Adaptive Signal Processing

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

Lecture No. # 35

Givens Rotation

So, you convert this recursively squares lattice filter at length. It will be algorithm. It was both order updating and time updating. The delta, there was at delta p n. The inner product between two prediction vectors that was to be time updated because that cannot be order updated, but variances of the backward and forward error, forward prediction error vectors you know those could be that is a norm square.

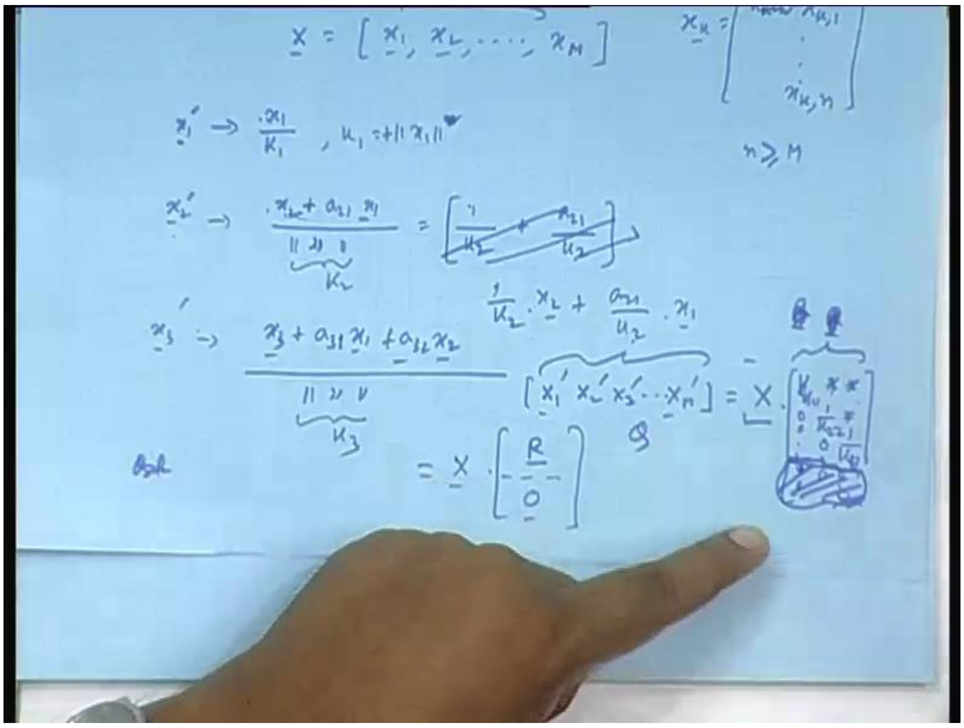
Norm square could have, it could be order updated and then, in that order updated relation, we found out that one term comes in the denominator which has the potential of becoming 0. That is of course another term. Also, the two terms come are I mean norm square of the forward prediction error vector and backward prediction error vector, but of these two, the backward prediction error vector can become 0, while by the order updating process at a very early stage that is when the time index n is much less.

You are doing order updating and then, when all 0 vectors could be projected on some space on the future space and an error take it that will give rise to all zero backward prediction error vectors whose norm square is 0. That will come in the denominator division by 0. It might come up. There is a linear dependent. So, to avoid that, I added the initial condition phase, one very small constant delta as the initial value of backward prediction error vectors. The initial value of the norm squared that is norm square of the backward prediction error vectors that is sigma p b minus 1 sigma p minus 1 b whole square. That is n equal to minus 1. The initial value sigma p, minus 1 superscript b whole square that I took to be delta, or delta is a positive, but very small quantity and then, you can see you just take n equal to 0. Then, n equal to 1, n equal 2 and see some stages and how that backward prediction error as you go further along the stages, nth order of the process will see even at n equal to 2 or n equal to 1 or n equal to 3.

As you increase the order, you will see that mechanism will see that backward prediction error vector. Norm square is still remaining 0, otherwise is not here. Here it will be equal to delta, so that delta will always propagate. It will make sure that division by 0 that would not come. I am not going to that thing. You know it is just I mean you have to run the recursion manually. Take 1 or 2 n equal to 0, n equal to 1, n equal to 2, and few stages p equal to 1, p equal to 2, p equal to 3 and then, you will see that you can remain backward prediction error that is ideally 0. Therefore, the norm square is ideally 0. That is the delta will be there as an offset, that is of the division by 0 thing. It is again care of there.

Then, it was end of the previous day. I told you about a new topic called q r factorization, where q is a unitary matrix; r is an upper triangular matrix, q r factorization. Recursive this square is q r factorization that is what I will concentrate on today. So, I will just start with the basics of what is q r and all that. Even for the time being, forget adoptive filter because again numerical linear algebra is very useful in solving linear equation on you know very fast algorithm for a solving equation and all that. Actually suppose I am again using the same rotation.

(Refer Slide Time: 04:35).



I have got a matrix x of columns x 1, x 2 dot dot dot dot say x m and each one is a column vector. So, in general x k vector will be x k 1. Shall I use this way or may be x k 1 dot dot dot x k some index n in this. So, total number of row is small n. Consider small n number of rows to be higher than capital N and that will always happen because this will be data. My vectors and data you know as time index because n is increasing always. So, naturally after some time I mean you will definitely have situations where length of this vector will remain higher than m because as time moves, the length will move.

So, for the time being, take small n to be greater than n and that will ensure that linear independence is not evaluated because the moment you take number of row less than number of column, then I have the problem that all the vectors are not I mean there is no linearly independent. There is say n, I do not want to avoid that. I will come back to the independent x n and then, again later when some algorithm, but for the time being assume that there is a number of row n is an index. Actually, it will filter in context to actually amount to time index, but here it gives the number of row that is higher than we assume that n is greater than equal to m, and also the columns are linearly independent. They are LI linearly independent. It is given. So, I have got a perfect linearly independent set for that only I need n greater than equal to m obviously.

Now, suppose I want to do orthogonalization of this and then, ortho normalization also, so that each of the orthogonal vectors, it come out of it as a unit norm. That is not a problem. What I will do is I will take x 1 as it is again there is nothing unique about it. I could start with x m or any other vector and then, go on building. I will start with say x 1. X 1 as it is x 1 will give rise to just x 1 by say k 1, where k 1 is norm square of, in fact norm should not be square. It should be square root of norm square, so k 1. So, it is a positive number. This is a positive thing.

So, x 1 from x 1, I get a vector with unit norm, so that then x 2 I have to generate that orthogonal set of orthogonal basis. You know I mean basis vectors out of these you all are familiar with. What will I do here? I take x 2 project it on this guy and take the error and then, divide by the norm square of the error to make it unit norm. So, that means, I will get x 2. This should be it. I should have a prime here. This is a new vector. I am projecting it. It cannot be x 1. X 1 prime from x 1, I develop the first orthogonal basis vector which is x 1 prime. This unit norm there I develop x 2 prime. How do I develop x 2 prime?

First x 2 and this will be minus, but I am just putting it plus for the time being, some constant times x 1. What is that constant inner product that we know? Inner product between x 2 and x 1 divide by here. It is no problem some constant, but that constant formula here is not I mean I am not much bothered about that nature of that. We will have some constant. Say a 2 1 into x 1, this is the vector and then, divided by norm of the above this thing norm of that is 1 by you can if you call this quantity k 2, this is 1 by k 2 plus a 2 1 by k 2 times. Sorry not this way. I immediately saw some in actually. I am frowning how can I put this in a same bracket? I would not do that. X 2 and a 2 1 by k 2 into x 1 see the structure.

First, see this 2 I will do one more and then, you will get this x 1. I had x 1. X 1 is just divided by k 1 that gives you this. I had x 2 from that. How do I get this? X2 and x 1, they are linearly combined and that is why I gave what is the form of linear combination x 2 fellow get 1 by constant. This fellow gets something by the same constant. Now, remember this cannot be 0 because x 2 is not lying in the space spanned by x 1. Then, x 3 prime, this is the last after this pattern will once X 3 prime will be x 3. Essential thing is person you are targeting that vector remains as it is not multiplied like here. Not multiplied by 1 here, multiplied by 1 here, and multiplied by 1. Just a norm will come in front of x 3 plus some constant say a 3 1 x 1, a 3 2 x 2 divided by norm of the thing and this you call k 3 so on and so forth. This is the structure.

So, that means, if I put these vectors like this in this page only, let me do. If I put the vectors, the orthogonal vectors x 1 prime x 2 prime x 3 prime dot dot dot dot x m prime, that will be this x. See some having a space. I am not rewriting the entire x is this x times 1. Matrix of this form 1 by k 1 1 0 0 0 0 first rest I am writing, but 1 by k 1 1 0 0 0 0 0. Isn’t it? How to do matrix vector multiplication? You know you have to linearly combine the columns by these elements. These zeros will be multiplied by these fellows. Forget that 1 by k 1 1 times x 1. That is what you get and then, x 1 prime. Then, x 2 prime will be 1 by k 2 2, right and here something I am not writing some star, but then 0 essential thing is these zeros. Then, again it will be some element here, some star, some star again 1 by k 3 3 0 dot dot 0 so on and so forth.

So, what kind of structure is coming? It is an upper triangular matrix. On top upper triangular matrix size is m cross m and below that there is a chunk of zeros. It is very simple. This upper triangular matrix may move in the chunk for the time being with the chunk zeros. Only I will call R, together R. Actually the problem is you know books sometimes take this to be R and write it as R and 0, or maybe if for the time being let me put this way. If I switch over later the change, the rotation later and take this, only this upper triangular part R, will there be confusion or maybe.

So, let me write this way. X R and 0, this matrix I can call Q. This is not a square matrix, but one thing is that all the columns are there mutually orthogonal. So, Q hermitian Q will be not QQ Hermitian. You cannot do the inner product. This is not a square matrix. Q hermitian Q, that will be no that matrix will be Q hermitian Q means this column will come as row with Hermitian. I mean these columns, all are real. So, hermitian transpose are same. So, you do not bring complex here. So, here you have got x 1 prime vector, here you got the corresponding row vector. Their product will be one identity matrix. Isn't it?

So, Q is unitary generalized. Unitary matrix is not a square matrix, but a rectangular unitary matrix Q transpose or Q hermitian transpose. They are same here because we are considering real data q transpose Q is I. Q transpose Q, but not QQ transpose. This is not a square matrix. That is why Q transpose Q is permissible, not QQ transpose. Always whenever you have a unitary matrix, we show Q transpose Q is I row times column. The inner product is either 1 or 0. That way since if there are square Q transpose Qi means if the two matrix square matrix is a, and b are such ab is i, then it can be proved linearly geometry and easily then, ba also i.

Using that we say that Q into Q transpose also I that follows from that, but Q transpose Q is i that is that comes from unitary index which is the unitary matrix is a square that applies here. Now, for the time being let us do this. Qs xR 0. Let us call this to be a Qr decomposition. Actually this is not x 2, b x will be some form of this kind of form QR. Some Q some R may not be exactly this, but that comes from this. For time being, we will see that given the data matrix, we can write it this way x into R. That is Q. Question is how to get R. Obviously, you have to first find out k 1. That means you have to take the entire vector and find this norm square. Then, after that you have to find out a 2 1 or the inner product and then, again norm and all those things will be there.

Suppose this time this is the actually the time index, and you have to do this business at every index. So, every index of time you have to compute this R. R will be every matrix will be a function of n and then, R n Q n Q as a function of n R as function of n, right. You have to do it. That means, it is too much of computation and computation will grow exponentially with time because as n increases, inner product length also increases. Isn't it? So, that is not way. That is not a practical way, but there are ways, smart ways to get this R from given x. Q I mean actually form of Q. We do not know the fact that Q is unitary. That is enough and that fact we will be using.

We will be using this R. How and all there that is coming, but you cannot look for this at every index. I will find out a new R by group force computation. That is ridiculous. That you cannot do. So, that business you have to do adapting. I am just giving a hint before and actually I have not even brought it to the context, where this is we know relevance of this, but this is an adaptive filter or optimal filter and all those things that I will deal with. That I will do. One way of doing this R computation, there are several ways. One smart way is by a technical Givens rotation which can be made adaptive and which basically leads to a very nice implementation on something.

When I apply Givens rotation to this kind of thing in context of adaptive filter, entire thing can be realized on is a very regular array of some unit process. This is a regular pattern local connection only. All process has a pipe line. I mean there is pipe line in latches between processes very fast. Unfortunately I cannot tell about here I mean those who have done my course earlier, they know it at length, but this will relate to specifically QR is look for those kind of adaptive filter applications, where we have got number of gaps, number of links to be very huge and computation because very huge means part sample, output computation, weight update computation, all those things. It will get too much of complexity.

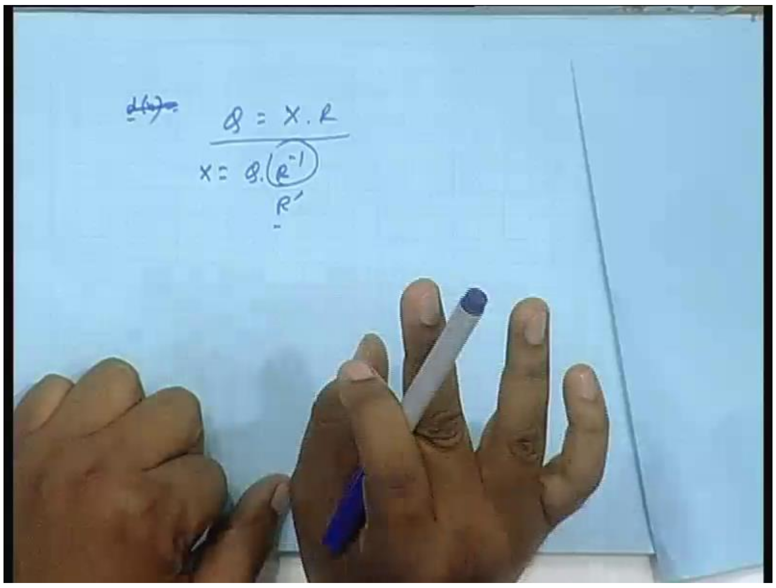
So, they are parallely in the computation for you know using the computational overhead, we go for this. Givens rotation I will come little later today, but that will take time. This is beautiful thing, but it will take time. Let us first consider its application in this case, adaptive filter case. So, then x 1 will be a x 1 with in bracket, n a vector instead of x 1, it will be a x 1, n x 2 will be I mean x 2 n in the adaptive filter case that we studied x 2 n will be z inverse x 1 n, x 3 n will be z inverse to x 1 z inverse 2 x 1 so on and so forth. That is what we, but I will still try to be more general here. I call them separate column vectors. No problem, so x 1 n x 2 n dot dot dot x m n. So, this will be capital X n m is fixed. This is not order recursive. It is a fixed order case X m X n, sorry that is the thing.

So, now I have got one desired response vector d n. Yeah please all questions are welcome tell me. Just a minute, just a minute, you are right. Just a minute this is one mistake here because number of column is n. Isn't it? In fact, the zeros would not come here. Sorry. Thank you. 0 should not come here and 0 is required. We do not think. So, zeros will come in this when I go here with this adaptive filter. So, this brought it forward from there. In this case 0 would not come. Actually when I do reverse rotation, you will see this chunk of 0 will come. So, I should not bring it forward here. In this context, zeros do not come. I think you should have, you have kept quiet and I also have made a mistake. Zeros do not come.

So, it is a smart thing that he has found out this chunk of 0. It does not come. I just have a zero in mind actually. There is something called 0, I mean column row and forming a 0 row and therefore, chunk of 0 rows will come. So, I will have that as a target actually in this Givens rotation I think. So, somehow that we have got carried here, this is not that is let us again look here. Thank you. Just take a relook here xx. This 1 by k 1 1 0 0 0, so 1 by k 1 1 multiplies x 1 prime. These zeros would not come easily. You see zeros do not come. It is just slip of tongue actually. Then, this 1 by k 2 2 1 by k 2, this is not k 2 2 k 1 k 2 k 3. Anyway, 1 by k 2 1 by k 3 and these terms, these zeros will take care of that k. Here 1 by k 1 multiplied by x 1 prime xx 1, then something which in this case actually a 2 1 by k 2 comes here. Some data that multiplies first guy x 1 and 1 by k 2 multiplies this guy. So, this is just an upper triangular structure square matrix actually.

Instead of stopping here only, I mean I did not notice I will come to this something more that comes to my mind. Now, in this context looking at this structure, I will come to that soon, but let me come back to the adaptive filter case, or may be here only I say in this context I complete this and I come to adaptive filter. In this context what I found?

(Refer Slide Time: 23:12)

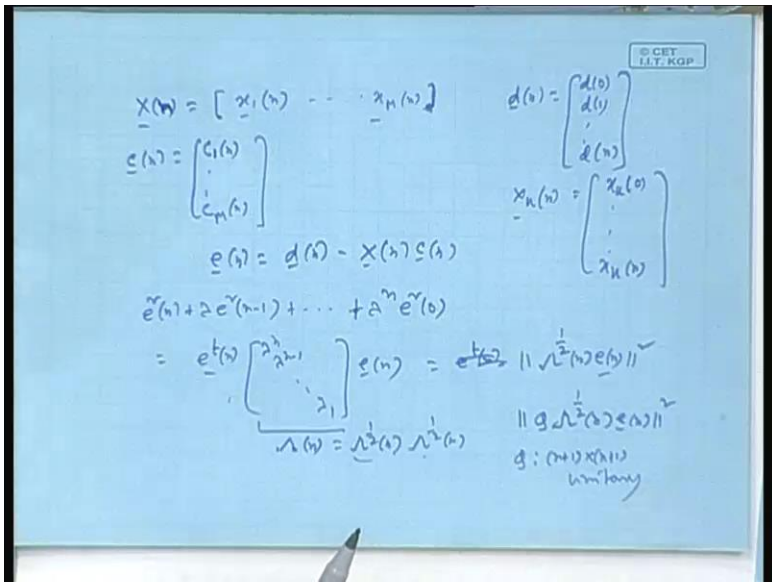


Q as x R. R is a strictly upper triangular matrix. No chunk of zeros. R is a strictly upper triangular matrix and then, x will be QR inverse and R inverse recalls the R prime and R prime also will be upper triangular. That is why it is actually called Q R factorization of x though we will be needing this form. Not this form, but this is called Q R factorization. Why the inverse of an upper triangular matrix? Upper triangular matrix must have no zeros diagonal entries, otherwise determinant will be 0 and it is invertible. So, that will come here. Obviously, you know 1 by k 1 1 by k 2 2 and they are not becoming zeros, and they are perfect linear independent.

So, this will be invertible, but upper triangular matrixes inverse is upper triangle, lower triangular matrices inverse is lower triangular. Anyway those who do not I want to see take it. So, next time we do it. We will take a class tomorrow. Actually I will let you know because we missed the class on Tuesday, so tomorrow at 3.30. So, that time we will see just take it as a problem. It is not a problem. Actually just to see how it comes because I will not be using this form. As I told it is just for academic interest. This form I am not so much bothered, but this is how it is called x. In a given matrix is QR factorized where Q is a rectangular, your unitary matrix.

Rectangular means number of row greater than equal to number of columns. That way this R prime is an upper triangular matrix. We will be using this form. Now, R prime is R inverse R is invertible if R is upper rectangular. How R prime is upper triangular? It is not difficult. You take it out. This is not coming in our business. So, I am not bothered about that.

(Refer Slide Time: 25:24)


$$\underline{X}(n) = [\underline{x}_1(n) \cdots \underline{x}_M(n)] \qquad \underline{d}(n) = \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(n) \end{bmatrix}$$

$$\underline{c}(n) = \begin{bmatrix} c_1(n) \\ \vdots \\ c_M(n) \end{bmatrix} \qquad \underline{x}_k(n) = \begin{bmatrix} x_k(0) \\ \vdots \\ x_k(n) \end{bmatrix}$$

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

$$e^2(n) + \lambda e^2(n-1) + \cdots + \lambda^n e^2(0)$$

$$= \underline{e}^t(n) \underbrace{\begin{bmatrix} \lambda^n & & & \\ & \lambda^{n-1} & & \\ & & \ddots & \\ & & & \lambda_1 \end{bmatrix}}_{\Lambda(n) = \Lambda^{\frac{1}{2}}(n)\Lambda^{\frac{1}{2}}(n)} \underline{e}(n) = \cancel{\underline{e}^t(n)} \|\Lambda^{\frac{1}{2}}(n)\underline{e}(n)\|^2$$

$$\|Q\Lambda^{\frac{1}{2}}(n)\underline{e}(n)\|^2$$

$Q : (n+1)\times(n+1)$ unitary

Now, coming into that erratic filter in problem. I can point it out, but again you did not say that zeros are not required. Zeros you put there right. Zeros are not required. It is a serious mistake. I should never overlook it. Anyway, this is given to you x n. Sorry I have got a desired response vector d 0 d 1 dot dot dot say d n, any general vector x k n, any column vector which is typically of this form x k. Now, did I put 0 here earlier with in brackets in the lattice case or 0 in the subscript with in bracket. So, with in bracket, this is desired response. I want to find out the optimal set of optimal combiner coefficients say c n as c 1 n dot dot dot c m n, so that these error vectors with this d n minus x n c n norm square of that is minimized.

Of course, we all know that using the orthogonal projections exists unique. Alternatively, the norm square will be the quadratic function of the combiner coefficients of unique minima. You can find out. What is the norm square? Norm square will be take the vector e transpose n. There is norm square, but we are using the generalized norm square here with the forgetting factor lambda. This is e n, but we are not simply taking e transpose n en. We have the lambda this thing that is if you have forgotten e square n plus, this is a repetition. Actually plus lambda to the power n e square 0. Isn't it? These are norms and this you can write as e transpose n.

This we have done in the lattice case. So, we should not get confused. That time I gave a name to this matrix. What was the name I gave? Anyway, you do not need e transpose n. This matrix I can say this can be written as lambda half n lambda half n. You can define 2 lambda, this is 1 matrix, this is a rotation. What is this? It is a diagonal matrix. Instead of lambda, I have got square root of lambda positive square root. See you understand if you multiply 2, you will get these. This is a diagonal matrix. By definition this is a diagonal matrix. What are the entries in this? Just I mean same as these, but lambda replaced by positive square root of lambda. If you consider such diagonal matrixes multiply, you will get back these.

Diagonal matrixes are symmetric matrixes also. So, then this norm square you can write as you can easily see what this norm square is. This quantity transpose times, this quantity e transpose n and transpose of this which itself and then, this. Isn't it? Actually it is more generalized norm square. It is not simply of norm square en, but en pre multiply by the diagonal matrix of this kind norm square and then, the usual norm square of that. The point is if I find out some unitary matrix, this vector what to say length n plus 1 vector. Isn't it? E 0 e 1 up to en. So, these are length into vectors.

If I have suppose unitary matrix Q square matrixes. Square unitary matrix Q and instead of these, I take this quantity Q into this. Q is the unitary matrix, Q n plus 1 into n plus 1, just any matrix. Very soon I will start using Q and make it a function of n because I will mainly bother about Q. That has to be evaluated at every index n. So, here just I am showing the results, I am just taking general Q. Only n plus n plus 1 unitary. Then, these two norm squares are same. You know you see that very simple. What is the norm square of this? It is vector transpose of these into itself.

So, e transpose, this guy diagonal dimension in Q transpose followed by Q. Q transpose Q is I remember unitary of matrices or unitary operators. In more general vector space sense, they all are always norm preserving. It can also be sure I was trying to work out in the morning. You can get some more results. Any norm preserving operator is a unitary at least for real and complex field. That is only when scalar real and complex operators which preserve norm transformations to give vector either as abstract vector for this kind of column vectors transforming to another vector. If they preserve the norm, then they are unitary matrices. Of course, the transformation has to maintain the length also. You cannot take that. I take a length 10 vector and make it length 5, where norms are same. I am not talking of that.

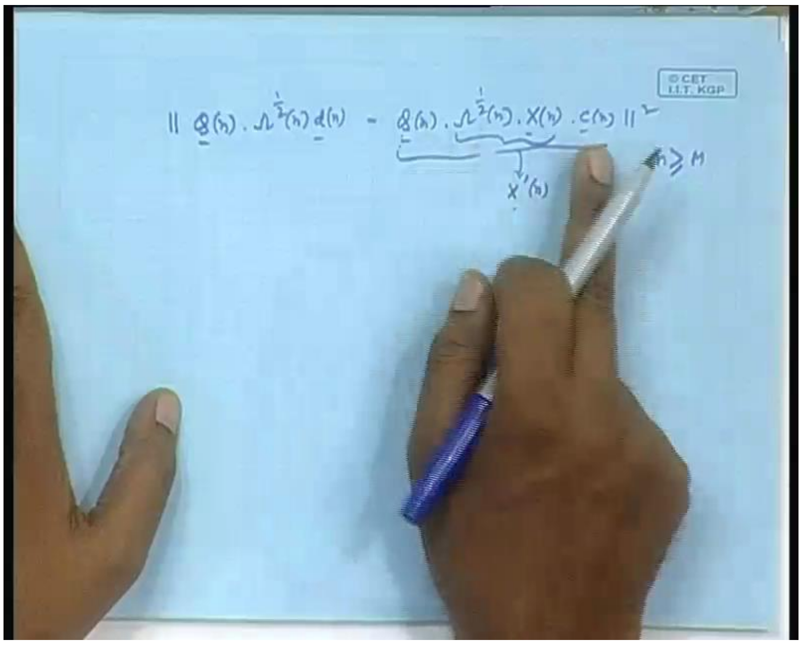
Our transformation which take a vector space to itself, and if is norm preserving, it is unitary and vice versa. If it is unitary, it preserves norm that I will not prove. This part is very simple. This we have proved. So, any norm squared, it remains in variant to unitary transformation of the vector concerned. I can you give some examples of unitary transformations of physical. These you know vectors. Don’t bring that also. Any transforms we deal with, we know that unitary that one day I told you also when I was discussing orthogonal basis and I told you I gave some examples of D F T basis I did in this class. So, they are unitary transform, but forget that I am saying that in real life world say in three-dimensional physical world, there are vectors and not abstract vectors. Simply x, y I mean x followed by y followed by z. That kind of vectors.

So, vectors of length 3 if I want to map it to another vector of the same size, you have to multiply by linear operator. You have to multiply by matrix, pre-multiplied by matrix of size 3 by 3. That is the linear operator here. Suggest some unitary operator. Three-dimension rotation is one. You rotate the vector and its length does not change. That is what rotation is. What we have been dealing with rotation is unitary translation. I am talking about operational vector rotation reflection. They are about these operators. You know they are you can walk out the operators, they are unitary operators. We will be considering rotation in particular definitely, but anyway our purpose will be to find out some appropriate key at each index n at appropriate versions that I am coming now.

I do not think I can consider givens rotation to get the time is running, but this is very interesting topic. This is very you know I mean visually you can easily understand what is happening. So, suppose instead of Q I call it Q of n. That is the only thing I will say because I am starting at my matrices are dependent on n combiner coefficients looking for they will be dependent on n. Isn’t it? Norm square is a norm square at nth index. So, the matrix Q I will be looking for its appropriate that we should make it a function of n. If it becomes independent of n, then we will drop the index n. So, how to choose Q n is unitary and square unitary, and if I put a Q n here, I will minimize this norm square. That will give me the same solution about that there is no problem because the two norm square are same.

Now, this is a quantity. I know this e n is this, d n minus this fellow I replace en by that. So, this lambda half n times d n 1 vector lambda half n times x n in to c n another vector.

(Refer Slide Time: 35:12)



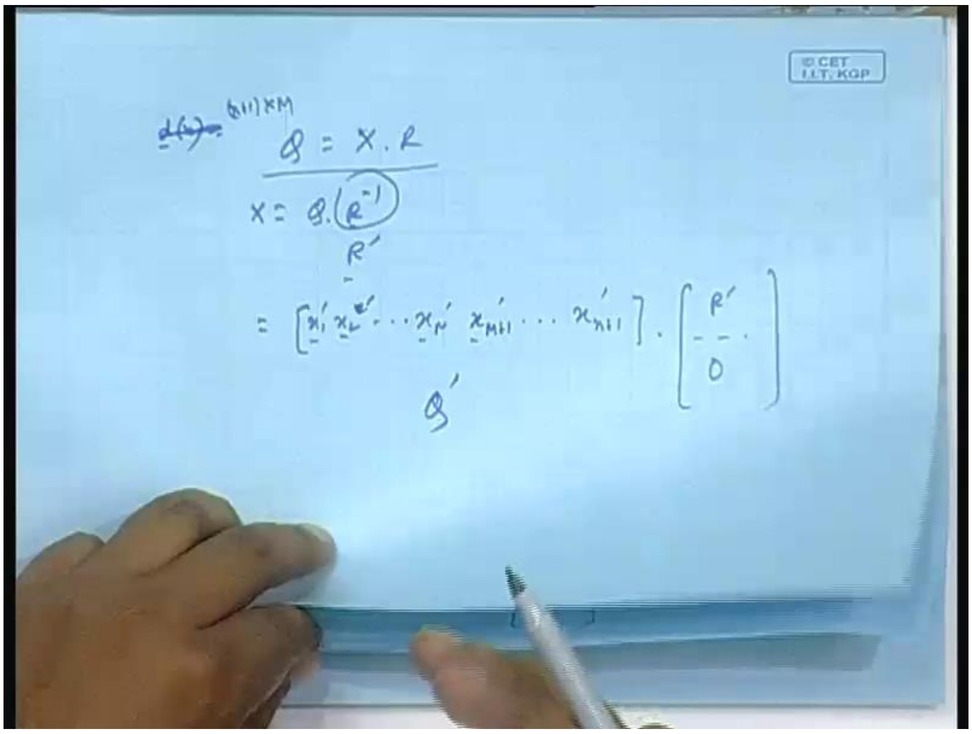
$\| \underline{Q}(n) \cdot \Lambda^{\frac{1}{2}}(n) \underline{d}(n) - \underbrace{\underline{Q}(n) \cdot \underbrace{\Lambda^{\frac{1}{2}}(n) \cdot \underline{X}(n)}_{X'(n)}} \cdot \underline{c}(n) \|^2$
$n \geq M$
© CET I.I.T. KGP

There is I am minimizing Q n into minus Q n into x n c n, right. This norm square I have just expanded it for the time being. You concentrate on these and I will choose Q from this side, not from this. Whatever Q comes out that will be simply used here. This is normally when you do not use the lambda. This matrix is not there, but since it is there, it is diagonal matrix. You consider these two together. You can even for your convenience you can give it a name say x prime n or something like that just for, so that I do not have to always say these into these x prime n and assume the index n to be of course greater than equal to m. That is always there. Otherwise, linear independence that will be validated automatically. That is always there.

How do you take care of situations, where n is less than m? That we will do separately. That is our Q n I will choose, so that it does QR factorization of this. There is one small difference that is I have to come back to that Q word here. This matrix Q that times Q n was just rectangular m number of column vectors to m number of normal column vectors, but here I said Q n I am looking for square Q n that I am looking for is square. So, that is why that is what I was goofed up that time actually. I tried to do too many things at a time. So, let us come back to this once again. That time I had m columns and x was also had m columns. So, obviously, there is an upper triangular matrix 1 times first column and then, 2 norm 0 element, 3 norm 0 element dot dot dot like that and I get this, but x actually each matrix is of length n plus 1 here, so that the vector space has dimension n plus 1 out of which I am getting only m vectors, but there is nothing wrong.

I can have extra vectors also. Only thing is they will not be obtained from these guys, but if you have got this x 1 prime n x prime x 2 prime up to x n prime and from that I can take the space spanned by them and go outside, that still remain in the main vector space get a vector projected on. I mean project it orthogonally on this. Take the error, divide by the norm square of that and I will get one component and go on doing it. So, I can always construct. So, that means, I can make the number of columns same as the number of rows theoretically. Isn’t it? So, in that case I mean there is now no restriction on Q being rectangular. From here I went to where is that one more page was there. Yeah here just a minute.

(Refer Slide Time: 39:50)



$$d(n) = (n+1) \times m$$

$$Q = X . R$$

$$X = Q . (R^{-1})$$

$$R'$$

$$= [\underline{x}_1' \; \underline{x}_2' \cdots \underline{x}_n' \; \underline{x}_{n+1}' \cdots x_{n+1}'] . \begin{bmatrix} R' \\ - - - \\ 0 \end{bmatrix}$$

$$Q'$$

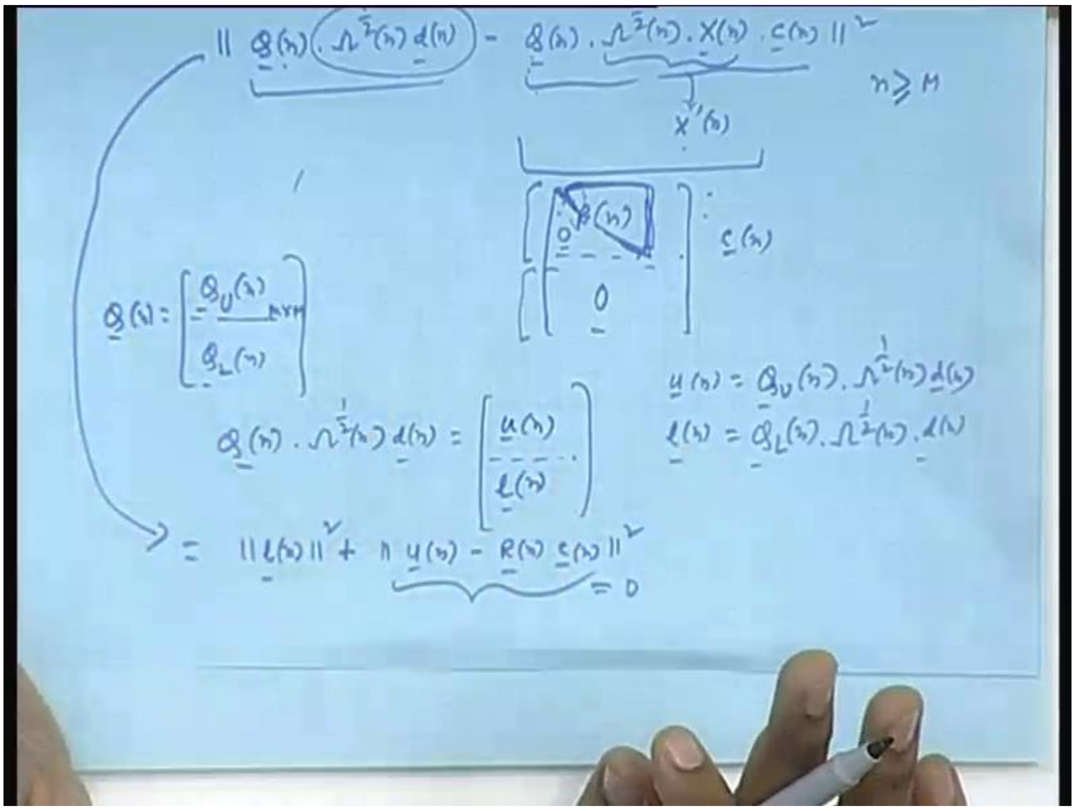
From here what I went to I think these here Q was n plus 1 cross m same as this from a QR minus R inverse in this form if Q number of columns increases, m number of columns increases. That is first n columns as it is and then, I generate the extra 1 by my own way. Then, you got m ortho normal vectors, but there is still some more left. If this way the total number of row is n plus 1, the vector space dimension is n plus 1, but you got me only n. So, n plus 1 minus m. So, how many extra vectors are generated. Then, I get a complete ortho normal basis for that vector space, so that I can carry out on my own. Isn't it?

Again that is not unique. I consider the m and then, take any one even though what anyone outside the space spanned by those m fellows. Take that project. It orthogonally on the space spanned by those fellows that error will be orthogonal and divide by x norm square to normalize. Again, this n plus 1th component. Again take this same n plus 1 fellows, take the space spanned by them, go outside that, take another vector, project it orthogonally, take the error and normalize it. That way you can. So, this Q can be made like this x 1 prime, sorry x n prime. This is what we had earlier and then, extra vectors. These vectors are coming purely from whatever I got here. That is why this side is not unique. This side is unique giving the data x and that way we generated that x 1. If you follow the stick to the data, start with x 1, then take x 2, then take x 3, then obviously you get this.

On this side you get, but then this matrix becomes square. What will happen to R prime? R prime was a square upper triangular matrix. So, now, that R prime will have that chunk of zeros. You remember I got the chunk of 0 here. It was just you know I mean a silly mistake. I without noticing that you know this is still rectangular and I square out of bad habit put the zero’s here. So, there is nothing. This Q can be this. The generalized Q you can say Q prime, this will be your x, but this is unitary. So, if I pre-multiply both sides by Q prime transpose, that is hermitian, then this will become and so Q prime transpose times x will be a matrix of this kind, where the top one is upper triangular matrix, and the chunk of zero’s. That is how to get such Q appropriately. I told Givens rotation is the one way which will see that technique, that algorithm I will go with the differential equation. I will come later. Let us find elements of this here.

So, I will generate a square matrix. Now, there is no problem. This can be square. I will generate that Q n which will be working on this rectangular matrix. This is a rectangular matrix. Total number of rows is n plus 1. N plus 1 column is only n that is a square matrix that will work on this.

(Refer Slide Time: 43:26)



$$\| \underline{Q}(n)\cdot\underline{\Lambda}^{\frac{1}{2}}(n)\,\underline{d}(n) - \underline{Q}(n)\cdot\underbrace{\underline{\Lambda}^{\frac{1}{2}}(n)\cdot\underline{X}(n)}_{\underline{X}'(n)}\cdot\underline{c}(n)\|^2 \qquad n \geq M$$

$$\begin{bmatrix} \boxed{R(n)} \\ \underline{0} \end{bmatrix} \cdot \underline{c}(n)$$

$$\underline{Q}(n) = \begin{bmatrix} \underline{Q}_U(n) \\ \underline{Q}_L(n) \end{bmatrix}$$

$$\underline{Q}(n)\cdot\underline{\Lambda}^{\frac{1}{2}}(n)\,\underline{d}(n) = \begin{bmatrix} \underline{u}(n) \\ \underline{l}(n) \end{bmatrix}$$

$$\underline{u}(n) = \underline{Q}_U(n)\cdot\underline{\Lambda}^{\frac{1}{2}}(n)\,\underline{d}(n)$$

$$\underline{l}(n) = \underline{Q}_L(n)\cdot\underline{\Lambda}^{\frac{1}{2}}(n)\cdot\underline{d}(n)$$

$$= \|\underline{l}(n)\|^2 + \underbrace{\|\underline{u}(n) - \underline{R}(n)\,\underline{c}(n)\|^2}_{=0}$$

So, entire thing should be equal to this form one upper triangular matrix of the form R n. This is actually upper triangular like these zero's here. This is often in paper and books you know upper triangular matrix is denoted like this is upper triangular, this is the chunk of zero's this times c n. Now, see things are very nicely considered. This kind is working on a vector, and giving you another vector, giving you another vector. This vector can have two components, the real time product vector. I can divide into two components. One component of this length n, another component that is lower part. Upper part and lower part.

I will write down later, but some upper part lower part that minus this norm square, just usual norm square and not conventional norm square. This lambda has been put inside that has to minimize this into cn outputs. Now, remember this part times c n will contribute 0. So, this lower half here of this vector minus this is independent of c n. That will always be present in the norm square. This is not I mean removable, and this upper half minus this into c n be remain exactly equal to 0. We are solving equations only. Upper half part and upper triangular matrix with norm 0 diagonal entries times c n if I suppose solve it.

Solving the equation is very simple, and can be normalized. You will get upper triangular matrix. So, first element times, first element here is first element. Find out the first element and then, use it here. First element, take the next one and like that solving is not. So, you can easily compute these optimal filter coefficients by this method. Only thing you have to compute this fellow this vector, let me put in term in mathematical form.

So, this into this quantity was x prime n Q prime n and this quantity is c n, and suppose you write Q n as two parts. This Q n matrix, one upper part that will work on this will give you the upper half and lower part work same vector give you the lower part. Upper part lower part Q, upper n Q lower have you used n somewhere Q lower n. This is your m cross m and this is; obviously, you know the total length I am not writing it here, and then Q n what I just now said I am writing mathematically. Using mathematical symbol into this, this can be what this matrix you write like this.

This is upper half times this vector will be here. You can call it here I am not used in anywhere today. No, have I used two vectors? Here an upper vector and this is lower vector ln. What is un in Q n. The upper part times this vector, this simple matrix multiplication. Upper part times that is this Q Q n times this guy and ln is. With this definition this norm square, this total norm square thing becomes equal to this, entire thing becomes equal to what mind you to this norm square is the usual norm square. Lambda is no longer present. Taken lambda inside this is a conventional norm square. This norm square means what total vector find each term square. All the terms add out of which this guy from this side, the lower half times c n will give rise to chunk of zeros that subtract from ln will give rise to ln only and upper half. So, that means, this total norm square will be plus this part un minus.

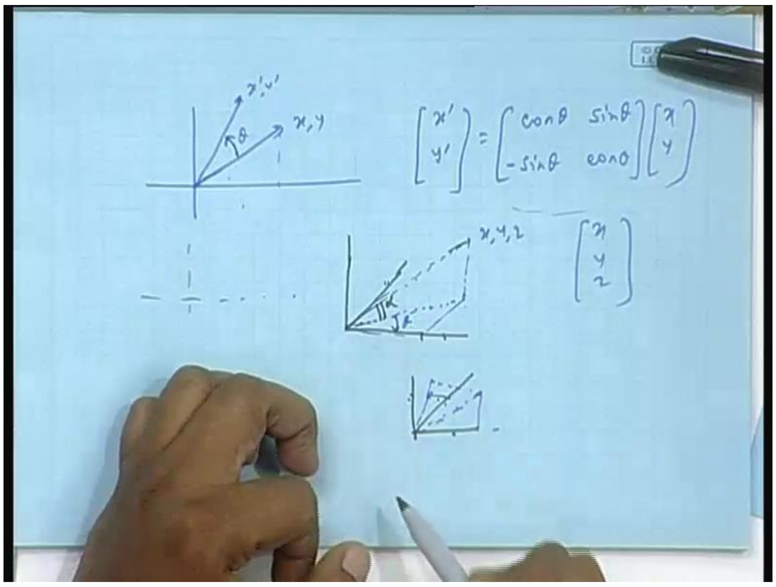
This guy I can always solve this. R n is invertible; R n is norm 0 diagonal entry. I told you see you can solve it and solving that will not take much time, say very structured matrix upper triangular. Only thing is it will lead to actually first you apply the solution for the first guy used. That is a pipe lining mechanism. In the first cycle if you find the solution for the first entry, next cycle you go for finding the solution for the second entry. Using those two next two elements. Then, use those two solutions to find third. It is a sequential thing. That means, pipe line we see detail that will actually give rise to arrays. Array you can see this very first thing. Once you solve that there means this is equal to 0 and this is the norm square. This is the problem.

Now, problem is I got Q n somewhere or the other side at index n plus 1. Again I have to find out. I need new Q n plus 1. I will get a new solution for c n, c n plus 1 there, but if I do manually every time, then you cannot do that. This process has to be adaptive from previous Q n or current Q n by some simple calculations. Generate the next Q that is Q n plus 1. Though effectively we will not be computing q n. We will be using we will be computing, will basically solving these equations as to made adaptive. So, we will be basically you know concentrating Givens rotation is one way which is very effective here to find this Q or decomposition, but there are other ways also.

There is something called householder transformation, there is something called recursive modified transmit. They all lead to various forms of adaptive filters, all leads to systolic arrays. Givens rotation is one way these are all algorithms. QR is given. There are various algorithms to operate that QR Givens rotation is one which is very elegant. Physically what is happening here you know can you see what is happening here those other ones? You are actually given a set of vectors, n vectors and only you are orthogonalizing them and then, going further and orthogonalizing that overall norm square will be norm square of those fellows plus norm square here.

Norm square of those fellows that I cannot touch, but here I remain in the space I can find out coefficients c n, so that there the error can be made equal to 0. That is what is happening. When I go outside the space, then that does not depend on c n also that in equation. That is what is happening.

(Refer Slide Time: 52:46)



x',y'
θ
x,y
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
x,y,z
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

So, four minutes rotation you know though it is school level stuff, but you know I forget often. Suppose there is a vector x y, you rotate it by theta and you are here x prime y prime. What is x prime y prime? You know all these or you know how you call the length p. So, this is if you give this name alpha. P cos alpha is x p sin alpha is y, and then here same p p cos theta plus alpha is x prime p cos p sin theta plus alpha is y prime. Expand them and use those facts that p cos alpha was x p cos sin alpha was y and you get a formula, but what you get is this theta. This is the add over which is going in this direction also. Cos theta, yeah has to be I can see it from unitary matrix.

So, you see two-dimensional world. Every vector when you rotate, you get another vector by this formula where theta specifies the rotation angle. This is the linear operator. You can see it as unitary multiplied by its transpose, you will get a matrix. You see this if you call it give it any name multiplied by its transpose cos square theta plus sin square theta will always be 1, otherwise plus minus usually come up and go to 0. It is a unitary matrix. Now, rotating by theta in this direction, you all agree it will give me the same thing. If I rotate, if I hold the vector as it is and rotate this axis in the opposite in the two axis, the system 90 angle will be 90 degree orthogonal, but that two are rotated backward by theta. Still the total angle will be theta plus alpha and therefore, the x prime y prime values that will come up, they will be the same.

In Givens rotation, suppose we consider one vector. I am not even close to Givens. One vector of length 3 I have switched off mine. Good musical interruption. Suppose I have a situation like this. There are three-axis and there is one vector given. I do not know pictorially. It might be I do not draw these diagrams very nicely. That is my problem. Suppose it is given xyz. Just few minutes, you project it here. This much is x, this much is y, this much is z. Suppose this angle is theta, and this angle, this is the projection. So, this angle is theta, this line is in the plane x y plane, this is the x y plane, this is the x y plane and this angle say alpha.

Suppose I rotate this vector backward by theta. Backward I mean by theta as shown because this vector could have been here also. Backward forward I should not use by the theta shown or effectively I rotate this plane, these two axis maintaining 90 degree angle between them orthogonality in opposite direction by theta first. What will happen? This will align with these and therefore, this particular line will be in the xz plane. Then, will be there. I cannot allow to draw it any further. So, I cannot draw. So, I will again pick up from here exact plane. Just hear me out. It will be in the exact plane. So, it will be like this. You know this and this fellow will come in this plane, exact plane.

So, now in this new coordinate system, this vector will have at least this one coordinate equal to 0. Isn’t it? When I rotate it, so y 1 coordinate is 0 and then, what will happen is you take the projection here on this fellow, on this plane. This is the vector. Keeping these axis as it is if I rotate these two now, so that this guy is aligned here. Then, finally what will happen you know what I am by this procedure, what will happen is this axis will finally come along this. This way rotation and this way rotation. If that happens, then in the new coordinate system, it will be some value. Then, 0 0, then it is a two stage Givens rotation. These has to apply to other vectors also. I will pick up from here in the next class one minute.

So, tomorrow we meet at 3.30 here. I do not have much time you know. I need some 2 hours at least to complete this. It is an interesting topic. Thank you very much.