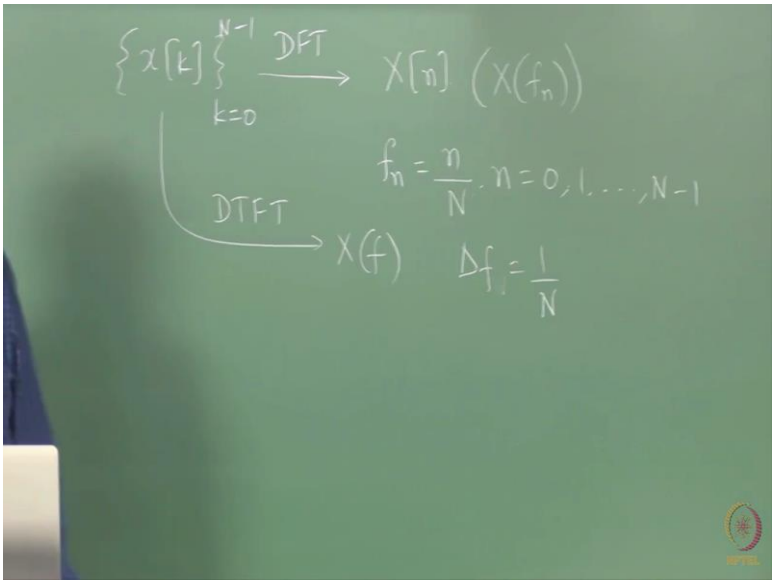
Applied Time-Series Analysis
Prof. Arun K. Tangirala
Department of Chemical Engineering
Indian Institute of Technology, Madras

Lecture – 69
Lecture 31 A - DFT and Periodogram 2

Very good morning, so what we are going to do today is we are going to discuss briefly on DFT. We have already started discussing that yesterday and one of the main things that we learnt yesterday is that the frequency grid spacing has got to do with the length of the signal and the minimum grid spacing, if we want to recover the continuous function X of f from the DFT coefficients, the grid spacing should be 1 over n and typically we choose 1 over n and when we do that we call as an n point DFT.

(Refer Slide Time: 00:59)



That is, if you have a signal of length N so what we have in practice is this signal length of length N and then through DFT, we compute X of n or X of f subscript n. So, you should notice that always we have use the lower case for the signal and upper case for the transform and k for the time keeping track of time and n for keeping track of the frequency that n you should all until you get into the habit of remembering it you know easily you should consciously make an effort to understand what is this n this n corresponds to f n and the n th frequency point is given by n over the big N, the big N again is a length of the signal and the small n runs from 0 to N minus and this is what

essentially DFT is telling you that you should be choosing the grid point as n over n that is that is what it is saying; you can look up the proof that indeed you can recover. Ideally we should be calculating dtf the X of f that is the continuous function through DTFT, but instead we are calculating this because of practical limitations that we have discussed already. So, this grid spacing will ensure if necessary that I will be able to perfectly recover X of f from X of n.

But we very rarely do that you know the other question that normally comes to our minds is what if I choose the grid spacing smaller than 1 over n; right now you can see that the grid spacing is 1 over n, I can drop the subscript here; can I choose smaller than this yes nothing prevents me from doing that, can I choose larger than this well you can, but then you would lose some information that is the point. So, there are some features in the signal that you would miss if you choose the grid spacing larger than this. All of this is done by default and FFT for you, but the user also has the opportunity to change the grid spacing in many of these expressions and there are a few other minor things that one has to remember when it comes to using this implementing this in any software package this is some minor things, but important ones as well.

So, if I choose a grid spacing smaller than this; will I get more information and the answer is no. You will not get more information, it amounts to actually in doing some kind of interpolation between the grid points, you cannot generate more information by choosing a grid space resolution better than this, you will only be ending up interpolating between two frequency points; that is point number 1 and point number 2 is what you see in the large body of open literature and even practicing aspects for example, if you go to certain websites which implement this Fourier transforms in a hardware fashion or even in a soft fashion; they will tell you that a common practice is to 0 pad the signal, to increase the length of the signal.

Now there is a historical reason to it, when computationally efficient algorithms namely the FFT; the fast Fourier transform was conceived by Cooley and Tukey in late 60s, it was required that the length of the signal be a power of 2. For example, it should be 32, 64, 128 and so on.

Now; obviously one cannot guaranty that always a length of the signal is going to be power of 2. So, what do you do right I mean no rhyming intended that, but what do you

do, so what people did is they said let me again do some kind of an extension within that already we know computing DFT amounts to doing what.

Student: (Refer Time: 05:22)

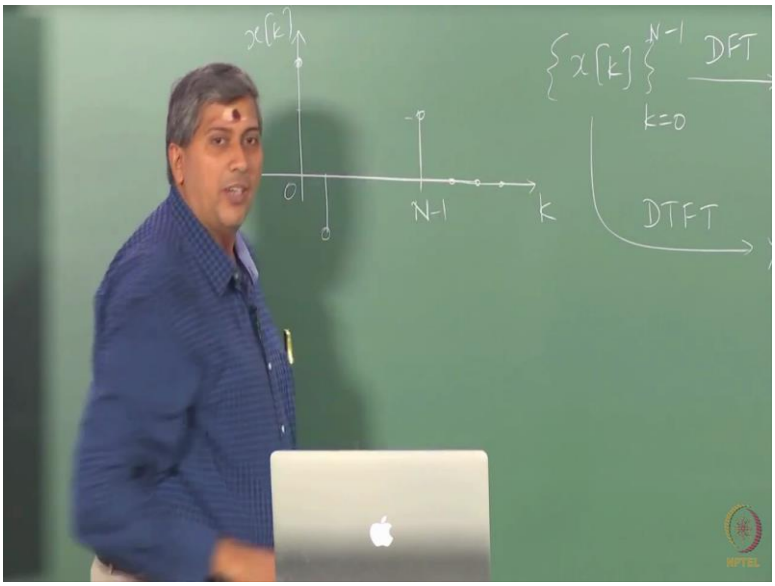
Computing DFT amounts to what kind of an extension for the signal.

Student: Periodic

Periodic extension, so that you should keep telling yourself until you know it by heart of course, you can formally prove it, but we do not go and go into the proof. So, you should remember that regardless of the underlying signal, whenever you compute DFT you are implicitly meaning that the long signal which we have not observed is a periodic reputation of this finite duration signal and with the period equal to the length of the observation, which is a bad assumption, but the badness or the error in that assumption decrease as n increases; obviously, as n goes to infinity then you are ok.

So, now coming back to 0 padding early in for a lot of a time in the sense for several years; may be even until 10, 20 years ago; it was required that the length of the signal be a power of 2, if you want to really exploit the computational efficiency of the FFT algorithm. Nowadays that requirement is not there, so people would 0 pad what 0 padding would mean is just pad zeros either at a completely at the end or half at the beginning and half at the end, do all kinds of what you call is [FL] right to bring up the signal to power of 2, but 0 padding can introduce a lot of artifacts that are not present in the signal.

(Refer Slide Time: 07:04)



Why because what you are seeing is here is this signal x k; which we have observed from 0 let us say n and at this point let us say this was the value right let us say at here it was some value here positive value and then negative value and so on, it had some values between 0 and n minus 1.

Now, what 0 padding does is that it assumes, it remains 0 beyond this up to the nearest power of 2. So, suppose n is suppose you have 100 observations; you will bring it up to 128 by padding 28; 0s right, now what is this 0 padding doing it is actually introducing artificial discontinuities at least of course, periodicity also introduces discontinuities, but the 0 padding introduces a more severe one and therefore, can produce some artifacts in your signal which are not present, they are just the artifacts due to 0 padding.

In fact, few years ago I saw an article; there are many articles in the open literature, but I saw this particular article illustrating what 0 padding can do what (Refer Time: 08:25) effects 0 padding can bring on the NI website National Instruments website because they implement FFT in hardware and software, but I am not advertising for NI; I am just saying that I saw this certain it was a nice article I thought, but there are many such articles in the open literature also telling you how harmful 0 padding can be and nowadays it is not recommended. So, you can just go ahead and compute FFT of any length, nevertheless you will find many text books talking about the 0 padding and hence I wanted to have a brief discussion on it alright.

Student: Sir.

Yes.

Student: sir you place the 0 padding you mention that if the sample mode number or times then what is require it is time. So, in place 0 padding why not take (Refer Time: 09:12)

Everything is possible, but what happens is all of this is see; that is a good question and that takes us back to two broad classes of data; one is called experimental data, other is called observed data; what is the difference between these two. In experimental data I perform the experiment right I am in charge of the experiment and I have access to the experimental set up I can get the data for you whereas, with observed data somebody has observed it and given it you; you may not even have access to the person who was observed it, in which case you cannot go back and say please give me more data, yes in places in situations where you have more data and you taken only a subset and you can go back to the original source and get more data, but many a times it is that you have only that data what do you doing. So, some imagination is required one; it is just like a film directors leave things to your imagination what happens after the end in any movie right here also the data leaves the puts that responsibility on you to imagine what happens after the end.

So, anyway it is a good question in situations where you can; why not you can go back and collect more data, but there are many situations where you cannot have that kind of a luxury and that is where this is all of this 0 padding business comes in, but generally I do not recommend 0 padding at all. Particularly now given that you have algorithms that can handle signals of arbitrary length.

So, let us proceed, so now, to summarize DFT is your finite length Fourier transform; you can think of FFT as finite length or Fourier transform also, but the actual abbreviation is fast Fourier transform; do not think FFT is another transform. FFT is an algorithm to implement DFT, when you compute the DFT at n points it does not matter how many points; n points then it is called an n point DFT and if n; by default this n is equal to the length of the signal because that is the minimum point number of points over which you have to compute the DFT.

Now yesterday there was a question after the end of the class;you know all of this seems to be theory where is a practice; this is the practicing one, the other four Fourier analysis different Fourier analysis that we have learned are all theoretical ones, but we needed to go through that theory do not understand what DFT is; we could straight away jump to DFT there are many short term courses or short lectures that do that, but it would be somewhat cruel to do that, it is better to start off with continuous time periodic signals and come up to DFT so that you have a full picture of how the DFT has come above, what is the history and story behind it. Otherwise you will not understand for example, the main assumption that DFT makes which is that the underlying signal is periodic right of length n.

So, you have the analysis equation on the top I have gone again, it is a convention that we have been following typically we present the synthesis equation first and then the analysis equation, but I presented analysis because in practice DFT is meant for signals that you have observed. So, the first thing that you would be doing is not synthesizing, you would be breaking up and then you would synthesis. Again here you can apply the same idea that is what I explained in the context of filtering, you can take the signal, perform the DFT right; that means, compute is DFT coefficients X of f n or X of n and plot the magnitude square what would the magnitude square be called the power spectrum; why would we call is power spectrum because we are assuming that the underlying signal is periodic.

Now here is where the catchiest in almost all the popular software packages, you would see the term power spectral density like you take MATLAB for example, it would have psd the as a routine in r of course, the command is nice spectrum or spec dot p gram we are going to talk of periodogram very soon, but spectrum is what you would see spectrum is more appropriate than spectral density. So, if you see in any package power spectral density, I will tell you what it is actually doing; it is not exactly computing some theoretical density because it is not possible, the DFT assumes that the signal is periodic, so you cannot think of a spectral density per sec; it is only a line spectrum that you can plot.

Anyway, so you would plot the magnitude square which gives you the power spectrum and then based on whatever you want and whatever knowledge you have of the underlying signal, let us say you want to recover the underlying signal, you are looking

at recovering signal from its measurement, what is the difference between signal and measurement well there is noise in measurement.

So, you would look at the h spectrum and then say look I know that the signal has the frequency, I am only going to retain that coefficient in my synthesis right exactly like what we do for example, when we wash our cloths, we know what is dirt and what is cloth. So, when we soak the cloth in the water medium it is like transformation you are transforming you taking into a new domain, why are we doing that because reparability is improved by leaps and bounds.

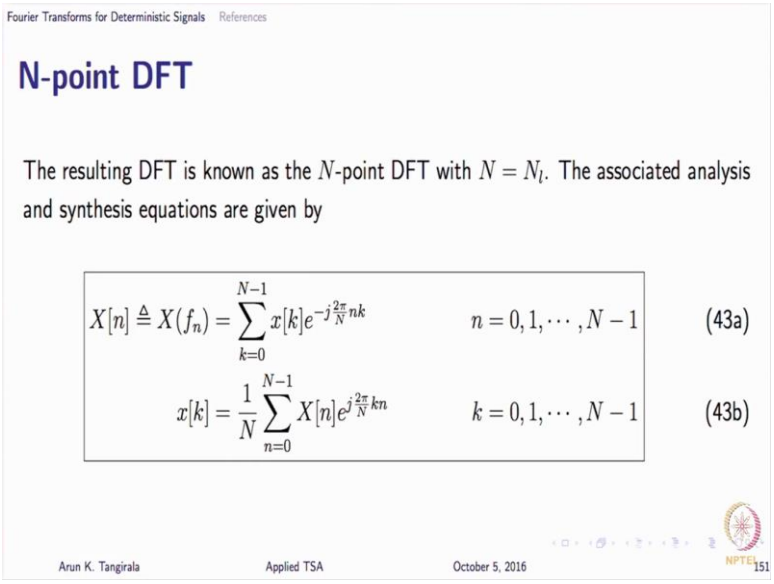
In air medium which is original medium, the dirt and cloth are not easily separable. So, in time domain the signal and noise are not necessarily separable and most of signals that we encounter that is the issue, but the moment you go into this new domain called the Fourier domain; the separability between the signal and noise improves like anything, why is it happening because this particular transform has the ability to look at the entire signal in time; if it is periodic then it will place that entire signal at single point in the frequency; what we mean by this is if the underlying signal is sine wave.

I will show you an implementation in r, but this discussion probably is good as a preview. So, you would search for peaks, if you know a priori the frequencies you would search for the peaks and at the time of synthesis you would throw away all the other coefficients, what we mean by throwing away is zeroing out. So, 0 out all the coefficients and then come back and use this synthesis equation in which case you would not of course, recover the measurement, but you would recover an estimate of the signal.

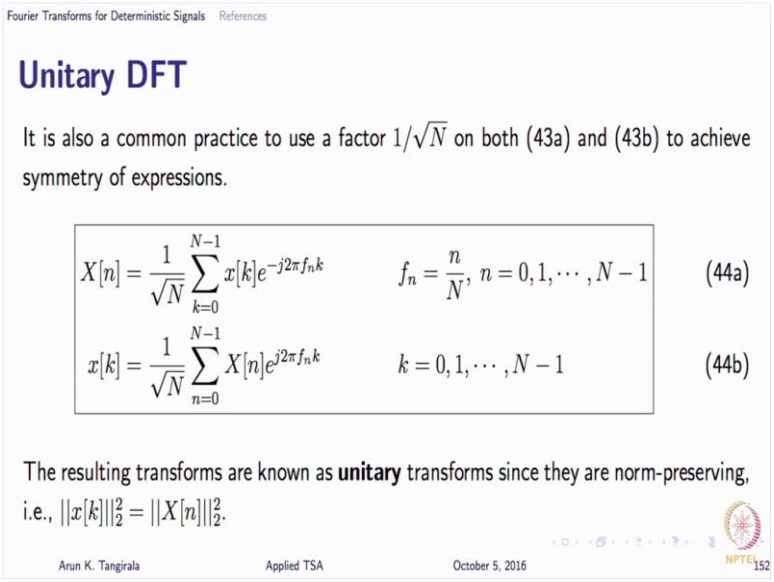
So, this is the classic idea that was proposed by Wiener long ago and this is the basic principle behind Wiener filtering and then of course, improvements have been made to that, but this classic idea remains at the heart of several estimation algorithms that are based on Fourier transform or kind of Fourier like transforms, you transform, figure out what are the most important ones in the transform domain, throw away the unnecessary ones synthesis; exactly like your laundry that is where the synthesis equation is useful.

Very often you would run into what is known as the unitary DFT; what is this unitary DFT, there is not much difference between this.

(Refer Slide Time: 17:05)



(Refer Slide Time: 17:07)

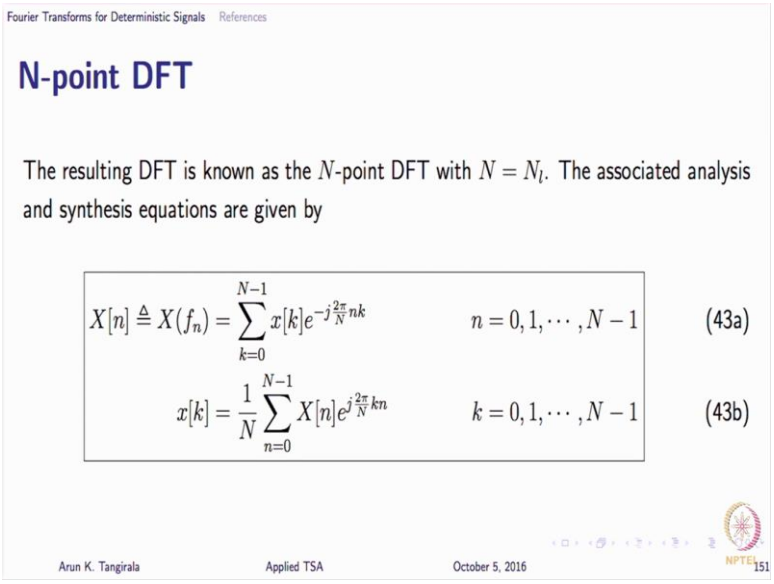


There is no new theory, it is not a new transform but you notice some difference, what is that; compared to this equation; what is the difference?

Student: 1 over (Refer Time: 17:08)

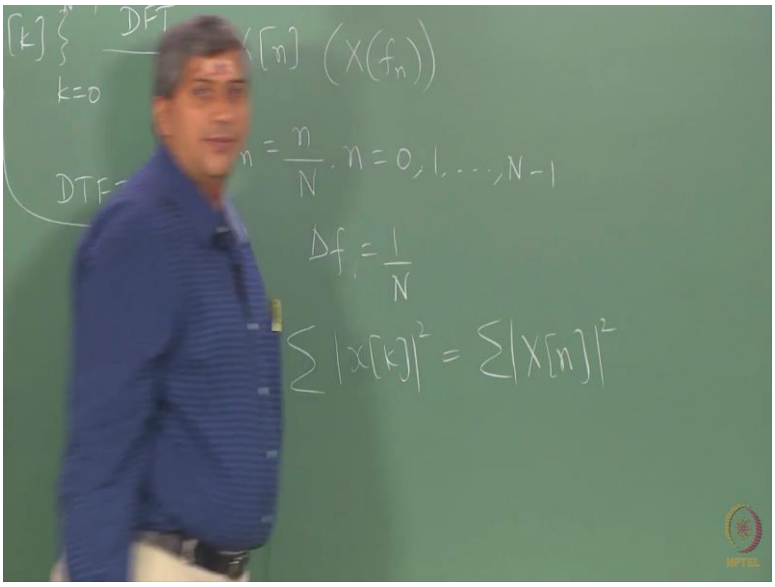
1 over root n why is that for people with memory loss problems not joking, but also it is a bit of joke and seriousness one of the advantages of working with unitary DFT is particularly when you writing in exam, you would be confuse whether the synthesis equation as 1 over n in front of it or the analysis equation.

(Refer Slide Time: 17:28)



If you go by the classical definition, 1 over n appears in the synthesis equation not in the analysis whereas, here there is no such issue; 1 over root n appears in both, so there is no confusion correct.

(Refer Slide Time: 17:58)



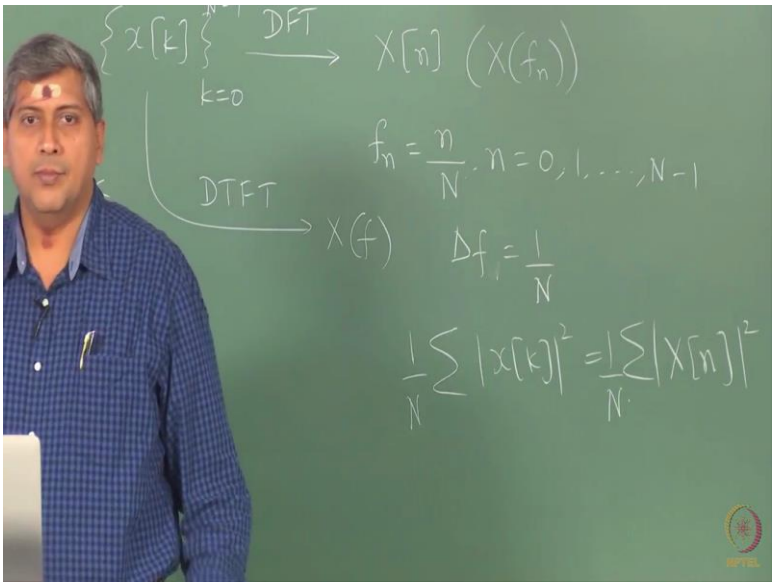
But apart from that the big advantage of working with unitary DFT is; the sum square of the signals is the same as sum square of the DFT coefficients; that is the big advantage. So, you do not have to worry whether you compute sum square of the DFT coefficients

or sum square of the signal both give you what; do they give you power or do I have to divide by something this is the energy over.

Student: (Refer Time: 18:25)

The duration right; this is the energy of the signal over the line duration of the signal that you have observed n, but we know that DFT assumes the signals to be periodic. So, it is meaningful to calculate the power rather than energy and what is the period that it assumes of the length; of the signal.

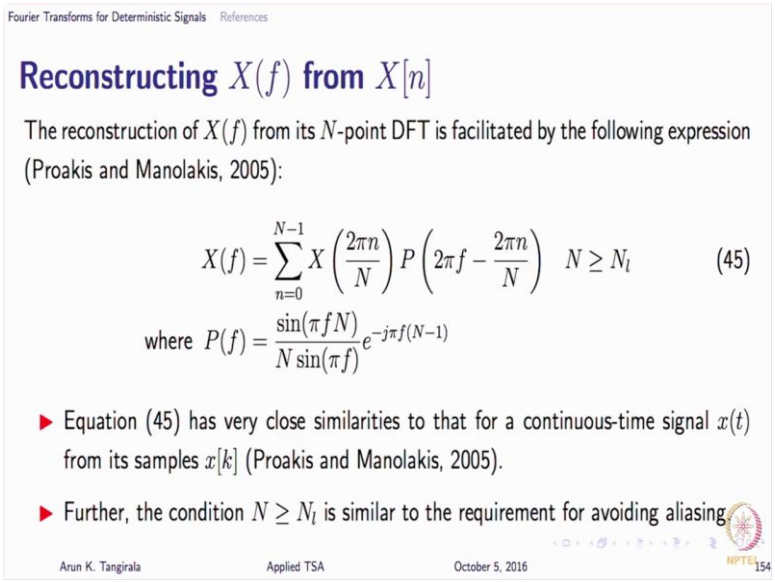
(Refer Slide Time: 18:48)



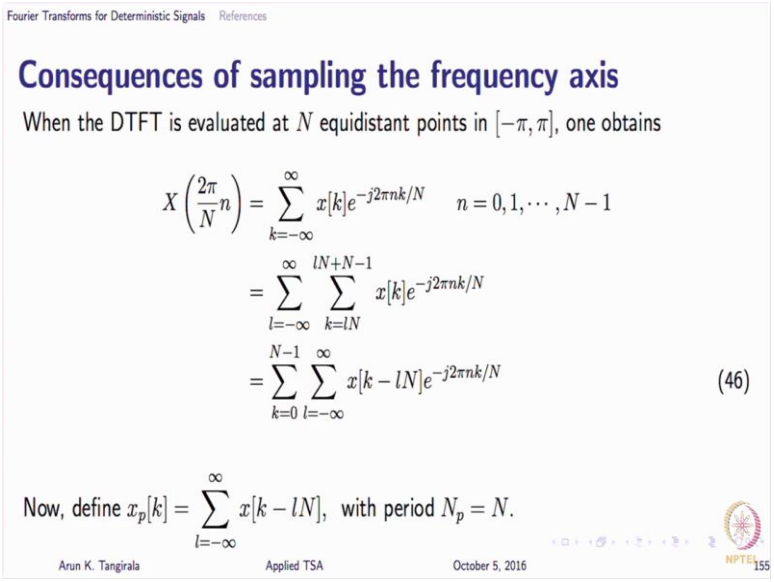
So, which means it is better in general to work with this; I mean, I am just saying that is got nothing to do the unitary DFT; the unitary DFT says the norms are preserved in both domains that is all the square 2 norms is the same; whether you are working with the signal or the DFT sequence does not matter.

Anyway, so that is something that you should remember therefore, two things when you are reading a text book; first confirm whether the author is using a unitary DFT or the classic DFT because some of the expressions slightly different they may miss out the factor and same goes to the software as well; whenever you use FFT go back and check in the documentation what DFT is it implementing; is it implementing unitary DFT or the classical one. Typically they implement the classical ones like the FFT in r and MATLAB and so.

(Refer Slide Time: 19:51)

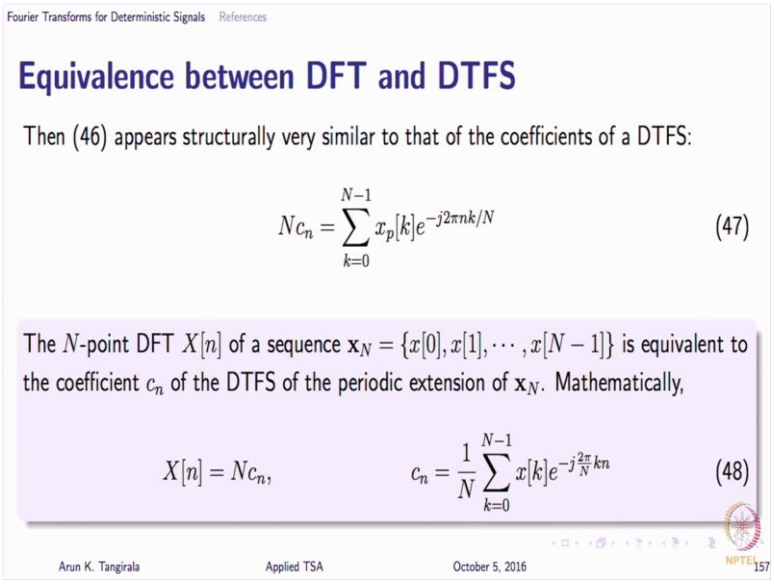


(Refer Slide Time: 19:57)



So we will skip this slide; this slide just shows you how you would recover X of f from X of n, this is not of interest to us; it is only for completeness sake I gave this slide and and in here what we are showing is that DFT assumes periodicity of the signal as when you sample the frequency axis, as the way we do in dft. So, what it essentially shows is sampling in frequency introduces periodicity in time.

(Refer Slide Time: 20:23)



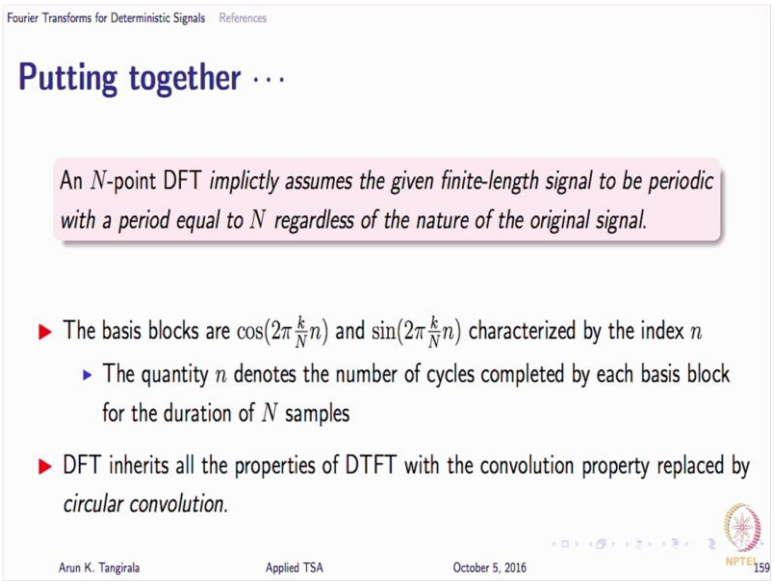
So I will not go through the proof but let us go straight to the consequence which is again sampling in frequency introduces periodicity in time. What does it mean? It means that it is actually; the end point DFT assumes is equivalent to computing DTFS; the discrete time performing a; constructing a discrete time Fourier series expansion of the signal. So, suppose I did not ask you to compute DFT instead I told you this is a periodic signal of length N; what would you do, you would construct discrete time Fourier series and in discrete time Fourier series, you would compute the discrete time Fourier coefficients. How would you compute the expression is given to you; I mean this is again what we have seen earlier.

Now, look at c n; what is the difference between c n and X of n, c n is the discrete time Fourier series coefficient that you would compute if I were to tell you that the given signal is periodic of length N and what is X of n; X of n is the DFT coefficient that you are actually computing and what I am trying to show you here is; they are almost one and the same except differing by factor of n; you follow. So, please distinguish between c n in r notion and X of n, cn is the notation that we have used for denoting coefficients of the Fourier series expansion and we use that for periodic signals.

So, if you are given that already that the signal is periodic of length and you would compute cn. If you are not given anything, you are just giving the finite length signal then you would compute DFT. All we are saying is they are one and the same almost just

by they only differ by factor, otherwise the expression looks alive; why did we do this I mean why I am striking a relation between DFT coefficients and the Fourier series coefficients because now we will shortly define what is known as a periodogram and I will tell you what is that. So the bottom line is now the DFT coefficient is n times the discrete time Fourier series coefficient.

(Refer Slide Time: 22:50)



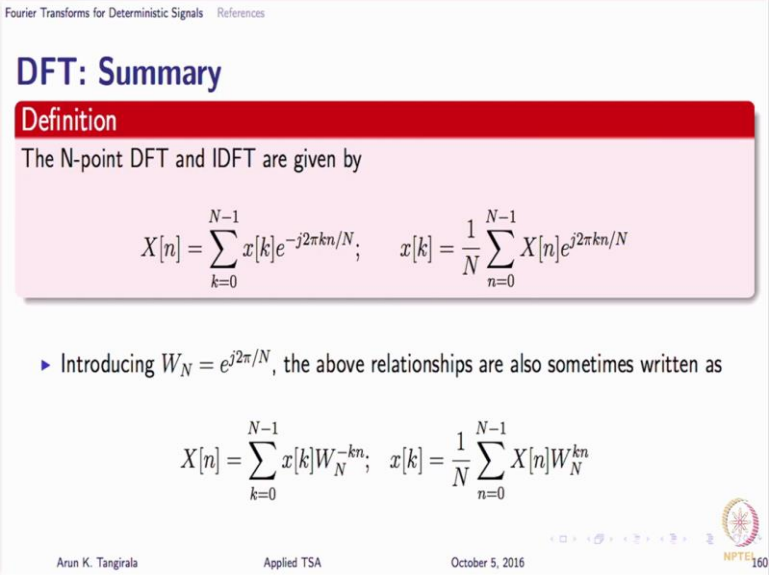
So, putting together an N point DFT implicitly assumes that the give finite length signal is periodic be the period equal to N regardless of what the true nature of the signal is. You would have observed may be 100; you would have for 100 points in time of an exponentially decaying signal; DFT does not care, it assumes it is periodic. Yes it is an alarming thing to know all along you been using FFT which essentially implements DFT without knowing that is the assumption that it is making.

And also couple of things that you should remember; essentially in the DFT the basic building blocks are cosines and sins and this n; small n that we have been talking of I have said earlier it refers to the n th point on frequency, but you can also give it a different interpretation which is that it is the small n denotes the number of cycles that your building blocks have completed over this length of the signal. For example, if I am looking at n equals 1, if I am computing X at small n equals 1 then I am breaking down the signal in terms of a building block that has completed 1 cycle in this length N. If I am looking at n equals 2 then I am breaking down the signal in terms of sins; sins and

cosines that complete 2 cycles and so on. Obviously, fundamentals and then harmonics, so 0 is 0 corresponds to the dc component, 1 corresponds to the fundamental frequency and then the rest are all harmonics that is what it is trying to say.

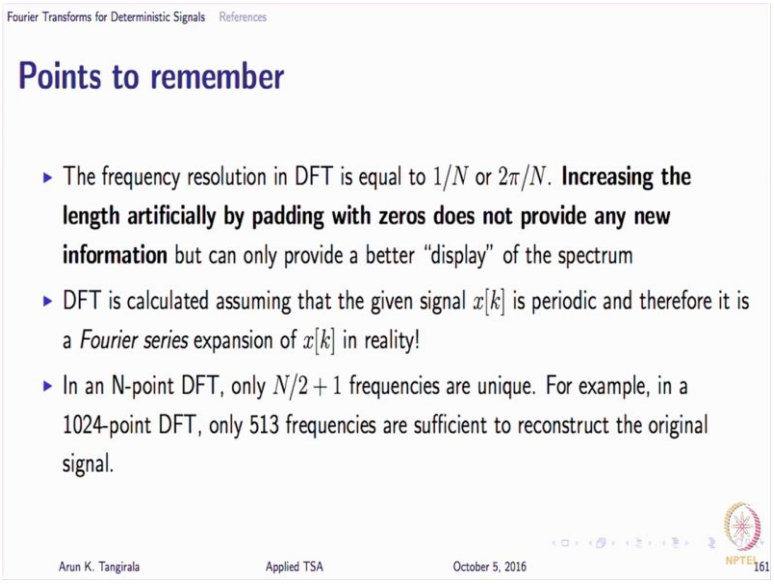
And also it is good to know the DFT inherits almost all the properties that we have discussed for DTFT, linearity, time shift and so on; particularly the convolution become a product and so on, but with the difference that the convolutions that we talk of in DTFT are replaced with what are known as circular convolution. Convolution itself is the pain, but then on top of it your circular convolution, but not much of difference in a circular convolution; essentially after the beyond the length of the signal, you assume it repeats itself that is what essentially circular convolution is; remember you will some past values in computing convolution circular; convolution assumes that the signal is periodic that is all, but we do not get into that, this is just for your information.

(Refer Slide Time: 25:30)



And very often you will see this DFT return in this way that is you introduce some W N as e to the j; 2 pi f over N; sorry 2 e to the j 2 pi by n and then you would express DFT in this way, but this is no additional information here; it is just saying that you would find the different notation.

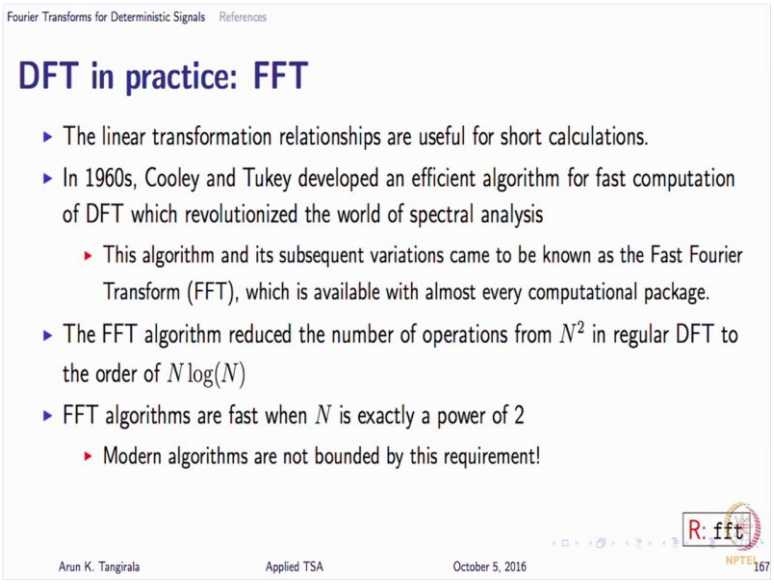
(Refer Slide Time: 25:50)



Just at quick round up of some points to remember, the frequency resolution in DFT is 1 over n which means if you have 1000 points, the resolution is 1 over 1000 so; obviously, as you have more and more observations, you have better and better resolution and we have talked about 0 padding earlier and also we have talked about the periodicity assumption and just like we have discussed before; although we are computing N coefficients here in DFT, the point to remember is only up to n by 2 plus 1; if assuming n is even n by 2 plus 1 coefficients are unique; unique in the sense that only those are sufficient to recover your signal, the rest of them are going to be conjugates not repetition they going to be conjugates ok.

Remember when we talked about discrete time Fourier series, we said the Fourier coefficients are conjugate symmetric. So, here also you have a conjugate symmetry and the point of symmetry is that N by 2 plus 1. Therefore even when you plot the periodogram or even the power spectrum, it is sufficient to plot up to n by 2 plus 1.

(Refer Slide Time: 27:37)



In fact, N by 2 itself is sufficient to plot; we will see through one illustration and here I am just telling you that whatever expressions we have seen and if you want to implement those, essentially DFT is some kind of a linear transformation because if you rearrange in a particular way then you can express DFT as a matrix multiplying your signal. All you have to do is write this DFT for every N and stack them up, then you will see that you can compute the entire sequence of coefficients by multiplying your signal with some matrix; that matrix will consist your e to the minus j; 2 pi by n minus 2 pi times 2 by N and so on and therefore, it is a linear transformation, but that perspective is useful when it comes to computation only for some short calculations; in general you need a computer and a computational efficiency algorithm and that is your FFT.

So, FFT as you know is the Fast Fourier Transform; it was conceived by Cooley and Tukey; who worked extensively on these algorithms and it reduces the number of operations from n square in regular DFT to the order of N log N and this is log base 2 and that is a considerable reduction. So, imagine that you had 1024 points. So, 1024 square is nearly a million correct; just above a million whereas, with FFT how many; what is the order of operations.

Student: (Refer Time: 29:03)

1024 times 10, so it has reduce considerably and that is a lot of time I can utilize that time for some productive things; you know what I mean right. So, originally FFT

algorithms were fast when the power of 2 but modern algorithms have overcome that limitation.