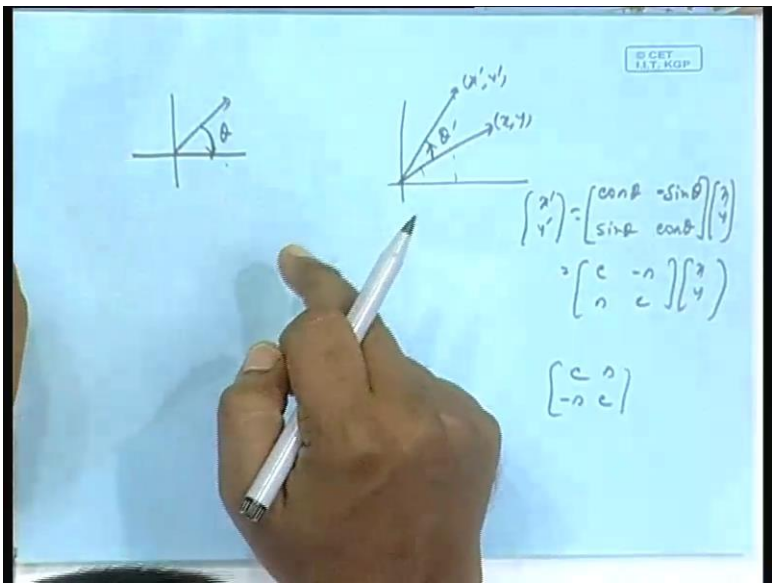
Adaptive Signal processing
Prof. M. Chakraborty
Department of Electrical & Electronic Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 37
Systolic Implementation

(Refer Slide Time: 01:10)

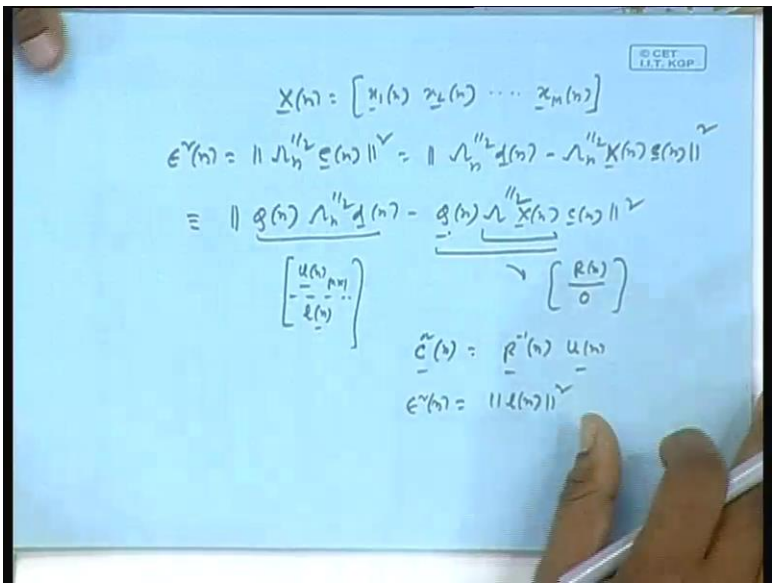


One thing, I just forgot to mention, you are considering rotations, elementary rotations; there is a vector x, y and we move it to x prime, y prime. This angle is theta. Then, we can work out x prime y prime, actually is, if you do not know how to work out, you call this angle alpha. This alpha plus theta length is p. This new x prime is p cosine, within bracket, theta plus alpha, and cosine theta plus alpha, there will be minus sign, Isn't it? That is why; if you work it out that way, it will be cosine theta, then minus; in my one of my earlier lectures I skipped this minus sign here; cos theta minus sin theta, this is important.

Cos theta minus sin theta and then sin theta cos theta. This, I can write down because this is unitary. If I move 1 row from orthogonally taken into the other row, this times x, y that is c minus s, scxy, but as I am following this move, you know that theta, that we consider typically, this is the opposite direction like, there is a vector here while you do rotate it. So, that it coincides with this axis, so that I am rotating this plane in the opposite direction.

This theta is usually, this is a convention theta. With that theta, cos theta will remain as it is. Because cos theta cos minus theta, they are same. Only this will become plus sin theta, this is minus sin theta cos theta. So, the convention will be cs minus sc and angle is in the clockwise direction. Yes, clockwise direction. Just keep this in mind.

(Refer Slide Time: 02:56)



Let us just go back to that what we are doing. That is we had xn, m columns, and we are minimizing this quantity. Epsilon square n, that is a norm square of this, you know very well, this is the usual norm; the Euclidian norm; just sum of squares. Lambda has been pulling in here. This was minimizing and this was giving rise to en, it was giving rise to dn minus Xncn norm square, but it is also equivalent to unitary matrix, Qn, we choose an appropriate unitary matrix square matrix. This is same as, this what we did at last time and what did you do? Qn, just Q, this vector, this works on this. So, that this reality thing is equal to the some upper triangular matrix and then zeros. Existence of such things; we have already proved we have discussed using (( )) and all that that is always global. Rn will be perfectly upper triangular matrix with non 0 diagonal entries; that is invertible, if you have got Qn, you have got this matrix having linearly independent columns.

So, QR will be perfect unitary and this will have on the non zero diagonal entries and in particular all that. So, this into cn and this vector we considered; we considered the upper half. The upper half will be given a name; un is m cross 1 and lower half, whatever be the dimension, this what we gave and therefore, this led to what, optimal filter; c cap n, this will minimize it, that will simply, if you take the upper half and solve this equation

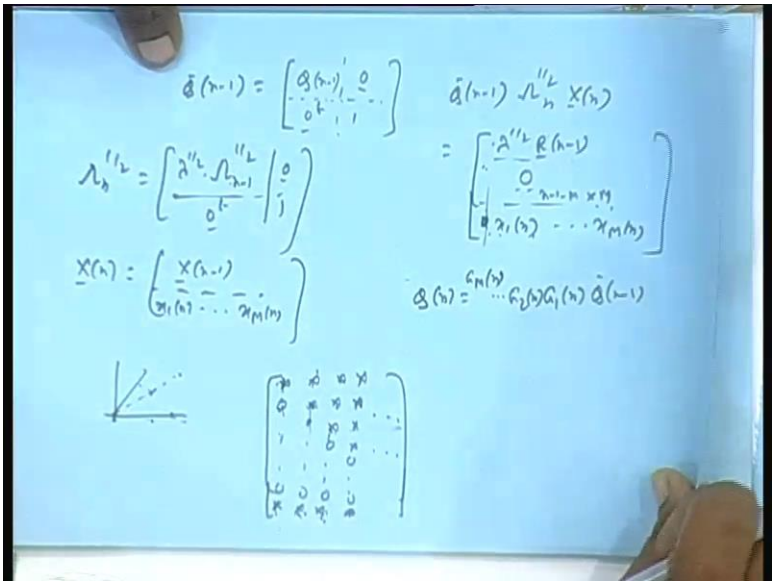
Rncn, equal to this. Because, total norm square, as you know, will be norm square of plane, this ln plus the upper part, isn't it? Ln minus 0 from here, that is ln minus Rncn, so total norm square is the summation of, the sum of norm square of ln and norm square of this, un minus Rncn.

Since, Rn is invertible, you can make, and ln is independent of cn. So, only thing you can do is, you can make un minus Rncn equal to 0. That will be the minimum possible and obviously, that will give the best filter. Because, that is the orthogonal projection and projection is unique. This minimizes normally, that will indeed corresponds to the orthogonal projection.

See; find out c cap n as R inverse n, your un of course, you do not have to compute R inverse n. Rn is triangular upper triangular. So, Rn into c cap n is equal to this. You can easily solve by back substitution. One value you find out, and then backward, you can do it in clock. That will give rise to an array, actually, pipeline manner. You can move it backward; back substitution. You find out and give it to somebody and take up the new data and again, solve it in that way.

It will not really amount to big computation having big complexity and all that. We will see that and your epsilon square n was simply this. Then, we did how to obtain this Qn we suggested Givens rotation and all. I am not getting into a recall of Givens rotation. I hope you recall Givens rotation, because that was quite graphic.

(Refer Slide Time: 07:11)



Now, suppose, we know Qn minus 1, then how to get Qn; we considered that also. That is, first you develop Q bar n minus 1 as 0 transpose, 0 col vector 1. If you apply Q bar n minus 1 on this guy, this will be what, you know this, is nothing, but lambda to the power half into delta, isn't it? It is a diagonal matrix only; 1 square root lambda, square of that square of that and so if I take the 1 ou,t take out square root lambda separately, this is this. This matrix times this means what, upper half Xn is what, Xn is x n minus 1 followed by 1 row consists of this x 1 n dot, dot, dot xmn.

If you do this, this into this, this part multiplies; this comes on top and this comes as a row. On that this matrix, this Qn minus 1 will work on lambda to the power of half to this into this only. So, that will give rise to a upper triangular matrix and chunk of zeros; that means, and multiplied by this. So, that will give rise to lambda to the power of half, Rn minus 1, then zeros. This is m cross m. These zeros will be n minus 1 minus m into m. Totally, then is up to this is n minus 1, out of which, m is gone and this will be your x 1 row, isn't it?

Now, I apply as I told Givens rotation. Actually, you see is Q is never required; first understand that Q is never required; it is required in analysis. If you give me odd n minus 1, I simply form at the new set of data, I have 1 slice, 1 row, whichever we stored that. This Rn minus 1, just lambda factor you bring in and you have to do Givens rotation. In the Givens rotation, you have to find out those cs factors because, by Givens rotation, I will annihilate this. It will become 0, but the c and s factors, I have to apply on the other elements now; second column; third column; fourth column. But those cs factors not difficult to be computed

First, I annihilate this. I applied Qa bar n minus 1 into this, I got this. Then I give Givens rotation. This is 0 and the first, this matrix has only 1 entry; here in the first and the last. Just, sum of the square root of the square of this, plus square of this; square of this plus square with length that will come here. This will be zero. This will give rise to a transformation for c and s, appropriately, apply that on the first row element and last row element of each other column. These last row elements will get modified; first row elements will get modified.

Then, you consider the second column. It is diagonal entry; second entry. That and the last one, there again, you rotate applying Givens rotation. So, the last part is annihilated, but that will not affect this 0 and this 0, because this 0 and this 0 is origin. So, you

understand that was the thing. Suppose, you had 1 vector here, you rotate the plane, so that your axis corresponds; I mean this corresponds to here; this has come back here, suppose. Then, if this has come back here means, only x coordinate yz both 0; 0 0, 0 0. There if you rotate the 2, this is untouched and these are 0 0. So, if these are not both 0 0, things would have got changed. But since that is origin only, even if you rotate this plane, the origin does not get affected. It gets affected, you see, in the first column these two zeros do not get affected.

But other columns, third column, fourth column onwards, the second row entries and last row entries do get affected, because they are not zeros. You see the difference. If you consider, if I even though I am annihilating, may be, I consider an example. These are data; these are zeros; first you had your data here. First, you are annihilating this. Length of this vector is coming here, actually. And this fellow is getting altered, I do not mind. Next, you are considering this. You are annihilating this by appropriate Givens rotation. That will not affect this firstly. Because, they had the, I mean, this axis. This axis will be under that corresponding, you know, there will be 1 in this matrix. For this row there will be 1. This will not get affected, but second row and last row will get affected.

There comes the question, as far as, the first column is concerned, there the second and last will not get affected because this already 0 0, means origin, isn't it? I am rotating this plane say, in this case yz plane, but in the case of first column, no effect. Because they are already 0 and 0. If you pre multiply a matrix, which targets second row element and last row element, in this case, you will target 0 here 0 here, so reality will be 0. Physically, you are rotating this plane, origin remains at origin only. But that does not apply to the third, fourth and other rows; here, this fellow and this fellow will get altered. This fellow and this fellow will be altered, I do not mind. Listen this. Then again, I consider this third guy; third guy and this last guy; last guy will be annihilated. That will not affect. That will affect and then I apply the transformation. So, fourth column, fifth column, they are these third row and last row; they will and those elements will get affected, but not the zeros applied in the first 2, whereas, same logic so on and so forth.

This Givens rotation thing, we are doing sequentially. See the mode of operation; you are first doing here. Suppose that you are targeting the first column. What is the computation? You are finding out few things; 1. square root of square of this, plus square of this length of this vector. That is one computation you find out because, that will be the new coordinate here and 0 here, but apart from that, c and s you have to find out. C

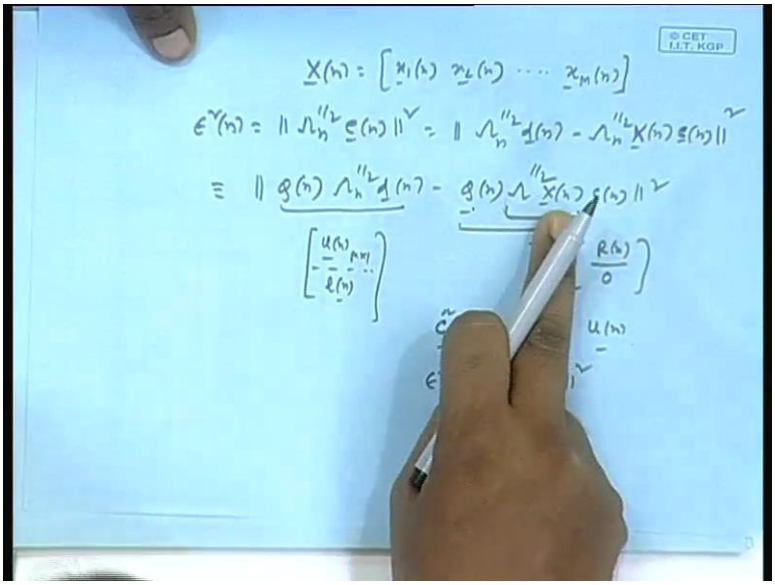
and s, at least, you know, base by hypotenuse and sin, what is that called? I mean hypotenuse that way you see this is one computation.

Once that is done, then you propagate c and s to this column, this column, this column and all these columns, so that, first and last guy are altered; all of them. This is one line of computation. Next time, you consider this again, you do the same operation as you did here same kind of process here; same length computation; same c and s computation and then you propagate to the right hand side. As you do that, this guy which was handling this and computing the new length and c and s, it can take a new column. If a new column comes, something, it can do that.

Anyway, I will come to that. I am just trying to give the computation flow. I will give detail here after a while. This is the nature of computation. Computations are very much similar; what will you do for first column, similar operation do here, and propagate the c and s, that column was between these two, you target, find out the length. Using the length, you find out this, I mean this c and s and propagate. So, apply this Givens rotation; that means, your Qn is nothing, but Q bar n minus 1 pre multiply, you do that and then 1 rotation corresponds to first column.

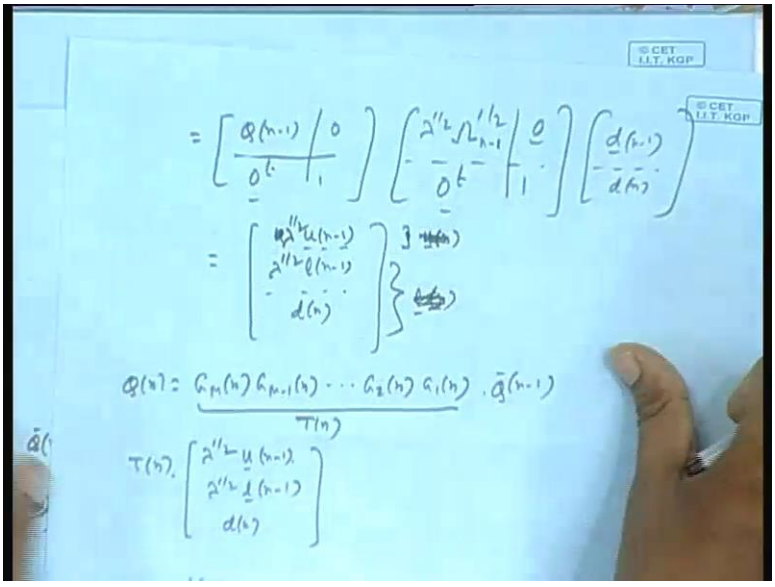
Anyway, you can call it G 1 n. You understand this rotation matrix. There will be a c, there will be a s, there will be a minus s, there will be c and all other diagonal entries will be 1, and elsewhere 0s. Typical rotation matrix, we discussed and where c, where s; that will depend on who, with particular column you are, like in this case, I mean that four boundaries c, 0 0 0, s and last column row also minus s, 0 0 0, c. Otherwise, it will be a diagonal matrix and so on and so forth. So, that is G 1 n, because I am making it depending on n I wrote, and then Gmn. Physically mind, I am doing like this, but I am never interested in finding out Q. No explicit computation of Qn is necessary. It is necessary in our analysis, you are not understanding. Computation does not need this. You give me Rn minus 1 and new data, I will rotate them by those computations.

(Refer Slide Time: 18:00)



And mind, Qn, I am not only applying on this. Qn, I am applying on this also, isn't it? Qn, I am applying on this also. What is the affect here? Let us find out, this also I have to find out. You are doing Qn on this by that method; the previous Rn minus 1 is giving you new set of data, just rotate. No need to compute Qn. Here also, I have to do the same thing, what computation? Is it not?

(Refer Slide Time: 18:25)



Qn, again let us see. First, Q bar n minus 1 and this rotation matrix. Q bar n minus 1 times this. What is Q bar n minus 1? Q bar n minus 1 is this. That working on this will be what; this lambda n to the power half. As I told, you can always write like this. Like,

these times dn and Qn minus 1 before that. So, these working on dn minus 1 like, 0 times the last. Forget the last entry, these working on dn minus 1. On that, Qn minus 1 works. There will be Qn, sorry, there will be Qn minus 1. This one, I forgot to write, Qn minus 1 and these working on, very simple, these working on last column into dn. This actually sorry, this is Qn minus 1 working on, no just a minute. Let me write the correct expression. You do not mind if I move to new page. This is your, let me write down step by step. So, this on this, on that Qn minus 1. These working on these; lambda to the power half is a factor, you take out first, this guy on this and Qn minus 1 on that; what will that be? That will be this vector; un minus 1 followed by ln minus 1. You are not following this. What was the un and ln?

If Qn walks on this vector, I take out the first upper m plus 1 part, I call it un lower part ln. If it is n minus 1, if it is n minus 1 and if it is n minus 1, what is this? That is what it is. Only thing is lambda to the power of half factor, has to be brought in now, isn't it? The last guy dn, this part is your, because this is m cross m, always following this. This is your ln. This is not e th variance, sorry, I have done only these. I have to apply that Givens rotation thing, that is the final matrix.

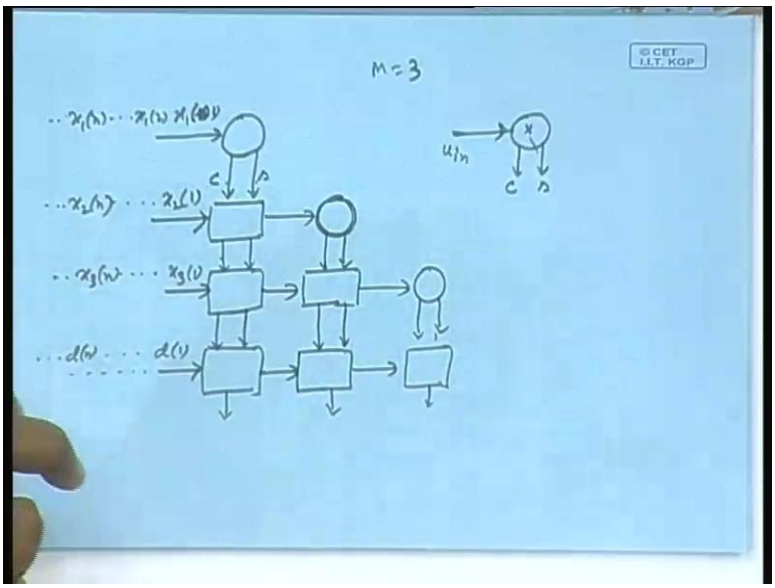
I am only applying Qn minus 1. I am only applying these. But this is not Qn. Qn has to be applied. So, this is not the thing. But what is our Qn? Qn is, this is called Tn, say let us say sequence of rotation and then that Q bar thing. That means, this vector, now you have to find out I can find out, this is Tn times this guy. Remember, I am not bothered about Q bar n minus 1, please see here also. I am not bothered about Q bar n minus 1 or Qn and all that, I have to find out these. You gave me what is n minus 1 and what l minus 1 is, that you gave me, like Rn minus 1, you gave me this fellow and this fellow.

But what I have done simply; I have brought in the lambda factor and appended the new data for desired response at the bottom. From this side, I am finding out the sequence of rotations, this side, those upper triangular, I am triangulizing this data matrix. I know what they are. And the moment, I know these things, this will be what? Just a rotation matrix, isn't it. One particular value will be c and s and correspondingly minus s and c. So, it will pick up just a pair of entries from this vector. This vector, if you apply G 1 n on that, it will pick up a pair of entries

In case of G 1 n, we know what it is; c 0 0 0 0 s; identity matrix minus s 0 0 0 c. C will work on the first guy and s on the last guy. And also that will come to the first position.

Remaining position is intact and minus s times first guy; c times last guy; summation that will come in the last position. So, just these also will be rotated; these vectors also will be rotated, isn't it, by which rotation angle, that which came from the data side should be rotated once by these, then rotated once by these, rotated once by these, like that. So, I have to just do these rotation operations. No explicit computation of qn is required. We will come to these computational things.

(Refer Slide Time: 26:13)



Suppose, let us consider m equal to 3 and now also, consider adaptive filtering problem, though it is not necessary here. For the time being, we assume that we are carrying out with our initial assumption, which our index of operation n is greater than or equal to M. So, that there is no linear dependence problem, that I have to take care of, mind you. Otherwise, things are not complete. How to take care of those indices, those early indices, where number of in n was less than m, then Qr does not take this, where number of columns is more than the dimension of the space. You cannot have Q matrix. You understand that. So, how to take care of that case, that we will see; that is another story.

But suppose, our index n is, we have all that. In that case, I will just give you a partial idea of the computation, then again, I will come to that initialization thing, where n is less than m. And there, I will consider adaptive filter problem, where these various columns of the data matrix will not be independent like x 1 n, x 2 n, x 3 n. It will be like what we did in the lattice case. First column; next will be z inverse of that; next will be z inverse 2 of that; z inverse 3 of that; that way, which is purely filtering problem.

At pre windowed case, that is 0; I mean, first column as it is, then next you will have 0 followed by data, then 0 0 followed by data; that kind of structure we will assume. And remember, that 0 0 structure is an upper triangular matrix, isn't it? That too make use of that, we will be using this. But there is no direct relevance, but there is upper triangular structure. We will make use of that, which is why pre windowed thing will be taken up. Just to give you an idea beforehand, there is computation. Suppose, this is a general case I am considering. But for the adaptive filter case, we will be simplifying it, x 1 0. For the time being let us assume what starting index is n equal to 1 not 0. Like, in the lattice case we started at n equal to 0 and went onwards. Suppose, n equal to 1 is the starting index, like that time I said initial condition at minus 1 remember, in the lattice case, I get the values at minus 1. Here there is a condition to get 0. This book also does the same. So, I just picked that from here, so first guy x 1, second dot dot dot dot x 1 n, dot dot dot; this is a current index. What it will do, do the businesses at current index please. Do not go to the starting index then that dependence independence will come up.

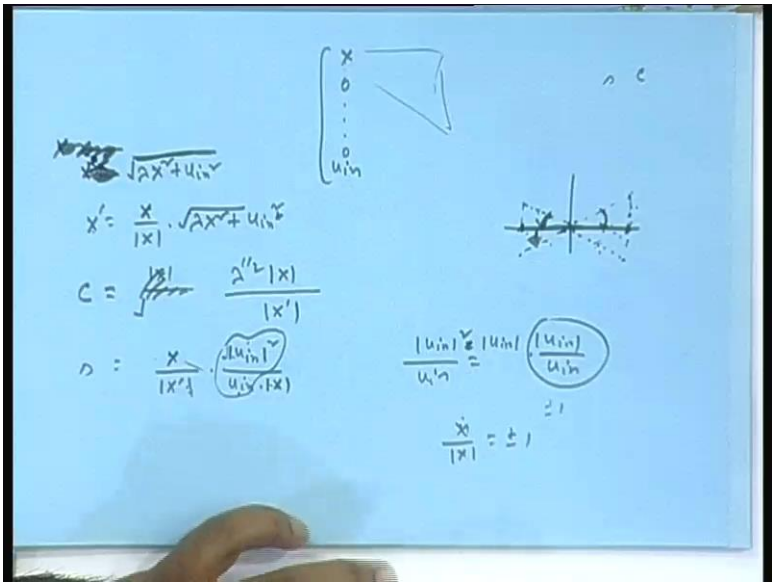
Suppose, this index is sufficiently large when doing it here, I am just giving you a brief idea beforehand. This is not final. Also, this will not be final; there are some other linear sections that I have to consider. I will consider tomorrow. But suppose it is so it will always assume the cs things. I will consider n equal to 3 only otherwise, I cannot draw an infinite array, m equal to 3. What happens, this kind of round; this circle; this is one kind of processor; this is another kind of processor. This processor will compute, will take up that particular row, where the 0, where the data has to be eliminated and therefore, c and s can be computed, like first column; there you generate cs and then spread it to remaining columns; remaining 2 columns.

So, this will just do rotation; this will do computation. After that, you target second column. Here, will be the one more thing, we will see all these. It will become a triangle array here. This will propagate, whatever I have here, this goes through. Same thing comes. Those who have done my course on VLSI telecom, they know this. Sometimes, data propagates through position. These are just broadcast, not broadcast, but they go through; same c same s is transmitted. I will give the function of these two processors separately. That is how you draw systolic, as you remember, if the order of processing unit you have, you draw it. It functions separately, you remember, dg and all that we used to do.

It is not that formal way of doing systolic (( )), where you are doing dg and transformation; this is just directly using intuition in things. You did not do any course; you should have done. That also was mathematics. Here it is d 1 dot dot dot dn. So, remember these guys, since I have told this for you, the same processor is used; same type of processor. What it will do? It will do some kind of c and s computation. It propagate them through this; operations do not change; processor is same; same processor.

This is not complete. There will be just 1 section for the triangularization, but this is not enough. But, let us just see the computation thing. What these guys do; this guy does; I am drawing its action. It has some storage also; it is storing some value x. What is this x; we will very soon see this. Suppose, x taking, I will be using the x many times. Remember, the x is just related with this particular processor. Same x, I mean, this x will not be the same thing and all just you find the x here. Suppose, it is un. It will generate two quantities, c and s. How? So, let us do some kind of analysis now.

(Refer Slide Time: 33:56)



It is like this; x and uin. What it will do? It will first compute x prime as what, this is the data you have to multiply by that lambda thing. This will be, suppose, the triangle part was there. So, lambda to the power half into Rn minus 1 and then this data. That is the situation. So, this is multiplied and then square square added, length is found out. Firstly, you should take, if you suppose this data is (( )) and this data is 0. Then, life is very simple. If this data is 0, it is already annihilated, is it not?

Then, your c should be 1; s should be 0; I suppose. C working on this, and s working on this; c is 1; s is 0. You do not have to really go through all this computation, in that case. In that case, your life is safe; I mean you do not do so much of computation. You are following me or not. If this is 0, this can be 0 or non 0. If it is 0 that is, if obviously, c equal to 1; s equal to 0; isn't it? This, you rotate by 0 degree; cos theta 1; rotate by 0 degree or otherwise, we will do it in the matrix form; just 1 times this; s times this; s times this; 1 times this; s is 0 0 times this; 1 times this; you get 0 comes back here. This goes back here. This is very simple case.

When it is not, you find out x prime. This is the length, then in fact, let me say that you consider these instead of x. This goes for x; not x prime. This goes for x. Unless it was x next will be replaced by this guy. Unless in the location the x was there, now x is to be replaced, but before that I am just saying this x. Now, but I will come to that later, just compute this, I will name it later.

Look at x. Certain things I did not say regarding rotation, that is, that will come. Look at x. x is what? x could be here, say, x corresponds to x axis and this corresponds to z axis or say, y axis; another x is at 0 and they are not present here. If this is here and u here, this is very simple case; theta in the opposite direction, isn't it? Theta in the opposite direction; c will be positive; s also will be, cs minus sc, that s will be positive. Cos theta minus sin theta and that give rise to that thing; theta negative means, it gives rise to, which is all given in the beginning. So, s was coming to be positive. That is the case.

But it could be the other way also. Then, what is the new value of the coordinate? Coordinate will be rotated. So, this length itself with a positive sign will be the new value here; this is one possibility. That is, if x is positive, then either you are here or you could be here. If you are here or here, x is negative. We will consider that later. If x is positive, either here; you rotate this way or you are here; rotate this way, but either case, this much length with a positive sign will be your x coordinate. You are either rotating this way or this way, but this much length with a positive sign. That means, if x is positive, then just this with a positive sign, will be your new x coordinate.

On the other hand, what happens to s here? S, we are rotating in the opposite direction in positive theta; counter clockwise. So, sin s will be minus then when theta was this way; I had plus s coming, when theta was this way; I will have minus s. Remember that. I am not targeting s at the moment I am targeting the c. On the other hand, it could be here and

again, I could be here or I could be here. If I am here, I rotate this way, but by how much angle; this angle only. And what is the x value? Negative, it comes out to be negative of that length. Square root is a thing, is a length, I find it is positive. So, I have to give a sign to that. Sign should be negative, if x is here. On the other hand, if it is here, even then I rotate, but sin remains negative.

That means, new x prime will be. How to take the sin of x fellow? X by mod x, you all know this. This is the convention, this analysis will be different. I am taking this way or that way because; this comes in the mind first; this way. No, what you saying is, instead of rotating this way, you rotate in this way. That could be done, I am sure. This is just a convention, I mean, there is nothing fixed about this. Make no mistake; there is nothing fixed about this. But, you have to follow a convention and stick to that. I am following a convention, given to basis guy, who is taking. I am just following it; I am not ruling out those things. But the advantage, this theta cos or sin computation, becomes very simple.

I am telling you why. If you just rotate only, because theta is always accurate. If you just rotate by this much; or this much; or this much; or this much; theta is accurate computation becomes, otherwise, you have to consider 180 plus theta, 180 minus theta, and all those things will come up. That is the thing. That is why; you are very smart guys you will understand more than what we understand here. So, this times, what was earlier x, that will be replaced by this fellow. Now c and s, c is, if you are rotating this way or rotating this way, or if you are rotating this way, or this way, c is always positive. Isn't it? Because c is cos, is it not?

Our convention was this way, but even, if the angle is in the opposite direction, cos of that is positive. Similarly, here counter clockwise and counter clockwise, both are c. Problem comes only with s. S again, both thing; whether in this direction and this is positive or negative, or this is in this direction; this positive or negative. s is more complicated. But again, you know that stage elementary trigonometry, all of us, not me. Certainly, you people are very smart on this. So, c will be what simply, I mean, do not bother about sin and all, just take the mod of the base, mod of the hypotenuse, the ratio, isn't it? Because, c is always positive.

That means, c will be your mod x divided by this quantity. In fact, it is not just x. There is a lambda factor. Mod x divided by, you can take the mod of this, or mode of the same way. This is your c. Remember, computation is in the processor, you need a squaring

operation; you need a square root operation. These are the computations. Then you will have done research, they modify this data; I am not going in to, where squaring operation and square root operations are not required, by again mathematical, I mean tricks and all that. This is just simplified thing, but those are beyond this course. Just giving an idea about QR factorization and that thing, it is very interesting thing.

Especially, when your number of input is very large, suppose 1000, then these arrays, because otherwise, if you get 1000 line thing coming, every moment of time, then so much of computation.

But if it is systolic and pipe line, there will be all pipe line, which you all know about that. I will come to the pipe line lattice. They know very well, I mean that you can apply cutset and (( )) and all that, that you know, is it not? Or you have forgotten. Cutset retiming and all you know very well; I can just apply cut set here; cut set here; cut set here; re-sequence the inputs and all that. About that I mean, to them, I will tell. It is not difficult because, there is no feed back here. So, cutset are, even if I do not use the term cut set, it does not matter.

Which page, where was I? Here. Now, comes the more difficult thing that is; s was, when I was considering this much, then s was coming and what was the s? Just sin theta. When angle was in the counter clock direction, it was just sin theta s was coming; no minus s. That s was this much by this much. When here, that is, if x is positive, this is also positive. Then, take the ratio of the, I mean, this by hypotenuse. That will be the sin. When this is negative, this is positive, then sin theta instead of s, it will be minus s. Isn't it. So, that time, you take the ratio of these two; minus will come.

When here, you have, this is negative and this positive. That time in the clock wise direction, plusses will come. But, when this is negative and this is negative, just a minute I made a mistake. When this is negative and this is positive, you are moving in this direction; counter clock wise. So, minus s, if I make mistake please correct me. And the other way, both are negative in this direction, plus s. So, you have to check the sin of both this part; x part and that uin, which is coming; uin is this along this axis. I am just giving you from the book, s should be, let us see.

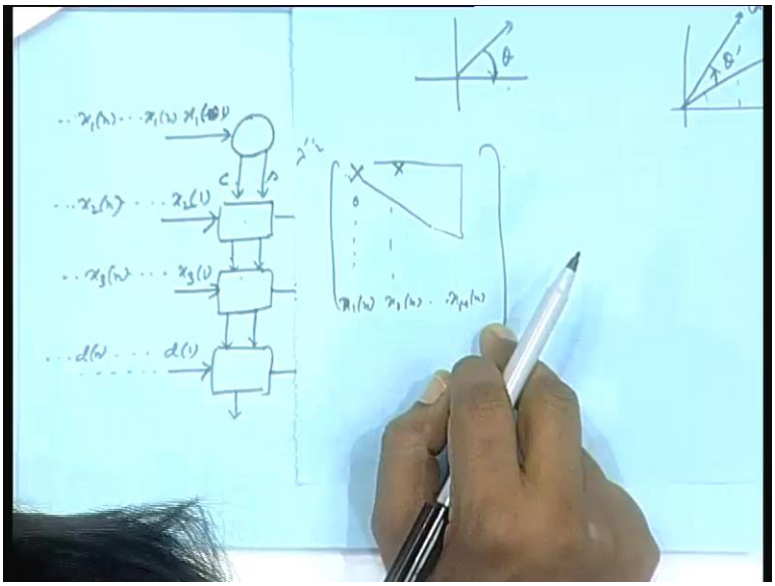
This is x by this means, if the x is in this direction, this is plus. In this direction, minus simply. In fact, I should take this ratio, mod uin. Let us consider this; mod uin square by this; what will be that be? This has the sin. So, mod, that is a mod for magnitude and if it

is in this direction plus, if it is in this direction minus, that will come up here in this ratio. This will be plus, if x is plus. This is minus, if x is minus. This ratio, mod uin, forget about all sin, mod uin by hypotenuse; that will give that magnitude of the sin.

Now you check, you will get this thing, that when both are positive, both of that and function kind things will come up, isn't it? Just you check, x by mod x, this can be the plus 1 or minus 1, and mod uin, this mod uin square, you write as this; as mod uin mod uin. And this by uin, you write as, this into uin, this quantity, is plus 1 plus minus 1; this quantity also, x by mod x also, plus minus 1. So, both the signs will be taken care of here, as per (( )). But here, we have to take the sin of uin, just uin. So, I tell, it looks like, anyway, it is very simple otherwise, I mean, there is nothing conceptual here. C and s are computed. Let us come back to array; what this array is doing, mind you, this is not complete story.

This is just part of that entire optimal computation, just that diagonal matrix of the R matrix will come over here. That is only thing I am saying, I need to complete this by tomorrow, actually it is not one day affair.

(Refer Slide Time: 50:25)

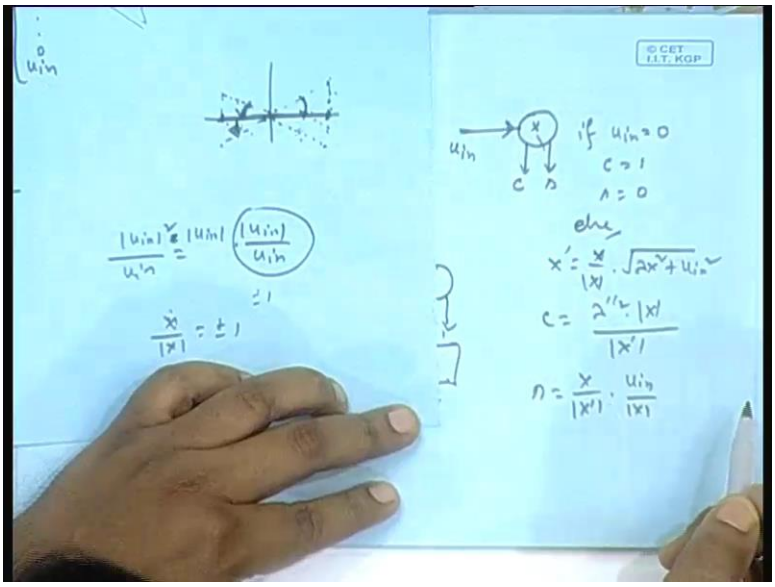


So, c and s found out. You have data x 0 0 0; this is upper triangle matrix given to you. New data has come up; x 1 n and then x 2 n; what you did, this guy did, it annihilated this fellow; replace x by that x prime and this appropriate sign and all this is here, stored, because next time, another data will come, next time square of that x prime multiplied by lambda plus this square, next value x 1 n plus 1 square root, again same operation will be

done. So, it is stored for using the next law cycle and still the cycle and all that. That is stored and in our mind, you know this guy is 0. We do not have do anything to implement that, this we know in our computation.

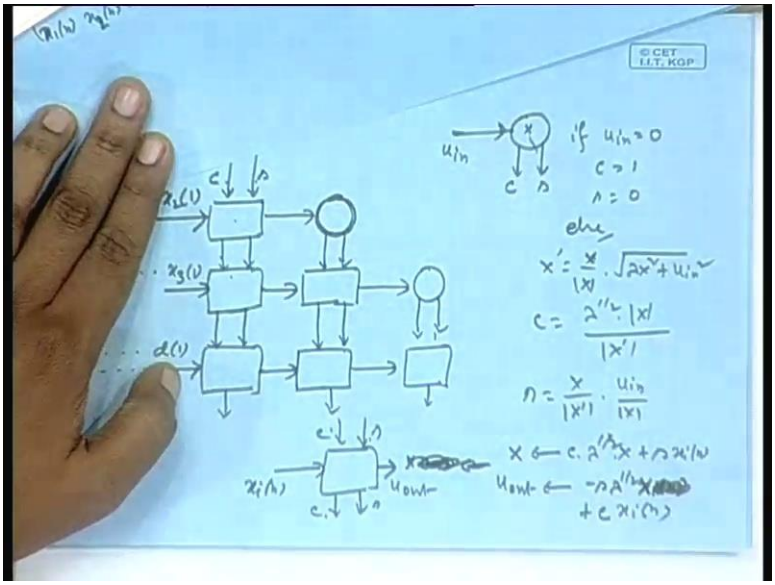
Only thing, new value, I am interested in new the Rn. So, new value is known. This cs goes to all these fellows. These guys, these people had, I mean, have their things stored. Like, suppose this, forget about this, to be x. Suppose, this you consider to be x. What it will do? This column gets c and s information and so it will find out cx. What will be the new things? Cx, cs and s times this; 149 and minus s time this and c time this. Lambda to the power half has to be here, on these lambda to the power half, because Rn minus 1 into lambda to the power half. So, always the lambda to power half, instead the multiply and activated all the elements here. That means what it will do?

(Refer Slide Time: 52:57)



It will replace this x by; this is the processor, as I was writing. Let me just quickly complete this processor thing; else x prime. This is just repetition of what I wrote. So along, you tell me what you want, I am not thinking, I will just follow you; x by x prime into uin by mod x. Well that will do. I am not checking, I have no time for checking. Both the signs are required. I am not so sure. This, I am writing based on the input of this guy. So, tomorrow I will see, if this is wrong, I will have his marks deducted from mid sem. This is the processor and other thing is this processor. I have to define this processor.

(Refer Slide Time: 54:00)



These processors, they take cs. They take new data; new data means x 2 n say, for this guy; x 2 n, there is the new data say, xin. Here x 2 n, x 3 n, like that x 2 n. And you know this guy. So, what it will do; this will be replaced by, x will be replaced by what, sorry, and something comes out. What comes out? Something we call this say, u out, its purpose will be seen, and something goes in this direction, this is c s; cc, ss; xin comes in and u out goes out. What does it do? It was storing some data x, that will be modified by new x. What was that x; this fellow. That x will be replaced by, I am blindly writing, this x will be replaced by what; c times lambda to the power half x, isn't it? c times lambda to the power half x, and s times this new fellow xin, that will go for new x here and what will be happened here? This guy will be called, u out, whether you should call u out n. But, I am just dropping the index, u out n, is just minus s times lambda to the power half xin, plus c times, sorry, minus this is x plus c times. This, we will consider in the next class tomorrow.

Thank you very much