any impact on the coefficients, optimal coefficients to obtain. That is what is the crux of the story.

That new data has come, but that will not change in this case the optimal combiner coefficients because this is only square, separate term this is, when we minimize it with respect to c 1 c L upto c L, this will be giving rise to nothing, 0. But what you obtain from here is what? You consider the remaining terms; you consider the remaining error terms.

See, I am not writing like, you know, teaching in the school; I am not writing, you please see it yourself; just last but one term is d n minus 1; last is d n, last but one is d n minus 1; that times, c 1 times, u 1 n minus 1, c 2 times u 2 n minus 2, c l times u l minus l, combination of that error square, multiplied by lambda; lambda I am taking common; so that 1.

Then again, d n minus 2 minus, c 1 times u 1 n minus 2, c 2 times u 2 n minus 2, c l times u l n minus 2, that error square multiplied by 1 lambda because 1 lambda is taken common, so on and so forth; n minus 1, thank you, thank you very much, in this yeah, thank you if you correct me, I am writing in a flow, thank you very much yeah; u i n minus 1, n minus 1 yeah.

So, that means, what we are doing, what, when you are minimizing this, you are basically minimizing this. You minimize this part, you forget that lambda. And when you minimize it, when you project d n minus 1 vector orthogonally on the space spanned by u 1 n minus 1 alone, u 2 n minus 1, and u l n minus 1, no 0 present. What is that projection? That is the projection p subscript n minus 1 d n, is it not?

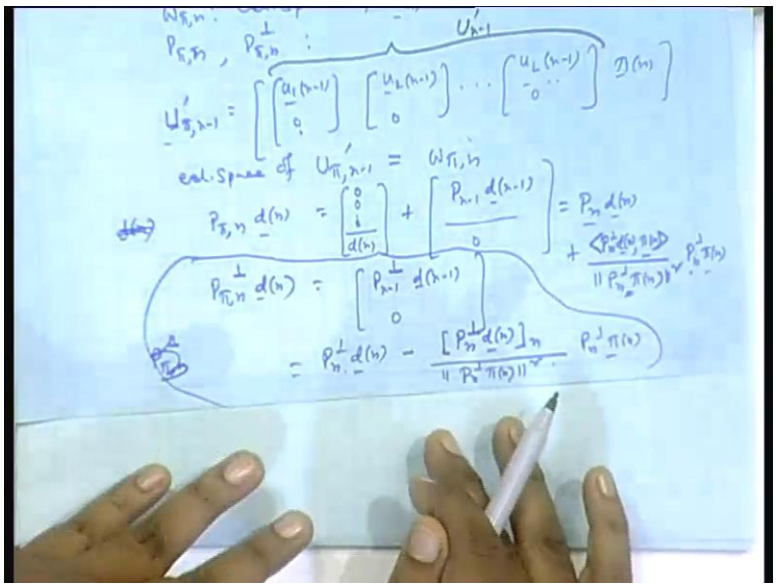
That is you get those combiner coefficients, you get those c i s which gives rise to, which give you this component, this thing P n minus 1, d n minus 1. When you computed P n minus 1, d n minus 1, you will get a set of coefficients by which you linearly combine u

n n minus 1 to u l n minus 1, those coefficients will be come up from here. So, if I now use those coefficients and combine them, last component will be 0; and add up that will be this fellow only because this vectors are combined by the same set of coefficients.

Now, those who are coming regularly will understand; those who are not, who are dropping in between they will find it difficult; please understand. You are getting the same set of first at that the coefficients; you are getting the same set of combiner coefficients because minimizing coming from here. This like minimizing the norm square of what, d n minus 1 vector and a linear combination of u 1 n minus 1 to u l n minus 1, that difference.

That is what you do when you project d n minus 1 orthogonal on the space spanned by u 1 n minus 1 to u l n minus 1. So, those combiner coefficient you get. Now, using them you are combining them. Last one will give rise to 0; these all zeros; but top one when you combine you get this.

(Refer Slide Time: 44:16)



So, over all thing will be what? When you are projecting d n on the column space of this matrix, first part will be P n minus 1 d n minus 1, last part will be 0; this combine, this 0s will give rise to 0; combine this upper part, this will be this. That means, instead of projection I take the projection error; if I take the projection error then from, I have to subtract this right hand side from d n.

If you subtract right hand side from d n this last guy goes, and on top of here d n minus 1 that minus this that will be the corresponding error. When you subtract, here it was,

when you subtract this right hand side now from d n because now you are finding out the error, the last component of d n is this d n; here also d n plus 0 d n, so subtracted 0. The other top half is d n minus 1 vector that minus this projection. So, it will be the corresponding error.

So, you see, I am somehow able to relate this projection for the time index n minus 1 to the projection corresponding time index n, but it is not simply P n perpendicular d n, I have to bother, I have to carry the burden of this extra guy, pi; whereas my intention was to have relation directly between that two. But this kind of relation does not exists, this P n perpendicular d n is simply this, that does not exist, this is what I could do.

But then how does it solve my purpose? From here I will proceed in another way. This is what? This again projection, this corresponds to what, projection on the space spanned by this also. You can either view it as the projection on the space spanned by these fellows, but the 2 spaces are same. So, this projection, forget about the error consider the projection. That is again the projection on the space spanned by these fellows.

This I can write as a direction of decomposition of what? This component this side plus, pi n projected orthogonal; if first take pi n projected orthogonally on the space spanned by this take the error, span of that. Atleast I am able to extract out this part separately because I finally, want a P n perpendicular d n, no, pi append n. So, pi I want take out so but I cannot throw away pi then everything will be incorrect.

So, what I will be doing here? This projection again, w pi n now you write as, you write w pi n as w n, sorry, w n, that is the space spanned by these guys; direct sum of what, span, I mean span of what, P n perpendicular pi n; pi n projected on the space spanned by these fellows that is P n pi n and the corresponding error, P n perpendicular error pi.

So, in this case again the same projection can be written as what? And I order by 2 n 2 a, this projection then I wrote like this. Same projection can be written in an equivalent way here as projection of d n on w n plus, projection of d n on this guy, is it not? That means, this can be written as P n d n plus, so at least you see P n d n is coming and P n minus 1 d n minus 1 they are coming. Extra terms are coming I agree; this is I cannot leave away, but at least there is some relation from n minus 1 nth index to nth index, is it not?

So, P n d n plus d n on this guy; I will write the inner product later; inner product, in the inner product simply d n with this, but I can swap this projection operator form pi n to 1 d n. I am not showing that step separately, there is a sort of the space, please you

understand. So, this can be written like this, pi n. When you multi, when you do inner product between pi n and any other vector when you basically get the last component, so this inner product will give rise to what, the projection error if you take d n, d n projected on this guy on the error, the last component of that error, is it not?

And, this was only P pi n d n. So, I do the formality for the corresponding error that is this guy, P pi n perpendicular d n was one, from one approach was this; and from this approach we call it to what? You subtract the right hand side from d n. So, this becomes P n perpendicular d n and this is minus sign only. And, you can write the top as the vector P n perpendicular d n this vector last component, nth component of that vector.

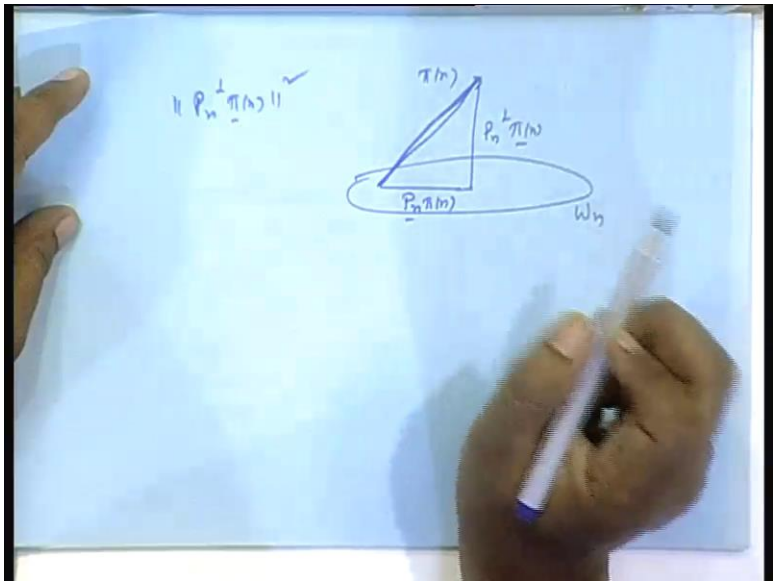
Divided by, ok; this is the relation we will be using frequently, this one. You can understand there is a direct relation between P n perpendicular d n and P n minus 1 perpendicular d n minus 1, but I have this additional burden. Even here also P n perpendicular d n is coming, this guy, last component the scalar, but I have this additional components.

Even you will see this will not create problem, when I apply in the lattice case this will not create problem because I will be able to swap this P n perpendicular from pi n to somebody else that somebody else's last component will turn out. I will after all have this guy; this guy will call give rise to a new variance again like various variances you have taken I have ordered updated. This will be another guy variable I have to leave with; that means, I have to update it, that can be ordered updated.

This is an extra variable that is generated, p, I mean here I have taken l th term; in our case we will see have simply order, p vectors. So, it till basically depends on order p. And, this actually is basically an angle parameter; I will show you that angular parameter now, and that can be order updated. So, that means, lattice will not only involve forward and backward prediction errors, other respective any variances or delta.

In the delta business, delta is nowhere in sight; do not get confused; delta is inner product between 2 projection error vectors; delta I have not come to; I am only updating, time updating projection error vector only, no inner product. But just hear me out, in the delta update I will be dealing with this addition another variable where we will come from there; and this variable then again needs to be updated, data dependent; this can be used in the order updating.

(Refer Slide Time: 52:48)



I will there give you the meaning of this and then we will call off for the day. P n perpendicular pi n square, if you take this to be the space, w pi n, not w pi n, just w n; w n, the space spanned by these guys, w n; suppose this is w n and here is your pi n, sorry, here is your pi n. So, this is your projection, P n pi n, and this is P n perpendicular pi n.

Now, you even without using any geometry, we all know that square of this plus square of this is equal to this. That means, if I take the ratio of norm square of this guy by norm square of this guy that will be less than 1, less than 1, greater than or equal to 0 less than 1. So, that that can always be; forget about the geometry; this is a quantity like P n perpendicular pi n, P n perpendicular pi n, norm square of that divided by norm square of pi n, that is can always be written as some kind of sine square theta.

Since in norm square I am saying; actually with the norm it is a square we will take a norm, norm of these guy; P n perpendicular pi n divide by norm of pi n. That can always be written as some kind of sin theta, but norm of pi n is equal to what, 1; this guy's norm is 1; last component is 1; others are 0s. That means norm square of this guy can always be equated to some kind of sine square theta.

In this geometry, in this, mean geometrical I mean what I am doing the analogy you can take that this to be that angle. That is why this quantity is called the angle parameter which is to be updated. I still took the general case of u 1 l, u 1 n up to u l n. I have not used these in the specific case of lattice. I gave you so far the general result, but my

starting point was updating delta P n; these results can be very quickly and very nicely used now in updating delta P n.

I suggest that when you come for the next class please come prepared with these, please. I will not rewrite these relations. I just cannot because I am running short of time. I will not. Every class I say this, but I do recap, this cannot be done. I will simply skip. You please on Monday you come with these. I will wind up this topic in 10-15 minutes time which I could not do today. And then I will, I have to move to another topic, another version of analyze filters which is also very good which leads to systolic arrangement. That is all for today.

Thank you very much.