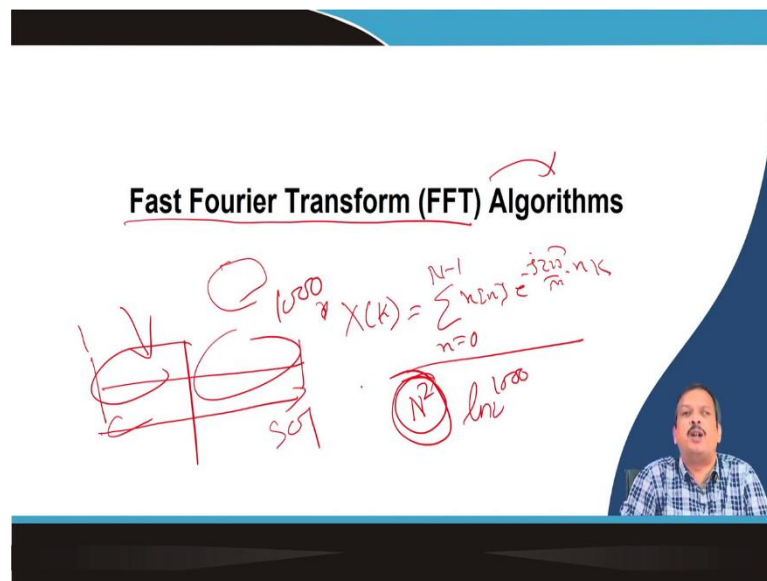


Signal Processing Techniques and Its Applications
Dr. Shyamal Kumar Das Mandal
Advanced Technology Development Centre
Indian Institute of Technology, Kharagpur

Lecture - 30
Fast Fourier Transform (FFT) Algorithms

So, let us start with another topic this week, which is called how efficiently we can calculate discrete Fourier transform; that is called Fast Fourier Transform. So, all the week, the 5th week and 6th week, I describe the discrete Fourier transform, their property, and how they can be used, and then we talk about the DCT. Now, whatever the discrete Fourier transform of BCT is, how can I compute it very fast? So, that is called a fast Fourier transform algorithm.

(Refer Slide Time: 01:01)



So, you cannot say the fast Fourier transform is a transform; it is an algorithm to compute discrete Fourier transform efficiently. So, you may ask why this is required, sir. Why do I require an efficient algorithm? So, suppose I want to implement the signal of this discrete Fourier transform in hardware. So, as you know, the discrete Fourier transform of a signal, if it is

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}nk}$$

that we know.

So, what is the complexity? I know this N^2 complex multiplication is required. So, if it is N^2 complex, multiplication is required, which means it requires 4 into N^2 real multiplication, and you know the multiplication is a costly operation in hardware. So, if I want to implement DFT in an ASIC chip, let us say. So, I have written an algorithm that requires the computation of discrete Fourier transform and inverse discrete Fourier transform.

As you know, the discrete Fourier transform and inverse discrete Fourier transform are both required because, after that work in that domain, you have to revive back the signal again in the time domain. So, in both cases, it has an N^2 complexity and N^2 complex multiplication. So, how do I reduce it? Reduction is essential for hardware implementation. So that much, how can I simplify that algorithm? There is a lot of computational simplification coming out in the picture, as you know; suppose you have a long string, suppose you have a long string, you know that.

Suppose I have data from 1 to 1000; now I say find out it is apertural there, and find out whether that 501 is there or not. So, I have to search for 501. If it is a linear search, that means the worst possible complexity is that I have to search and compare each and every data with 501, and once I find out, the worst possibility is that there will be a 1000 comparison I have to make.

But, again, you know there are a lot of efficient algorithms there, the simplest one is binary search. If I divide the search, I divide and conquer. So, if I apply the divide rule, let binary search means dividing the entire sequence into 2 sequences and the search for each of the sequences. Then you know the complexity is reduced, but the order of $\log n$, $\log 2$ and n . So, if I there is a there is a 1000 search required, now I require $\log 2$ 1000.


So, that is, obviously, a very reduce reduced form. So, how do I make this computation instead of N^2 multiplication? Reducing the number of multiplication is the basic motto of the fast Fourier transform algorithm.

(Refer Slide Time: 04:56)

Discrete Fourier Transform (DFT)

- The DFT provides uniformly spaced samples of the Discrete-Time Fourier Transform (DTFT)
- DFT
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi nk}{N}} = \sum_{n=0}^{N-1} x[n] W_N^{nk}$$
- IDFT
$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi nk}{N}} = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk}$$
- Requires N^2 complex multiplies and $N(N-1)$ complex additions
- $4N^2$ real multiplications
- $4N(N-1)$ real additions

Handwritten notes on slide:
 $W_N = e^{-j2\pi/N}$
 $k=0 \rightarrow 1$
 $N \times N$
 $W_N = e^{-j2\pi/N}$
 $e^{-j0} = \cos 0 - j \sin 0 = 1 - j0$
 $x[n] = \dots$



ECEN4002 Spring 2002
FFT Intro
R. C. Maher
3

So, you just as I said; so, if I say the DFT required N^2 complex multiplication. In this lecture, we always use W_N notation;

$$W_N = e^{-j \frac{2\pi}{N}}$$

So, instead of every time I write $e^{-j2\pi nk/N}$, I use the word W_N ok. So, that is the one thing, and the next thing is that whatever the algorithm I want to do, I have to reduce this N^2 complex multiplication N^2 into N^2 real multiplication.

Because you know that x can be $x[n]$ can be in the form of $a+jb$, and you know $e^{-j\theta}$ is $\cos\theta - j\sin\theta$. So, it is again a , let us say, a_1 minus $j b_1$ form. So, when I multiply these two things x with this thing, what is required? I require one complex multiplication equal to 4 real multiplication because a will multiply with a_1 , a will multiply with b_1 , then b will multiply with a_1 and b will multiply with b_1 .

So, I require 4 real multiplications for a single complex multiplication. And, if you see $X[k]$, k varies from 0 to N minus 1. So, I can say I require an N number of multiplication for every k . Now, on this side, I require N number of multiplication for 1 k . So, N number of k I required N cross N complex multiplication that is nothing but an N^2 multiplication. So, I have done I required $4N$ into N minus 1 real addition. Now, I want to reduce this complexity; instead of N^2 , can it be made something that is less than N^2 ? Is that an efficient algorithm?

So, depending on the algorithm, its efficiency and computational complexity will be reduced. So, that is called fast Fourier transform. Then, on what basis do we want to improve the? What is the redundancy? You know that if I want to reduce the number of multiplication, that means some of the multiplication is may not required, but unnecessarily I am doing it here. So, I have to want to reduce that redundancy. How do I reduce that redundancy, and what are those redundancy? I want to know that.

(Refer Slide Time: 07:58)

Handwritten notes on the slide:

- $W_N = e^{-j2\pi/N}$
- Symmetry property: $W_N^{k+N/2} = -W_N^k$
- Periodicity property: $W_N^{k+N} = W_N^k$
- Diagram: A sawtooth wave representing the phase of W_N , with a period of $N/2$.
- Text: Direct computation of the DFT is basically inefficient primarily because it does not exploit the symmetry and periodicity properties.
- Equation derivation: $W_N^{k+N/2} = e^{-j2\pi/N(k+N/2)} = e^{-j2\pi/N k} \cdot e^{-j2\pi/N \cdot N/2} = W_N^k \cdot e^{-j\pi} = W_N^k \cdot (-1) = -W_N^k$

So, what is there? You know, discrete Fourier transform has two properties: symmetry property and periodicity property. So, if I say $W_N e^{-j2\pi/N}$, that has two properties: symmetry property and periodicity property. Symmetry property: If I know that at N by 2 , the DF discrete Fourier transform spectrum is symmetric, So I can say $W_N k$ plus N by 2 is equal to nothing but a minus $W_N k$, you can prove it. What is W_N ? What is $W_N k$ plus N by 2 ?

It is nothing but an $e^{-j2\pi/N(k+N/2)}$. I can say it is nothing but an $e^{-(j2\pi/N)k} * e^{-(j2\pi/N)N/2}$. So, N N cancel, 2 2 cancel. So, this is nothing but a $W_N k$ again. Now, what is $e^{-j\pi}$? $e^{-j\pi}$ is nothing but a $W_N k \cos \pi - j \sin \pi$. So, this portion is 0 , and this is equal to minus 1 . So, it is nothing but a minus $W_N k$, which is a symmetric property. And what is periodicity property? You know that the DFT transform is periodic at the period of N , which is the length of the DFT.

So, W_N again W_N to the power k $W_N k$ plus N is nothing but a $W_N k$ because it is periodic because it becomes $\cos 2\pi - j \sin 2\pi$. So, \cos of 2π is equal to 1 and minus $j \sin 2\pi$ equal to 0 so, it is 1 . So, it will become $W_N k$. So, while I am computing DFT k equal to 0 to N

minus 1, I know that k equal to 0 to N minus 1, there is a symmetry, and there is a periodicity property.

So, can I exploit these two properties to reduce the computational complexity of discrete Fourier transform? So, when I directly compute DFT, we are not using this; we compute all $X[k]$, but it may not be required; for the using that redundancy information, can I do that? So, that is called a fast Fourier transform algorithm.

(Refer Slide Time: 11:18)

The slide is titled "Goal of an Efficient computation". It contains three bullet points with handwritten annotations:

- The total number of computations should be linear rather than quadratic with respect to N .
 Handwritten notes: N^2 is crossed out and circled, and N is circled. A red arrow points from the word "linear" to the circled N .
- Most of the computations can be eliminated using the symmetry and periodicity properties.
 Handwritten notes: "symmetry and periodicity" is underlined in red. A red arrow points from this underlined text to the circled N in the first bullet point.
- Adopt a divide-and-conquer approach to reduce computations.
 Handwritten notes: "divide-and-conquer" and "reduce" are underlined in red. A red arrow points from "reduce" to the circled N in the first bullet point.

A video inset in the bottom right corner shows a man speaking.

So, how do I use that? So, the goal is the goal of the algorithm, whatever the algorithm fast Fourier transform algorithm; that means the efficient computation of discrete Fourier transforms; the goal is the total number of computations should be linear rather than quadratic. If I say N^2 is quadratic, instead of N^2 , can I make it N into something a or something b which is less than N ? So, instead of square, I make it linear, a into N or b into N ; where the a and b are both are less than N .

So, instead of N^2 , I want something that is nothing but a into N . So, if a is less than capital N , then my algorithm required less number multiplication compared to the direct computation of DFT ok. So, most of the computation can be most of the computation can be eliminated using the symmetry and periodicity property. So, I use these two properties to make it an N . How do I make it? It is called adopting a divide-and-conquer approach to reduce competition. I now use the divide-and-conquer approach to reduce the computation. So, that is the goal of my fast Fourier transform algorithm.

(Refer Slide Time: 13:04)

FFT algorithm

- FFT is an algorithm to convert a time domain signal to DFT efficiently.
- FFT is not unique. Many algorithms are available.
- Each algorithm has merits and demerits.
- In each algorithm, depending on the sequence needed at the output, the input is regrouped.
- The groups are decided by the number of samples.

Diagram: A horizontal line with four vertical tick marks and a red 'X' at the far right end.

If you see that the FFT is not unique; many algorithms are available. So, FFT is not a transform fast Fourier transform algorithm. It is an algorithm to efficiently compute discrete Fourier transform. Discrete Fourier transform is a transform and fast Fourier transform algorithm to implement discrete Fourier transform in an efficient manner. Then the algorithm may not be unique, there may be many algorithms, and all algorithms may not be the same computationally, so you can see the simplification is done.

So, maybe algorithm 1 is less computationally complex compared to algorithm 2. But there may be some merits and demerits of each of the algorithms, not only reduction of the computational complexity, there may be another artifact. As that input required shuffled, the output will come with shuffled, and sometimes I require you to say some kind of arrangement of the signal, some kind of arrangement of the output. So, a lot of merits and demerits will come.

So, how will the input be arranged? How we? So, I have used the divide and conquer approach. How do I divide? Should I divide that entire sequence into 2 sequences, or should I divide it into 4 sequences? Should I divide it into an N sequence? So, how do I divide? How do I compute? So, depending on that approach, different FFT algorithms are available, and every algorithm has its own merits and demerits.

So, again, I am saying the purpose of a fast Fourier transform is to efficiently compute discrete Fourier transform. So, FFT is not a new transform; it is an algorithm that is

computationally efficient to compute DFT, utilizing the properties of DFT: one is symmetry, and the other is periodicity.

(Refer Slide Time: 15:33)

Computation of an N-point DFT using divide-and-conquer approach

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi nk}{N}} = \sum_{n=0}^{N-1} x[n] W_N^{nk}$$

Let sequence $x[n]$, has N numbers of sample
 Now $x[n]$ can be stored in either a one dimensional array indexed by n or as a two-dimensional array indexed by l and m, where $0 \leq l \leq L-1$
 $0 \leq m \leq M-1$

$N = ML$
 $N = 2000$
 $N = 50 \times 40 = 2000 = 50 \times 40$
 $L = 50$
 $M = 40$

$n = M \cdot l + m$
 $x[n] = x[l, m]$

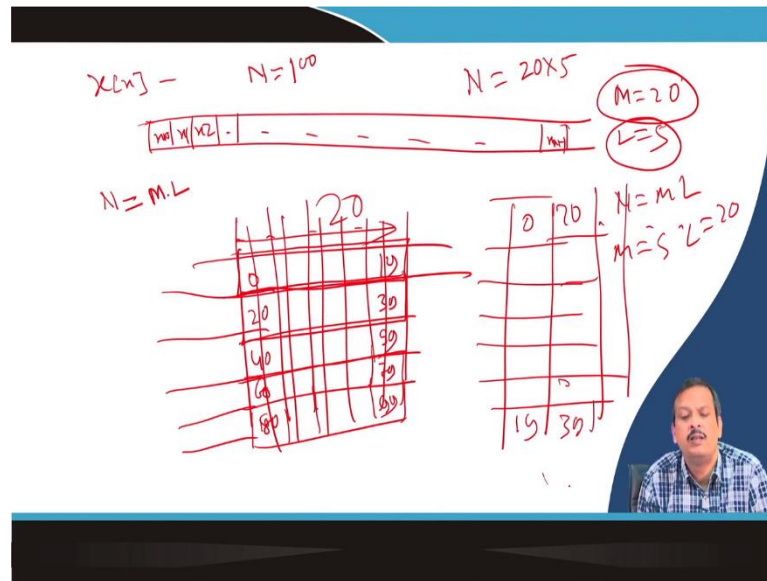
The first row consists of the first M elements of $x[n]$, the second row consists of the next M elements of $x[n]$.

Let us say divide compute N point DFT using the divide and conquer approach. So, I divide it. Let us say I have to compute N point DFT. So, this DFT length is N and N is equal to I have divided N. So, N is equal to L into M; that means, suppose I have N equal to, let us say, 200; I can say that 200 is nothing but a, let us say, 50 divided by 50 means 50 into 40.

So, I can say L is equal to 50, and M is equal to 40. So, if I say 40 N is equal to 50 into 4, it's 200. If it is 2000, then I can say 50 into 40, let us say 2000. So, 50 into 40 again, it is 2000. So, let us say the length of the DFT is not a prime number. When I say when I directly compute DFT for N point DFT, N is equal to M into L or L into M whatever where M is equal to N and L is equal to 1 or L is equal to N, M is equal to 1. So, except N is a prime number, if my length of the DFT is not a prime number, then I can use the divide and conquer approach.

So, when I go for the divide and conquer approach, the first criterion is that N must be a pro, not a prime number. N cannot be a prime number; if it is a prime number, then I cannot apply the divide-and-conquer rule. So, if it is a non-prime number or it can be expressed in a product of two numbers, then I can divide the N sequence into two parts: one is M, and the other is L.

(Refer Slide Time: 17:58)



Now, suppose I have an $x[n]$, which is N number of samples are there, N number of samples are there. How is the signal stored? So, let us say there is an array here that is store $x[0]$, this location is store $x[1]$, and this location is store $x[2]$ like that up to $x[n-1]$. $x[n-1]$ a number of samples are stored. Now, I said that N can be divided; N is a non-prime number and can be expressed as an M into L .

Now, instead of storing the signal in a single array, can I store it in a two-dimensional array, which is here? Can I store the data in a two-dimensional array? This axis is L , and this axis is M . So, how do I store it? I have an N number of samples. So, I said the first M number of the samples occupies the first row. So, 0 sample to M minus 1 number of sample occupies the first row, then M sample to $2M$ minus 1 number of sample occupies the second row and that way up to $x[n]$ sample.

So, if N is equal to M into L then I can show, I can say that there will be a L number of row required to store the N number of sample. For example, suppose I have an $x[n]$ or N is equal to, let us say, a 100 and I express. So, I can store 100; let's say N is equal to I have divided it into, let us say 20 into 5. So, M is equal to 20, and L is equal to 5. So, 0 to 19 samples will occupy the first row.

Then 20 to 39 samples occupy the second row, then the 40 to again 20 samples, 40 to 59 samples will occupy the third row, and then the 60 to 79 samples; then, I can say 80 to 100 samples, 100, not 100 samples; it starts from 0. So, it is 99 samples. So, this is 20 samples;

so, there will be 20 columns and 5 rows. So, the number of rows is defined by L , and the number of columns is defined by M , where I store the data in row-wise. Now, the same thing can be done in column-wise also; I said I would store the data column-wise. So, N is equal to ML ; now I said M is equal to 5, and L is equal to 20.

So, I can say 0 to 19 samples will be stored in the first column and then 20 to 39 samples will be stored in the second column that way. So, how many columns will be there? 5 columns will be there. So, both are possible. So, if I want to store N number of data in a two-dimensional array. So, instead of $x[n]$, I now say $x[l]$ and m , and the index is changed to $l\ m\ n$. l is the whatever, l is the number of columns, and m is the number of rows. So, m is the number of columns, l is the number of rows, whatever you can say. I can store the data row, or I can store the data column-wise; both are possible.

(Refer Slide Time: 22:52)

If mapping of $n = Ml + m$

Then the first row consists of the first M elements of $x[n]$, the second row consists of the next M elements of $x[n]$.

$x[0]$	$x[1]$	$x[2]$...	$x[M-1]$
$x[M]$	$x[M+1]$	$x[2M-1]$
2				
...				
$x[(L-1)M]$...

Handwritten notes for row-wise mapping:

- $n = m + m$
- $n = 0 + m$
- $n = m + m$
- $m + 1$
- $m \Rightarrow 0 \text{ to } M-1$
- $l \Rightarrow 0 \text{ to } L-1$
- $m + 2 \quad x[n], x(l, m)$

If mapping of $n = l + Lm$

stores the first L elements of $x(n)$ in the first column, the next L elements in the second column,

$x[0]$	$x[L]$...	$x[(M-1)L]$
$x[1]$	$x[L+1]$...	$x[(M-1)L+1]$
2			
...			
$x[L-1]$	$x[2L-1]$

Handwritten notes for column-wise mapping:

- $n = l + Lm$
- $n = l$
- $n = l + L$
- $0 - L-1$

So, let us say my mapping is n is equal to Ml plus small m . So, I am storing the data in row-wise, then n is equal to m into l plus m , m varies from, m varies from 0 to M minus 1, and a small l varies from 0 to L minus 1. You can see that when l is equal to 0, m , it varies from for every l ; so, for l equal to 0, n is equal to 0 plus m . So, from 0 to M minus 1.

Then, when l is equal to 1, then n is equal to L . I know m is equal to 1 means m into 1, so m is equal to 1, plus m . So, first, we will start from $x\ M$, then we will start from $x\ \text{capital } M + 1$, and then we will start from $\text{capital } M + 2$ up to $\text{capital } 2\ M - 1$. So, I

store the data row, and then my mapping index is n is equal to m into l plus m . So, there is an m number of columns and an l number of rows.

Now, I said no, I want to store the data column then what will be the index? n equals l plus L m just reverse, n equals small l plus capital L into m small m . So, when m for m equal to 0 n is equal to l , l varies from 0 to L minus 1 and when m equals small, m is equal to 1 n equal to small l plus capital L . So, it will start from l store 0 to L minus 1 in the first column, then L 2 l minus 1 in the second column, third column, and fourth column like that way.

So, when I store the data column-wise in a two-dimensional array, then my index is n is equal to l plus capital L into m . When I want to store the data row, then n is equal to the capital M small l plus m . Is it clear? So, I have one-dimensional data that can be mapped in a two-dimensional array. So, my new x n , x n is not $x[n]$, it is $x[l]$, m .

(Refer Slide Time: 25:58)

Similar arrangement can be used to store the computed DFT values of $X[k]$
 $x[n]$ is mapped into the rectangular array $x[l, m]$ and $X[k]$ is mapped into a corresponding rectangular array $X(p, q)$

mapping of $k = Mp + q$ mapping of $k = p + qL$

$$X[p, q] = \sum_{l=0}^{M-1} \sum_{m=0}^{L-1} x[l, m] W_N^{(l+Lm)(Mp+q)}$$

$$W_N^{(l+Lm)(Mp+q)} = W_N^{lMp} W_N^{lq} W_N^{LmMp} W_N^{Lmq}$$

$$W_N^{lMp} = W_N^{LmMp} = W_N^{Mp} = 1$$

$$W_N^{LmMp} = W_N^{Lmq} = W_N^{mq} = W_N^{p}$$

$$W_N^{Lmq} = W_N^{mq} = W_N^{mq}$$

$l = 0$ to M
 $p = 0$ to L

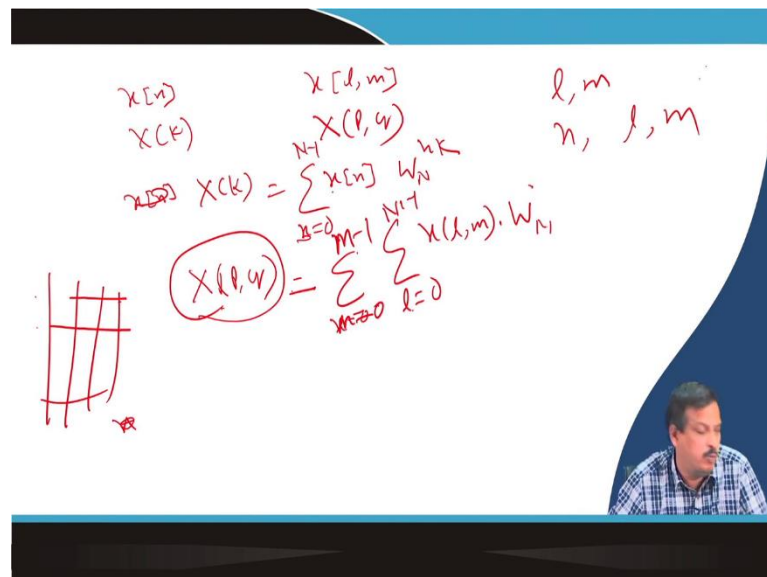
$X(k)$
 $x[n]$
 $X(l, m)$
 $X(p, q)$

Now, if my x is $x[l]$, m ; then $X[k]$ is not $X[k]$, $X[k]$ will be $X(p, q)$ ok or not. So, now, when $x[n]$ is expressed in the $x[n]$ single-dimensional array, I get $X[k]$. Now, I have an $x[l]$, m two-dimensional, single-dimensional signal to make an image two-dimensional array. Now, I should have an X with two dimensions, p and q , l and m ; both the dimensions will be there. So, the transform domain can also there will be both dimensions; it is not only X k .

It will be X_p and q again. I can say that if k is equal to $Mp + q$. So, p varies from 0 to M , and q varies from 0 to L . So, if it is $Mp + q$, that means the output is stored row-wise, and when k is equal to $p + qL$, the output is stored column-wise. Sorry, q if this will be q varies from 0 to M minus and p varies from 0 to L ; if you see $1 + Lm$, the same thing here is $1 + p + qL$ ok.

So, I can say that $X[k]$ is also stored row-wise or column-wise; both are possible. Now, if it is that things that $X[k]$ is stored in row-wise or $X[k]$ is stored in column then what is the DFT? So, let us see this is my DFT equation. So, let us say I will take another slide here.

(Refer Slide Time: 28:12)



The image shows a whiteboard with handwritten mathematical expressions and a diagram. At the top left, $x[n]$ and $X[k]$ are written. To the right, $x[l, m]$ and $X(p, q)$ are written. Below these, the DFT equation is written: $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}$. To the right of this, l, m and n, l, m are written. Below the DFT equation, the 2D DFT equation is written: $X(p, q) = \sum_{m=0}^{M-1} \sum_{l=0}^{N-1} x(l, m) W_N^{lp} W_N^{mq}$. To the left of this equation, a grid diagram is drawn with 4 rows and 4 columns. A small 'x' is written at the bottom right of the grid. In the bottom right corner of the whiteboard, a small video inset shows a man speaking.

So, I have an $x[n]$ instead of $x[n]$, I get $x[l, m]$; then I have an $X[k]$; instead of that, I get a capital $X(p, q)$. Now, what is $x[n]$? x of what is capital $X[k]$ is nothing but a discrete Fourier transform n equal to 0 to N minus 1, $x[n] W_N^{nk}$, that is my discrete Fourier transform. Now, what are $X(p, q)$? So, there are two dimensions: one is the p dimension, and the other is the q dimension. So, if you see that m small m is equal to not here small m that is p , p is equal to 0 to M minus 1, and sorry, this one is m .

So, m is ok let us come in here. So, I have a p and q in k is divided. So, $X[k]$ is store p and q array, and my $x[n]$ is store l and m array. So, n is expressed in terms of l and m . So, n can be let us say l is equal to 0 to or m is equal to 0 to n M capital M minus 1, then l is equal to 0 to N minus 1, $x[l, m]$ into W_N ; instead of n , I said how I represent n here? How

I represent n here? I can say if I say m is; so, there is an ok, there is a two for loop ok, let us start fresh again.

(Refer Slide Time: 30:16)

Handwritten notes on a whiteboard:

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk}$$

$$K = mP + q$$

$$X(l, q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l, m) W_N^{(l+Lm)(mP+q)}$$

$$n = l + Lm$$

A diagram shows a grid of points with axes labeled $m=0$ to $m-1$ and $l=0$ to $L-1$, illustrating column-wise storage.

So, what I said? I have a capital $X(p, q)$ is equal to I now capital $X(p, q)$. So, $X[k]$ is equal to n equal to 0 to N minus 1 $x[n] W_N^{nk}$ endpoint DFT. So, capital $X(p, q)$, I have said the n is divided in terms of M into L . Let us say the data is stored column-wise; data is stored in the column. So, the data will go like this. Then again, come back here and go like this: data is stored like this. So, what is the mapping function?

I know n is equal to l plus m into l plus capital L into m small m ; capital L into small m . Why? That means I am computing column-wise. So, the first loop will start from 0 to L minus 1 for every m and the first second for the loop will start from m equal to 0 to capital M minus 1. So, for every m , I am calculating L minus 1. So, I can say that inner loop inner let us say I write in here $X(p, q)$ inner sum will come from l equal to 0 to L minus 1 and the outer sum will come from m equal to 0 to M minus 1.

Because the data is stored in a column wise and the mapping function is n is equal to l plus capital L into m ok. So, now, I have an $x[l]$ m or m l whatever e to the power or $W N$. So, instead of n , I have an l plus capital L into small m . Now, if it is computed in column-wise it can be shown that when I get the output, it will be stored in row-wise. So, if it is row-wise, what is that? What is row-wise is n equal to Ml plus m ?

So, here, it is not m n ; it is the index k . So, index k has to be stored row-wise. So, how is the index k stored row-wise? That is nothing but a k equal to; so, it is this mapping is row-wise, this mapping is column-wise. So, k is equal to M p plus q .

(Refer Slide Time: 33:25)

$$X[p, q] = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x[l, m] W_L^{lp} W_M^{mq} W_N^{lq}$$

$$= \sum_{l=0}^{L-1} W_N^{lq} \left(\sum_{m=0}^{M-1} x[l, m] W_M^{mq} \right) W_L^{lp}$$

Step-1: Compute M point DFTs for each row
 $F(l, q) = \sum_{m=0}^{M-1} x[l, m] W_M^{mq}$ for $l = 0$ to $L-1$
 Step-1: Number of complex multiplications: M^2L
 Number of complex Addition: $LM(M-1)$

Step-2: Compute $G(l, q)$ defined as
 $G(l, q) = W_N^{lq} F(l, q)$ for $l = 0$ to $L-1$ and $q = 0$ to $M-1$
 Step-2: Number of complex multiplications: ML

Step-3: Compute L -point DFTs
 $X[p, q] = \sum_{l=0}^{L-1} G(l, q) W_L^{lp}$
 Step-3: Number of complex multiplications: L^2M
 Number of complex Addition: $ML(L-1)$

So, I can say here I can write k as equal to m into p plus q . Do you understand or not? I said I store the data column-wise there is a m number of columns. So, the mapping function is l plus a capital L into m ; now I said the output, the computed output $X(p, q)$, is stored in row-wise. So, I get the k index in terms of m p plus q , both l m p q , l varies from 0 to capital L minus 1, p varies from 0 to capital L minus 1, and q varies from 0 to M minus 1, is it clear? So now, I get this equation, ok. So, m equals 0 to M minus 1, l equals 0 to capital L minus 1, this one.

Now, if I see so, I can say I have a 4 WN product, WN l into M p . So, W small l M p , this one is 1, then small l and q , small l and q . So, I get small l and q . So, this is this WN l multiply with M p this one, then l multiply with q this one, then L m multiply with M p . So, L m M p and then L m will multiply with q so, L m q . So, those are 4 WN I will get, now say WN L M p , W L L m p . So, WN L m small m capital M p is nothing, but a L M m p .

So, what is L m ? L m is nothing but an N , so WN is okay. So, WN N m p is nothing but a N p m p . So, it is small m and p , small m and p both are integers. So, WN to the power any integer is nothing but a 1, WN N m p which is nothing but a $e^{-j2\pi/N}$ into N m p ; so, N N

cancel. So, it is nothing but a minus $j 2\pi m p$, m and p are integers. So, if it is an integer, the product is also an integer.

So, it is nothing but a $\cos 2\pi$ minus $j \sin 2\pi$. So, minus $j \sin 2\pi$ is 0, n into or p into minus $j \sin 2\pi$. So, nothing but a 0 and this one is 1. So, this is nothing but a 1, then what is W_N l M l capital M p ; W_N l capital M p ? So, I can express in terms of W l p N by M because $e^{-j2\pi/N}$ into m l p . I can say it is nothing but an m it can be written in here also. So, if it is $e^{j2\pi/N}$, then I said W_N .

Now, if it is instead of N , it is N by M . So, instead of N , I can write N by M . So, that is why I write N by M . So, what is N by M ? N is nothing but an M L . So, it is nothing but a L . So, that is why W L to the power l p , similarly W_N L L m q same thing vice versa, only W L q N L q will remain same. So now, we simplify this one and let us see what we will get. So, in the next lecture, I will show you how it will be simplified.

Thank you.