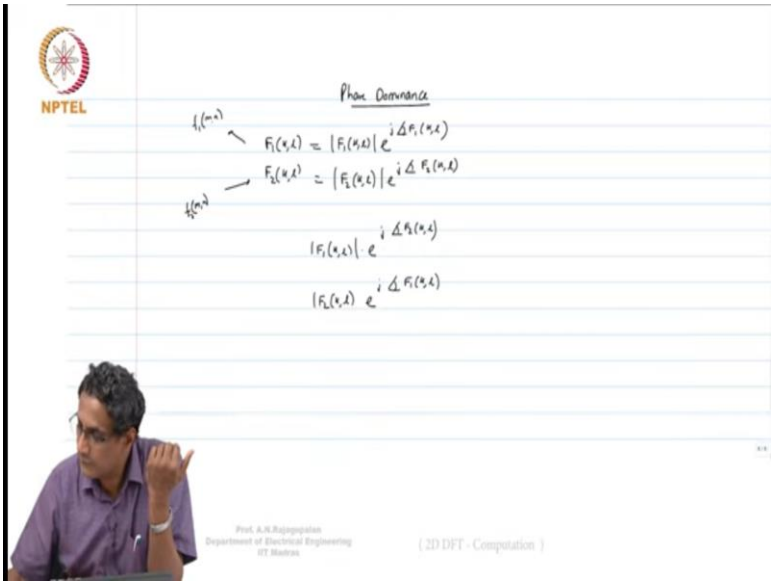
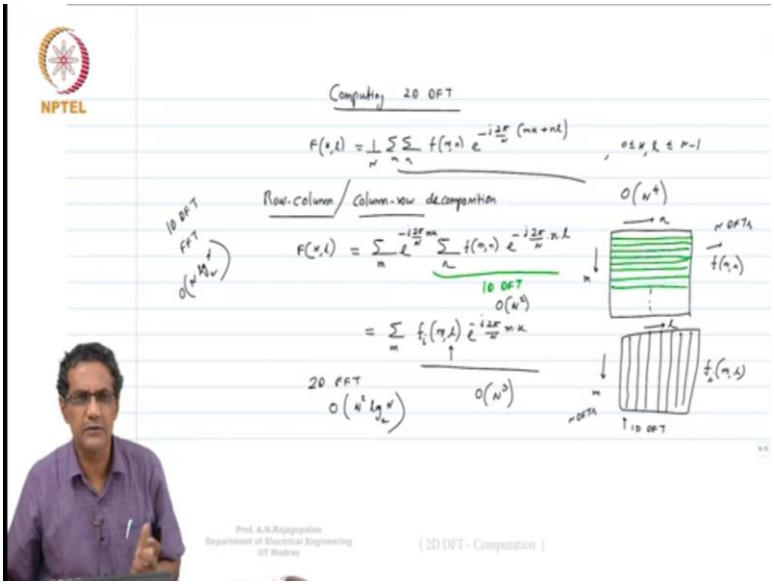
Image Signal Processing
Professor. A.N. Rajagopalan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Lecture No. 47
2D DFT – Computation

(Refer Slide Time: 00:20)



Now, moving forward I just wanted to talk about the how fast can you actually do the computation because it looks like when I do an F of kl.

(Refer Slide Time: 00:33)



So if you imagine, if you think of f of kl, so computing a 2D DFT, computing a 2D DFT, so if you look at it, if you look at the summation part f of kl, it involves a double sum, of course, maybe 1 by n. Double sum and then f of mn e raise to minus j, 2 pi by n, mk plus nl. The rectangle of will change a little bit but we will we will just stick to something simple. Now the point is, this this inner sum itself involves n square n square arithmetic operations because you have got to now do this multiplication and there so many additions and all that and then k and l are also numbers that are going to change from to computer to whole 2D DFT.

K and L also has to be computed for every value of k and l. F has to be computed every value of K and L. So, which means that which means that you are looking at something like order is the power 4. If you do it just like, 1D DFT, if you do a brute force 1D DFT, it is order n square but here because it is 2D DFT, it looks like you may have to you may have to do something like order n power 4 but then what you can what you can show is because of the separability property that that is inherent in this one 2D DFT so that kind of brings it down.

It does not have to be order n power 4 and then on top of that, the separability itself can be used now in the sense that because you know that the fast fourier transform exists for 1D and the fact that this this notion of separability is anyway there in your computer 2D. So these 2 can be clubbed in order to be able to able to really compute a 2D DFT very fast.

Well, I mean it is in the computation is of course. I know it is a kind of little higher than 1D DFT but certainly not like you need order n power 4 and all that. In order to compute that so in order to show that what what I will do so this is called a row column decomposition or a column row decomposition. Row column or you can even interchange the two or column row depends upon how you write it, column row decomposition and by the way, this is not this is not this is not only for a DFT.

In fact, all the transforms that that we have now, we have only talked about DFT but in fact even the other transforms are all separable. So, the very equation AU A transpose captures the fact that there separability and therefore this decomposition is not unique to DFT alone. So, this fast algorithms you can and the fact that row column decomposition can be used, it is something that you can exploit.

Student: (())(3:28).

No, by separable what we are saying is I will show now. While we do it, you will realize that it is not like they are independent. No. Row column, Column row decomposition. Now what this means is the following. Suppose I ignore this 1 by n and all. I do not want to carry this baggage but then you should always remember that if it is unitary, then that 1 by n should be there. Now suppose we look at F of, what is that, F of KL. Suppose we go with F of KL and then we have let us say so suppose suppose I suppose I take it off. Suppose I split this sum and then bring in terms that that involve only m outside then it is like e power minus j 2 pi by n mk.

This is the only this need not be sitting inside the sum over and, so then inside I get f of m comma n, e raised to minus j 2 pi by n into n into L. So, if you look at your f of f of m comma n so we are, the notion of m is this, the notion of n is this. So, here is our f of mn image. So, if you just examine this inner sum, if I if I kind of freeze my m, for a certain m, if you if you if you if you look at this inner sum it looks like if you freeze an m and if your index runs along n then it looks like you are looking at the looking at the m th row of this image.

If you fix m and then let the sum run over n, it looks like you are looking at this kind of a 1D row. This row which is like a which is like a which is like a 1D sequence. Now, which then means that means that if you look at if you look at an operation like this what it amounts to saying is that you can look upon this as computing 1D DFTs of all the rows of f because your n is varying. So, this is this is really a 1D sort of sort of a DFT operation and if you think that this takes n square, if you do kind of a brute force thing and so if you use a fast fourier transform, it is less than that.

But suppose you do a brute force thing, then it is like order n square for a let us say, 1 DFT. So, then what it means is now suppose we go all the way down so so we compute all of this and suppose we we create a new array now. Let me call this a semi intermediate array. Fi, now this fi is going to be if you look at this guy this will be a function of m and l. Function of m and l. So, m this way and l this way. So this inner, if I evaluate because I am summing it over n, so it will be some some lattice which will some let us say intermediate array fi which will be a function of so look upon this as m and then you have like fi of m comma l e power minus j 2 pi by n mk.

Now, now now what you can see is, if you freeze l now because the sum is running over m, then this looks like if you freeze l, then it looks like for all m you are trying to compute compute this quantity which then means that you are kind of looking at the looking at computing the DFT of

the columns of fi. It is not really independent. It is acting on the values computed with respect to the rows now and now on this new array you can think of either transposing it or you can just look upon that as the columns. So, it is like you did initially the rows, you did a computation which is intermediate array. Now, you are doing the columns of the array. These numbers can be complex now.

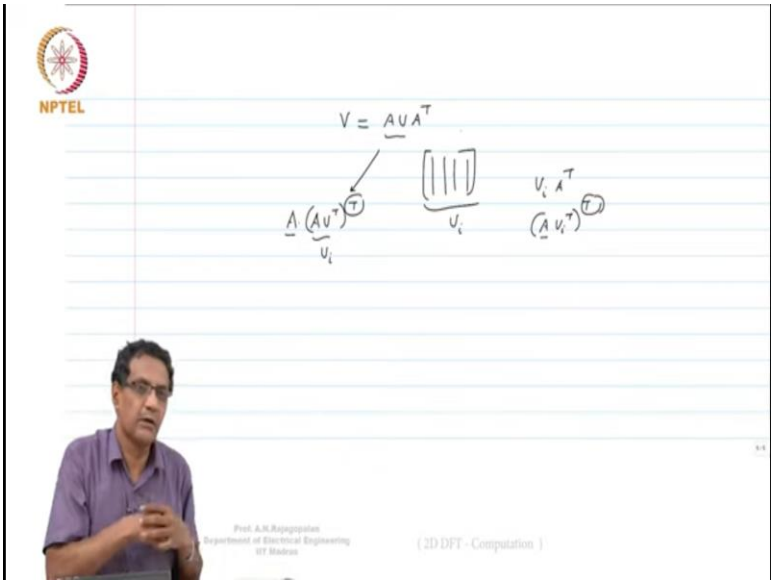
Yes, fi can have complex entries and even though you may start with F that is real but having taken a 1D DFT, all those entries could have become complex for all we know. Now, now what you are doing is, this is like this is like doing 1D DFT again. Now along along each column. So, what this is saying is like if I did order n square brute force. Now, it looks like I have to do NDFT is here for all the rows and then I have to do another NDFT is here for all the columns.

If you do that then I am done. Now, I will get my f of k comma l so which means that I have to do 2n DFTs 1D DFTs which is why if and this is happening because of this separability. I split that because this complex exponential is separable and therefore that is why I could pull that m out because or you could have done the other way. You could have pulled n out and then it would have become a column row. That is why I wrote this is row column or column row and the point is so so which is why you can you can you can see that it will require order n cube now because you have got 2n 1D DFTs.

So, your order of complexity if you just rule brute force, but if you did not put brute force f of k comma l that will be an order n power 4 but if you use the separability property, then it reduces to order nq which also means that now if you had a fast fourier transform let 1D DFT fast fourier transform 1D DFT. If you do a fast Fourier transform FFT if you compute using an FFT, then then that is what is this order complexity, n log n to the base 2.

Now you can imagine that you have got 2 n number of them that will break because of the separability and therefore, therefore if you if you look at the 2D fast fourier transform that will have order n square log n to the base 2. That will be the that will be the level of complexity. So, which is why if you try, if I give you a 1024 by 1024 image and ask you to do some brute force DFT computation you will be sunk whereas, whereas if you just run it through 1D FFTs and try to compute it, that is only way to do it. Nobody will do a brute force calculation, but but then what I wanted you to know is that this kind of a decomposition is in fact embedded in that.

(Refer Slide Time: 10:54)



This is not surprising in the sense that if you look at your that equation that we had when we said V can be written A U A transpose when you come from when you come from 1D to actually a 2D case. Here itself you can actually interpret it. In the sense that, think about AU for example, it is like, it is like you are operating so A is your, think of A as your DFT matrix. So, acting on on each of these columns giving you let us say some intermediate image array UI. Now what you have got is, UI A transpose. This you can write as A UI transpose, the whole transpose is, A UI transpose, the whole transpose.

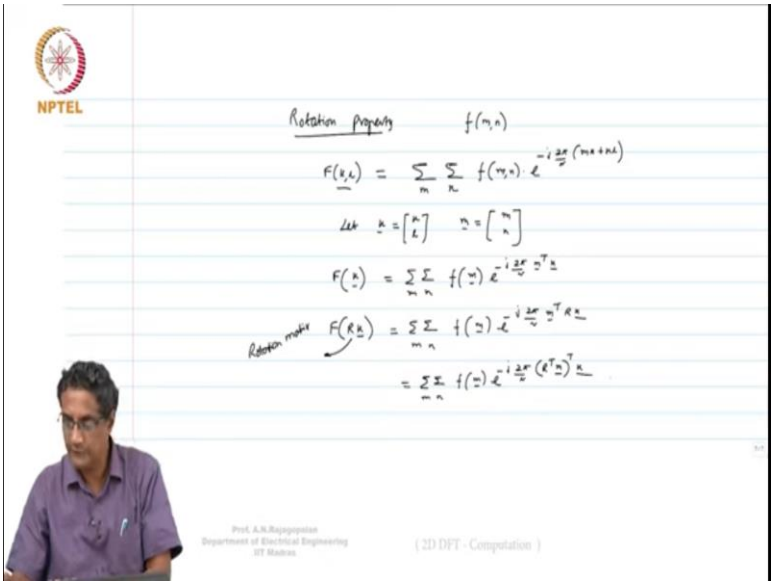
So, now this transpose of UI. That means which is same as column and then row. Correct? It is not A UI. It is A UI transpose and then again, of course you to try and do a transfer because you have to come back to the original, V and U should map the right way no? K comma L should be along like m n. Variation along m should be captured along captured l otherwise your interpretation will be.

So, that is why that is why this transpose on top will take care of that or you can also look at it this way, think about UA transpose S. What will you write it as? U transpose transpose. That is also fine. That is also UA transpose? So, now you look at it and so you look at AU transpose. This gives you an intermediate UI and and then and then and then and then you take the transpose and then you act A on that.

So, this can be interpreted either either through, so so this separate ability and this notion of row column decomposition or a column row decomposition this is kind of inherently there even in the way we wrote as A UA transpose, but for DFT, I just wanted to show explicitly but then from now on we will not show it explicitly for every transfer.

We just know that the very very manner in which we write a 2D transform unitary transform separable cases itself will lend itself to this kind of a to this kind of a fast sort of this one, this one a computation and most of the properties I do not want to, we do not want to spend all our time talking about DFT and all. So I am just going to go to limit myself to just one property which might be of interest which is actually a rotation property

(Refer Slide Time: 13:37)



Because rest of it and all is very similar to what you have in 1D and the goal of doing the unitary transforms is not to just look at the DFT or something. So, I will just run through this rotation property. What do you think will happen? What do you think should happen? If I if I have an image, m comma n and then it has something as Fourier transform and if I kind of rotate this image, what do you think might happen to the Fourier transform of the rotated image?

So, ideally what? I mean, you would think that the transform should also probably undergo the same angle of rotation. When you rotate an image and the Fourier transform ideally you would think that it should also it should also probably rotate. Is it not? Both phase and magnitude. In

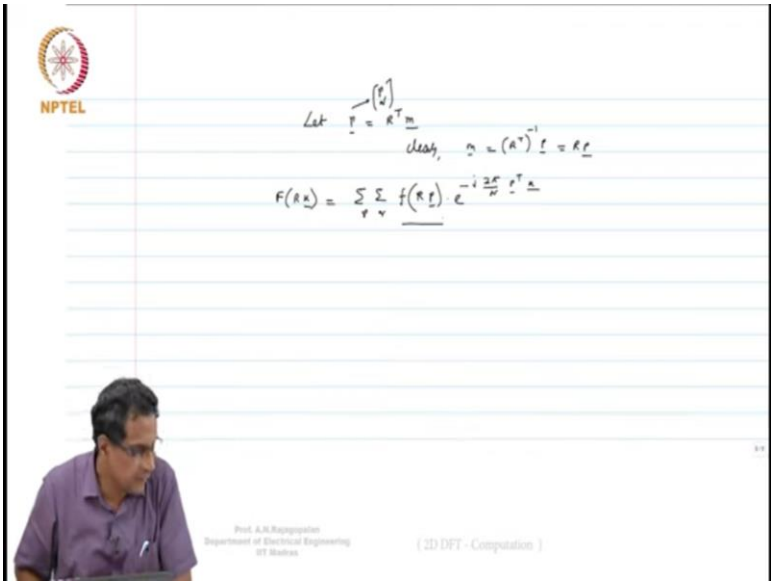
fact the Fourier transform itself should rotate in which case it means that the magnitude spectrum would also have rotated, the phase spectrum would also have rotated.

So to show this, we will just we will just show this today. So, imagine that we have f of k comma l. Let me see if I missed anything. So let us see, so if you have a f of k comma l and it is 1 by l and all. I am not writing but it is there. So, summation m n going from 0 to n minus 1 and then you have got f of m comma n to e raised to minus j 2 pi by n, nk nl. Now, let let us write this vector k to be k comma l so so this k comma l, I would just just to simplify matters and let me write some m subscript to be equal to m comma n.

So, this means that, I can write this as fk where this k represents that index k comma l, the summation mn, f of n e raised to minus j, 2 pi by n. Now, I can write this as m transpose k because this is mk plus nl. So, I can write this is m transpose k. Now, if you look at what will happen if I multiply by R and of course, this is a rotation matrix so, it could be like a cos sine minus sine cos or cos minus sine, sine cos, one of the two. So on the plane, you are trying to say rotate this.

So if you do this, this will turn out to be summation m n, f of m e raised to minus j 2 pi by n, m transpose and I have to replace k by rk. Replace on the left k by r k, or right l replace by rk. This is equal to summation over m n f of m, e raised to minus j 2 pi by n, r transpose m the whole transpose m transpose r and k. I will just rewrite it like that for, of course, a reason.

(Refer Slide Time: 17:09)



Now, let let us let us let what is that. what do you have, r transpose m. So, let some P be equal to R transpose M on the on the same grid. Then, clearly M is of course R transpose inverse p and I can talk about the inverse because this is a rotation matrix, RP. R transpose identity. So therefore, you have like f of rk . Now, we will come over the new sum. Let us call this some PQ and then f of, now M has to be replaced by RP e raised to minus j 2 pi by n and then R transpose m is p.

So, P transpose K and so equivalently so this F gets rotated by by R. So this, I do not have a figure to show here but you know, you can see any book and then they will show you this one rotated spectrum spectrum when you have for the for the Fourier transform.