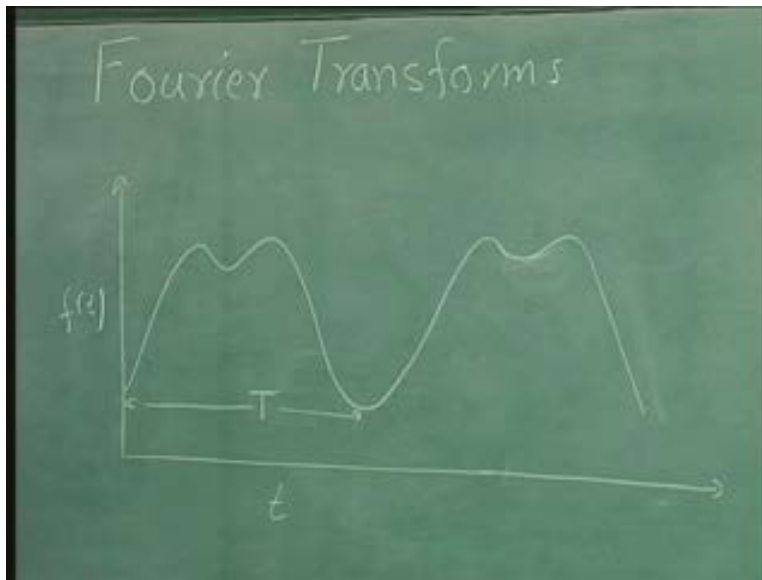


Numerical Methods and Programming
P. B. Sunil Kumar
Department of Physics
Indian Institute of Technology, Madras
Lecture - 37
Fourier Transforms

Today we will move on to a newer topic that is Fourier computing, Fourier transforms using numerical methods. So we are all familiar with familiar with Fourier series that is when you have a function which is periodic in time, say this such a function which is periodic in time or any variable t and it is periodic in the sense that.

(Refer Slide time: 01:41)

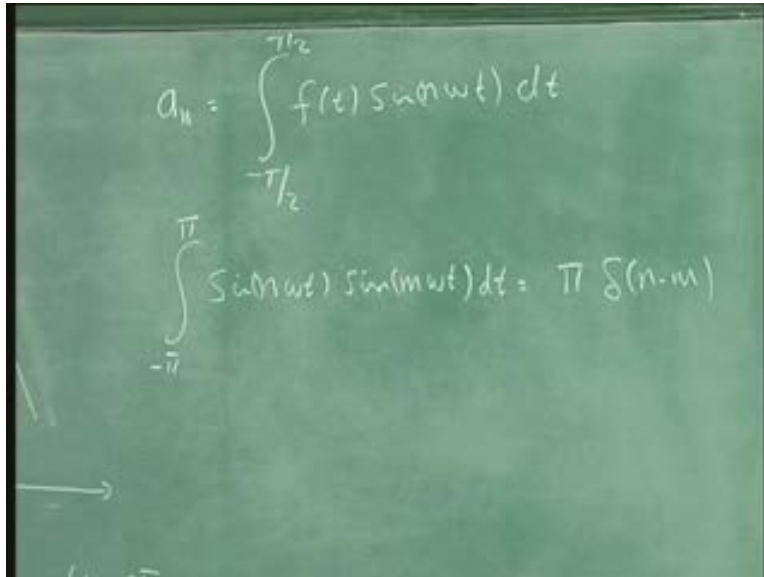


Okay now it has a repeat period of given by t here and let us say this continues and then we know that such a function can be written as a sum of sine functions and sine and cos functions that is $a_n \sin n\omega t + b_n \cos n\omega t$.

So we can write, we know this this is the Fourier series when the function is periodic and where ω here would be $2\pi/t$. So we have we can write this in this form and then we know that to obtain the coefficients a_n and b_n , we would integrate this function f of t . So we would say that to get a_n we would say that we will integrate this function between the limits $-\pi$ to π of $f(t) \sin n\omega t dt$.

So that is that is what we would do to obtain this coefficients because we use the property of the sine functions that $\int_{-\pi}^{\pi} \sin n\omega t \sin m\omega t dt$ is equal to $\pi \delta_{nm}$ that is what we are familiar with, so this means that this is non-zero only when n equal to m and when it is n equal to m , the values is π .

(Refer Slide time: 04:21)


$$a_n = \int_{-T/2}^{T/2} f(t) \sin(n\omega t) dt$$
$$\int_{-\pi}^{\pi} \sin(n\omega t) \sin(m\omega t) dt = \pi \delta(n-m)$$

So we can use this property and obtain these coefficients as a_n as $f(t) \sin n \omega t dt$ and b_n as $f(t) \cos n \omega t dt$, this is something which we are familiar with in the Fourier series and we always that the function has to be periodic for you to obtain, for that you can write it in this form. So Fourier transforms is the extension of this into even non-periodic functions.

So that is what we are going to look at here. So remember that what we did here was to write the function as a sum of sine and cosine functions with some coefficients a_n and b_n and then we obtain the coefficients a_n and b_n by integrating multiplying this function with sine and cos respectively and integrating from minus t by 2 to plus t by 2.

So it is a 1 over t of minus t by 2 to plus t by 2 is what we will get like this. Okay so that is something which we are all familiar with, so now let us say we have instead of a periodic function like this let us say we have a function of this form. Let us say, we have a function which is something like this. So we have a function t and f of t .

So here, we have a finite set of coefficients we will get in your periodic function, we have a finite set of coefficients, you will find this a_n 's and b_n which would make up this periodic function which repeats itself. On the other hand, if you have a blip like this and then you will have, if you want to write this in terms of sine and cosine function there will be all, there will be infinite number of harmonics in this kind of a function.

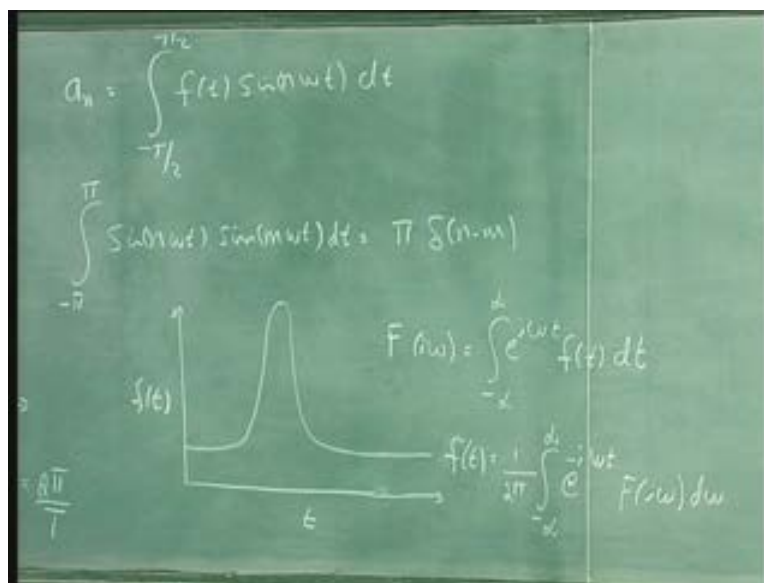
So the extension of the Fourier series to functions of this form, the extension of the Fourier series to functions of this form would be to write this as $f(\omega)$ as integral minus infinity to plus infinity, $e^{i\omega t}$, $f(t) dt$. So that would be the form.

So actually you should write $i\omega$ here f of $i\omega$ as so normally written with a capital F . So f of $i\omega$ as minus infinity to plus infinity of e to the power of $i\omega t$ f of t dt and so this is this function are called the Fourier transform of f of t for some variable t . So for some variable t , I can write an integral of this form e to the power of $i\omega t$ f of t dt and then I would get f of ω . So that is the extension of the Fourier series which we know of for non-periodic functions.

So for non-periodic function instead of writing a Fourier series, we would write a Fourier transform and the Fourier transform is given by this and the inverse Fourier transform would be then, we can also write f of t in terms of f of ω and then I would write 1 over 2π integral minus infinity to plus infinity, now e to the power of minus $i\omega t$ f of $i\omega$.

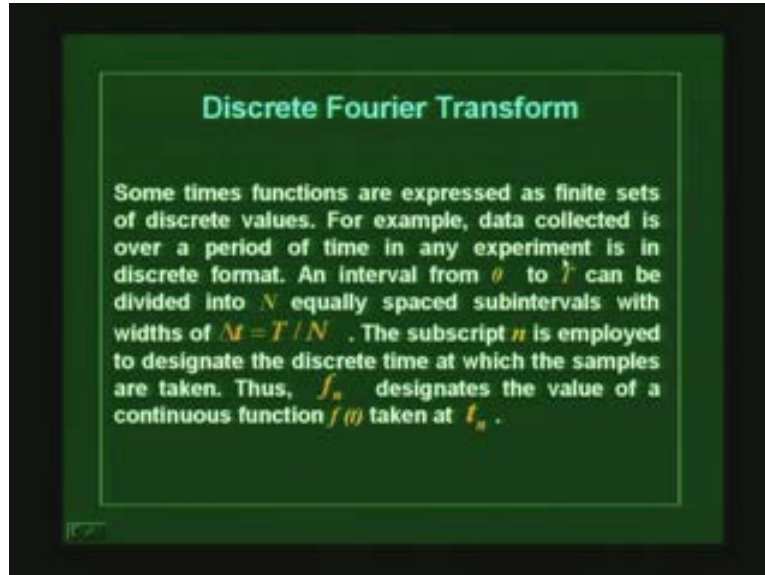
So now you write f of $i\omega$ $d\omega$ that would be the inverse Fourier transform. So now we have a definition of a Fourier transform and an inverse Fourier transform. So what we would be concerned about is how to compute such quantities, when we have a discrete set of data points, when we have the function itself sampled at a discrete set of points in t and then how do we compute the transform of such functions and that is what we would be concerned about here, that is to say that we have values of the function given at let us say some discrete interval t by n , let T is the total interval we have.

(Refer Slide time: 09:01)



So we have a finite set of discrete values collected, let us say in an interval from 0 to capital T , that is the total interval which we have and then let us say it is in n equally spaced sub intervals, we have this data collected that is equal space Δt is T by n and then how do we compute the Fourier transform of this. So that is we will such computation will be called a discrete Fourier transform as opposed to the integral transform, we will use a discrete Fourier transform.

(Refer Slide time: 09:32)



So as opposed to an integral to obtain the Fourier transform, we will here do a discrete sum and that is what we will be looking at, so to fix a notation the data collected let us say f of t is our sample the data collected in this interval we will write as f subscript n . So that is the data f at time t , f as a function of t collected at the interval n th interval will be written as f_n and that is what we would be using it. So remember these things, so what we have to do is to do a Fourier analysis of a non-periodic function of this form. So that is what we want to do.

So and then we would write that Fourier transform and the inverse transform in this fashion this $1/2\pi$ could be in either of them not in both you could put either here or here. So this is part of the differential, so this 2π which is normalizing factor here. So that is what we are trying to compute. So now let us say we do not have the data in a continuous form like this but we have it collected at equal intervals in this variable t , let us say in the interval from 0 to t and then we want to do the transform of that and compute the transform of that, that is what we will be now looking at.

So in the discrete Fourier transform when we compute we said that we would divide the interval into n equally spaced sub intervals and denote the function by f_n . So graphically it does that you have a function which goes from 0 to t and then you sample this at equal intervals here and then each of these values in each of these values we denote by f_n .

So that is the idea and then this transform which is our continuous transform which is our continuous transform that is minus infinity to plus infinity $f(t) e^{i\omega t} dt$ which is now sampled at n equal intervals. So we define this notation ω_k here which I wrote as $2\pi/n$ times some integer k . So this transform is then written as in this form. So we would write the discrete transform from the continuous 1 that is f of k . So what we have written before as $f(i\omega_k) = \int_{-\infty}^{+\infty} f(t) e^{i\omega_k t} dt$ will now be written as a sum of all the

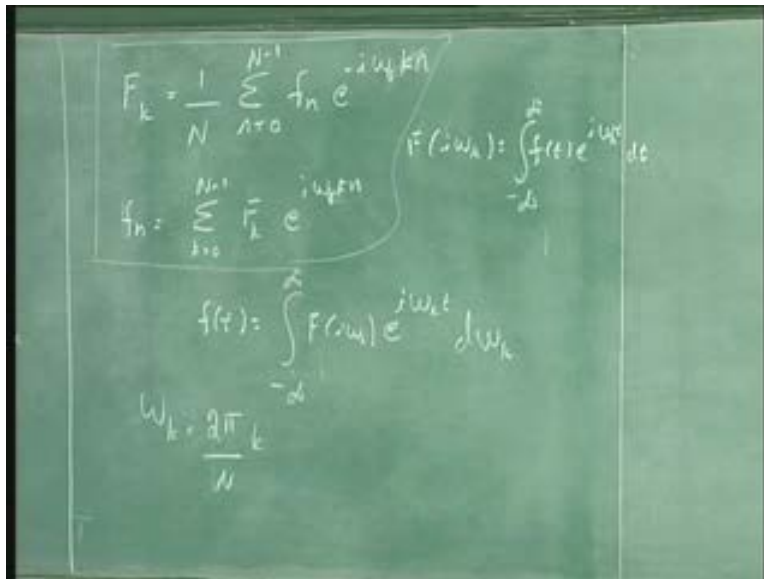
function values f_n remember n is the function value n sampled at the n th interval multiplied by e to the power of minus i omega k times n . So where k_n here are integers and the sum goes from n equal to 0 to n minus 1 there are n points here.

So n goes from 0 to n minus 1, so this $1/n$ is again is a normalizing factor. So which can be put either in this or in the inverse Fourier transform? So this continuous transform is then now turned into a discrete sum and the inverse transform which is f of i omega k , e to the power of i omega k t d omega k for every t is now written as f_n is k going from 0 to n minus 1, $f_k e$ to the power of i omega $0k_n$.

So remember k and n are integers here. So now this is equivalent of saying f of t is equal to integral minus infinity to plus infinity f of i omega k , e to the power of i omega k t d omega k . So remember omega k here is 2π by n times k , k is an integer. So now this is a 2 sum expressions which we are going to use to compute the discrete Fourier transform.

So that is the Fourier transform this is the function evaluated at a discrete set of points and then we would get a f of k 's and using the f of k 's, we can compute the function at that discrete set of points using the inverse Fourier transform.

(Refer Slide time: 15:14)



So this is the 2 transforms and now you can see that if I want to actually compute this transform using this n functions, n intervals and then to compute this quantity for every k we have to do n into n operations. So let us look at that, so this has to be computed. So we have n values of f capital n values of f from 0 to f n minus 1. So this sum now that we know what the discrete transform is now this sum used for every k going from 0 to n minus 1.

So there are n such cases for each of this case we have to do n sums. So n products and n sums, so the number of operations goes as n square. So as you increase the k number of k values or the number of n values the computation involves also blows up as n square and similarly, for the inverse Fourier transform so that is the problem with computing the discrete Fourier transform. So you can compute this, so 1 way of computing this is the following that we use the property that this can be written as \cos that is I can write f_k as 1 over n so times $\sum_{n=0}^{k-1} f_n \cos(\omega_0 k_n + i \sin(\omega_0 k_n))$.

So that is how we compute this, so we have to evaluate these functions at all the for each k , for each k we have to evaluate these functions n times and then do this product separately. You write it in this fashion so that even in languages like c which cannot handle complex numbers, this is a complex number. So languages like c which cannot handle complex numbers can do this transform can compute this transform by using the doing 2, it in 2 different parts.

So we will have a f of k is the real part. So we will have, we will compute f_k real, so real part of f of k let us call it as f_k real. So that will be 1 over n $\sum_{n=0}^{k-1} f_n \cos(\omega_0 k_n)$ and then we compute f_k imaginary part as 1 over n $\sum_{n=0}^{k-1} f_n \sin(\omega_0 k_n)$. So we can compute these 2 separately and thus I do not have to handle a complex number but the problem is really that the numbers of operations are extremely high.

(Refer Slide time: 18:53)

The image shows a chalkboard with the following equations written on it:

$$F_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n e^{-i\omega_0 kn} \quad k=0, N-1$$

$$f_n = \sum_{k=0}^{N-1} F_k e^{i\omega_0 kn}$$

$$F_k = \frac{1}{N} \sum_{n=0}^{k-1} f_n [\cos(\omega_0 kn) + i \sin(\omega_0 kn)]$$

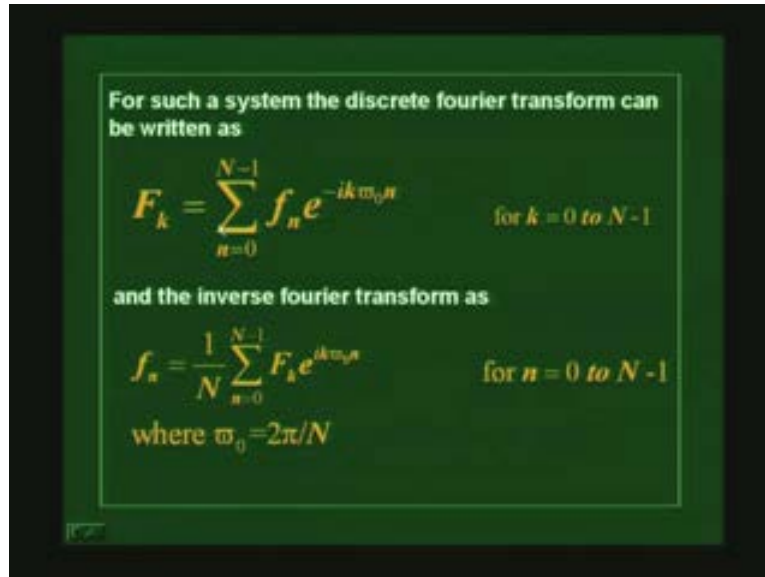
$$\text{Real } F_k = \frac{1}{N} \sum_{n=0}^{k-1} f_n \cos(\omega_0 kn)$$

$$\text{Im } F_k = \frac{1}{N} \sum_{n=0}^{k-1} f_n \sin(\omega_0 kn)$$

So that is the problem with this discrete Fourier transform computation of discrete Fourier transform. Okay. so there are better methods of doing this and 1 of them is called the fast Fourier transform and that is what we would be actually interested in. So let me just summarize this, so we would divide the interval t into n equal intervals and then denote the functions by f_n and then you would compute the Fourier transform as using

this form, that is e to the power of $ik\omega_0 n$, f_n as the Fourier transform and the inverse Fourier transform as 1 over n , $\sum_{n=0}^N$ equal to here, it should be k equal to 0 to n minus 1 e to the power of this sum is over k and this sum is over n .

(Refer Slide time: 19:35)



For such a system the discrete fourier transform can be written as

$$F_k = \sum_{n=0}^{N-1} f_n e^{-ik\omega_0 n} \quad \text{for } k = 0 \text{ to } N-1$$

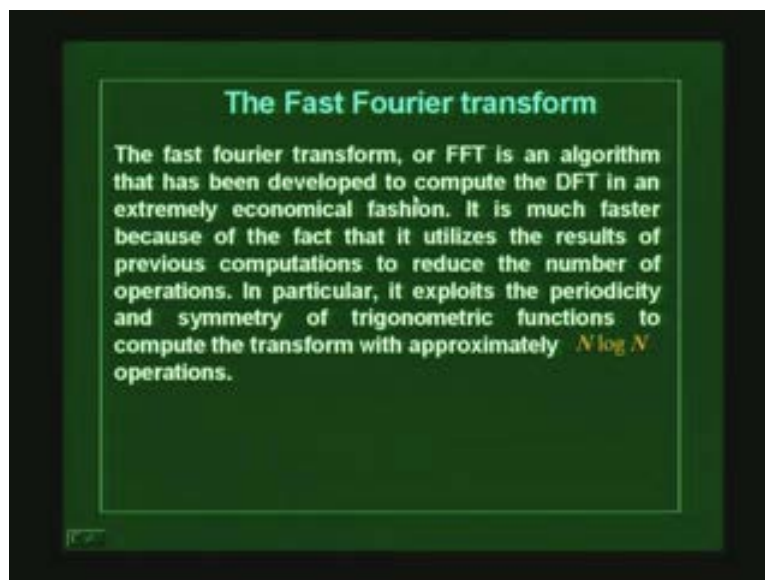
and the inverse fourier transform as

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} F_k e^{ik\omega_0 n} \quad \text{for } n = 0 \text{ to } N-1$$

where $\omega_0 = 2\pi/N$

So our ω_0 here is $2\pi/n$. So as I said the problem is the number of operations here and the number of operations can be reduced while going to what is called a fast Fourier transform.

(Refer Slide time: 20:51)

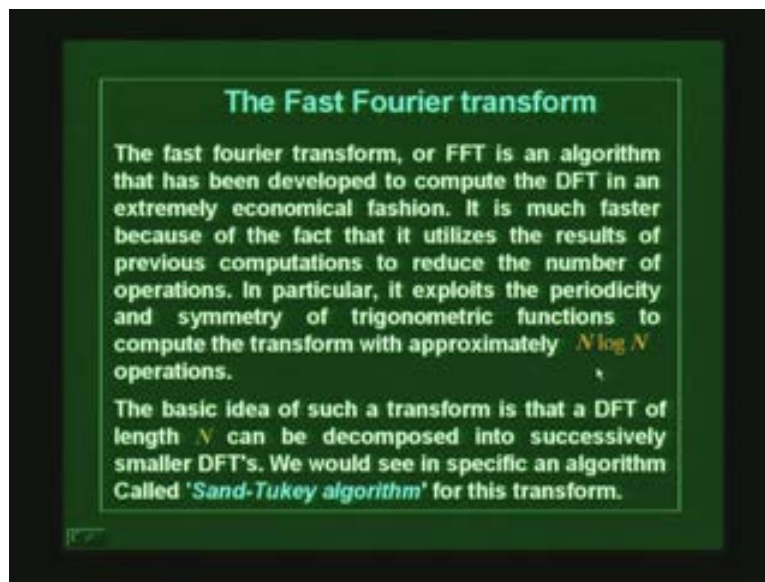


The Fast Fourier transform

The fast fourier transform, or FFT is an algorithm that has been developed to compute the DFT in an extremely economical fashion. It is much faster because of the fact that it utilizes the results of previous computations to reduce the number of operations. In particular, it exploits the periodicity and symmetry of trigonometric functions to compute the transform with approximately $N \log N$ operations.

So this dft which is discrete Fourier transform requires n^2 complex operations the complex operations can be reduced to the real operations by writing this as $e^{i k \omega_0 n}$ to the power $\cos k \omega_0 n + i \sin k \omega_0 n$ and then computing the real and imaginary part separately that means we actually have 2 times n^2 real operations in this computation. So 1 way to go around this is to use what is called the fast Fourier transform, this is a very clever technique which allows us to compute this discrete Fourier transform in $n \log n$ operations. So we use the symmetry of the trigonometric functions to get to compute this discrete Fourier transform in $n \log n$ operations.

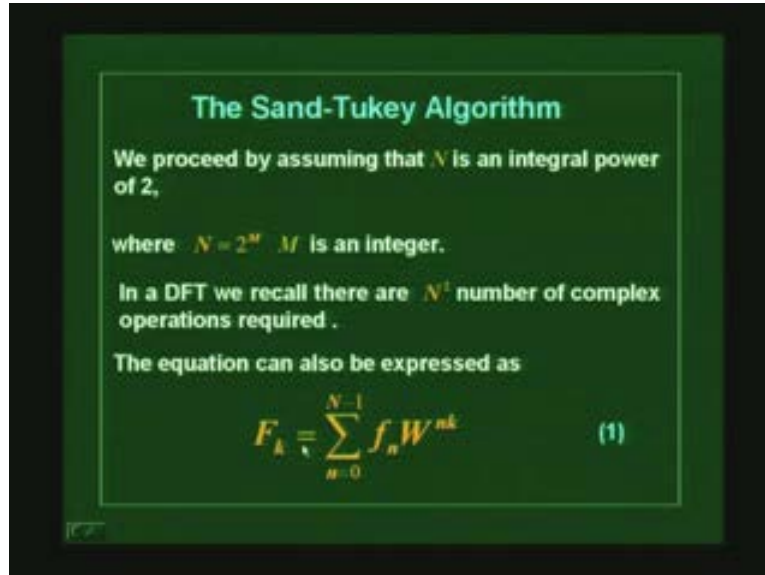
(Refer Slide time: 21:16)



So that is what we would be now looking at, so the basic idea is here that you have this n number, n functions, we have n functions now this n is now we have n such intervals. So let us write that here. So we will, let us say we have functions like $f_0, f_1, f_2, f_3, f_4, f_5, f_6$ and f_7 , okay, let us say we have eight such function values and then the idea is that we would divide this into 0 to 31 and 4 to 71 and compute the discrete Fourier transform each of them and then we can show the whole idea behind fast Fourier transform is that we can actually divide this into smaller and smaller sub sets and then compute the Fourier transform as a cascade that is what we would see.

So that is the whole idea because what is called the Sand-Tukey algorithm. So we will see that now in the, so the algorithm is the following that we would write the Fourier transform f_k of the function f_n as a function of this form that is $\sum_{n=0}^{n-1} f_n w^{nk}$. Okay so now that is same as what we have written here.

(Refer Slide time: 22:51)



So this 1 now the w_n 's now w^{nk} is nothing but that. So we are going to write this function as write this discrete Fourier transform we are going to write as let us forget 1 over n for the time being, let us say this is our discrete Fourier transform, we will put the 1 over n in the inverse Fourier transform and then we write this as $\sum_{n=0}^{N-1} f_n w^{nk}$ we will say that it is f_n we say some function called w to the power kn , k and n are integers.

So now this w is nothing but e to the power of $-i \omega n$ and remember ω is 2π by N that is what we have here. So now this is the form which we are going to write remember we will always use this w , so this w is e to the power of $-i \omega n$ and I would write this as this transform as $\sum_{n=0}^{N-1} f_n w^{kn}$.

So now as I said, we will divide this interval into 2 so this interval $n-1$, this n functions will now divide it to n by 2 and n by 2 and write the sum now $\sum_{n=0}^{n/2-1} f_n w^{kn}$ plus $\sum_{n=n/2}^{n-1} f_n w^{kn}$. So that is what we want to we would write so let us use slightly different it does not matter.

So we have $\sum_{n=0}^{n/2-1} f_n w^{kn}$. So this n is now split into, so the n functions are now n function values which we have is split into now 2 intervals 1 goes from 0 to $n/2-1$ and other goes from $n/2$ to $n-1$. So these 2 intervals we will write, so now we will make a small change of variable. So that the sum is actually going from 0 to $n/2-1$.

So we want to write that so this the way we can write is to say that this is $\sum_{n=0}^{n/2-1} f_n w^{kn}$, w to the power kn and then we say that here in the \sum we say this is f_n , we call some variable n plus $n/2$ and w power km now the m now goes from 0 to $n/2-1$. So m goes from 0 to $n/2-1$.

So you see when m is 0 this is n by 2 and that is what, when n is n by 2 that is what we have so these 2 are the same I just made a small change of variable went from n to m and then by adding, so n is m plus n by 2. So I made n is m plus n by 2, I made this small change of variable to write this as both the sum as 0 to n by 2 minus 1. So I can, here also I had to write that n by 2.

(Refer Slide time: 27:01)

The image shows a chalkboard with the following handwritten equations:

$$F_k = \sum_{n=0}^{N-1} f_n e^{-j\omega_0 kn} \quad k=0, N-1$$

$$= \sum_{n=0}^{N-1} f_n(W)^{kn} \quad W = e^{-j\omega_0}$$

$$= \sum_{n=0}^{N/2-1} f_n W^{kn} + \sum_{n=N/2}^{N-1} f_n W^{kn}$$

$$= \sum_{n=0}^{N/2-1} f_n W^{kn} + \sum_{m=0}^{N/2-1} f_{m+N/2} W^{k(m+N/2)}$$

$n = m + \frac{N}{2}$

Okay so I have this thing here. So now you see these are dummy variables m and n are dummy variables, so I can just write them as n itself so I will write the Fourier transform my discrete Fourier transform as n going to 0 to n by 2 minus 1, we will say $f_n w^{kn}$ plus $f_n w^{kn}$ plus capital N by 2. So now you see my sum is now only over n by 2 function values and 1 this $f_n w$, so what I have done is this is a dummy variable. So I just took the, this line is exactly the same as this line only thing is I have put the sum as 1 here because m and n are dummy variables.

So I can actually pull out f_n from here, we can see so you can write I can pull out w^{kn} from here and then write this sum as f_n plus f_n plus n by 2 w to the power n by 2 multiplied by wk. So let us write it that fashion. So going back from here, we would write so what we have written here is this that the Fourier transform we wrote as f_k as sigma n equal to 0 to capital n minus 1 and we said I can write it as $f_n w^{kn}$ plus sigma m going from n by 2 to capital n minus 1 this is n by 2 minus 1 this is n minus 1.

We said I can write it as $f_n w^{kn}$ I just split that into 2 this Fourier sum and then we made a change of variable and then we got this now as sigma n going from 0 to n by 2 minus 1 I can write it as f_n plus w to the power k^n by 2 into f^n plus capital n by 2 into w^{kn} . So what we have done is to make a change of variable here and then I wrote it as m going from 0 to n by 2 minus 1. So by making a change of variable, so this will become f_n plus m by 2 and w to the power k_n plus m by 2 and then we split that. So let us go through over this again once again.

So we have this written here and then I made a change of variable here and then make this going from, instead of going from n by 2 to n minus 1, I have it going from I am going from 0 to n by 2 minus 1 and then I change this to m plus n by 2 and here also m plus n by 2 basically I made this change of variable here and then I said that this n and m are dummy variables.

So I can pull that out and write it as f_n both as sum over n there is nothing which prevents me from doing that because these are dummy variables and then I can pull out this ω^{kn} from this ω^{kn} I can pull out, it is common in both the terms and write it as f_n plus ω^{kn} by 2 f_n plus n by 2 ω^{kn} . So now remember what is w , so w is nothing but e to the power of minus i times 2π by n that was our w , that is i omega naught was omega naught was 2π by n .

So it is i times minus i times 2π by n , so what is w to the power kn , so w to the power kn by 2 so w to the power kn by 2 is nothing but e to the power of minus i , 2π by n into k times n by 2. So that is we know is nothing but e to the power of minus i phi to the power k .

(Refer Slide time: 32:00)

$$F_k = \sum_{n=0}^{N-1} f_n w^{kn} = \sum_{n=0}^{N/2-1} f_n w^{kn} + \sum_{n=N/2}^{N-1} f_n w^{kn}$$

$$= \sum_{n=0}^{N/2-1} (f_n + w^{kN/2} f_{n+N/2}) w^{kn}$$

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

$$w^{kN/2} = e^{-j\frac{2\pi}{N} \frac{kN}{2}} = (e^{-j\pi})^k$$

So realizing this is the key to fast Fourier transform. So from up to this it is a discrete Fourier transform all we have done was to split this into 2 halves and we are still doing the same number of sums. So here again we are doing the same number of sums but now we realize this w to the power of kn by 2 is actually e to the power of minus i phi to the power k and we know that e to the power of i phi is just minus 1. So this is just minus 1, so what we have here is actually minus 1 to the power k , now that is the trick.

So we have minus 1 to the power k , so all we have to do is to write this as now f_n plus minus 1 to the power k f_n plus n by 2 now this is obvious that this would mean that for all

odd integers this becomes minus and for all even case it becomes plus. So if you are computing f_k for even case then this is f_n plus f_n by 2 and if it is for odd case then we are computing it as f_n minus f_n plus n by 2.

So that is the sum, so we will write this sum now remember this. So we can write this as minus 1 to the power, so that is nothing but minus 1 to the power k. So we have now this f_k as so we say f_k as sigma n going from 0 to n by 2 minus 1 so we say f_n plus f_n plus n by 2 w^{kn} for k odd, for odd k and equal to sigma n going from 0 to n by 2 minus 1, f_n minus f_n plus n by 2 w^{kn} for even k minus for even k and plus for, minus for odd k and plus for even k. So we have this sum, okay now we have 2 different sums, we have sum of the functions here for all the even k values and difference between the functions in the 2 halves for the odd k values.

(Refer Slide time: 34:53)

The image shows a chalkboard with the following handwritten equations:

$$F_k = \sum_{n=0}^{N-1} f_n W^{kn} - \sum_{n=N/2}^{N-1} f_n W^{kn}$$

$$= \sum_{n=0}^{N/2-1} (f_n + W^{kn/2} f_{n+N/2}) W^{kn}$$

There is a note $(-1)^k$ with an arrow pointing to the $W^{kn/2}$ term in the second equation.

$$F_k = \sum_{n=0}^{N/2-1} (f_n - f_{n+N/2}) W^{kn} \quad \text{odd } k$$

$$= \sum_{n=0}^{N/2-1} (f_n + f_{n+N/2}) W^{kn} \quad \text{even } k.$$

So remember if you have the function tabulated in this form. Okay so now that is taking the difference for, difference between 2 functions for all the odd k values and taking the sum between the functions, so this and that. So we divide this into 2 halves here, so what we are going to do is we are taking the difference between these 2 and the sum between these 2. So this is 0 and n by 2 minus 1, so that is what we are going to do.

So we are going to take the sum and the difference between these half functions and this half functions and write them that is the key to this Sand-Tukey algorithm so we, so let me summarize this again here. So we write this function f_k as $f_n w^{nk}$ and then we would write that as 2 different sums that goes from 0 to n by 2 minus 1 and n by 2 to n minus 1 2 functions f_n e to the power of minus i this is omega naught which I had written as kn and e to the power of minus i 2 phi by n kn.

So now these can be written by a small change of variable going from this again, the second part as we said the second part here we make a small change of variable here and

make this sum go from same 0 to n by 2 minus 1 that I can do by defining a variable m which is n minus n by 2. So that is what I have done here. So we have that here.

(Refer Slide time: 36:45)

where $k=0,1,2,\dots,N-1$. A new variable, $m = n - N/2$ is introduced so that the range of the second summation is consistent with the first,

$$F_k = \sum_{n=0}^{(N/2)-1} f_n e^{-i(2\pi/N)kn} + \sum_{m=0}^{(N/2)-1} f_{m+N/2} e^{-i(2\pi/N)k(m+N/2)}$$

or

$$F_k = \sum_{n=0}^{(N/2)-1} (f_n + e^{-i\pi k} f_{n+N/2}) e^{-i2\pi kn/N}$$

So we have m going from 0 to n by 2 minus 1 f_m plus n by 2 e to the power of i now there is this m has to be replaced by m plus n by 2. So this part, so second part of the sum we modified a little bit and to this form. So the sum is going from 0 to n by minus 1 and going from 0 to n by 2 minus 1 here also so I could pull out this sum here and also pull out this e to the power of 2 phi, minus i 2 phi by n^{kn} which is common for both of this because m is just a dummy variable.

So I can pull out these 2 and then write it as $f_n e$ to the power of minus i phi k, now phi k because now when I multiply it here you this 2 phi by n and this n by 2 cancel, so we have i phi k, so f_n plus e to the power of minus i phi k $f_{n+N/2}$ e to the power of minus 2 pi kn by n. So that is our Fourier transform and then we said that now this is nothing but minus 1 to the power k, so that things become simple. So I can write now this as 2 different parts.

So I can write this as 2 different parts 1 is for the even values that is f_{2k} 's, so now I can write for the even functions I write it as sum because minus 1 to the power k is 1 for even k values. So that is f_{2k} if I can write it as for k going from 0 to n by 2, so I can write it as f_{2k} and then for all the even functions I have it as sum of these 2 and for the odd functions that is $2k + 1$ so I split this sum k's into $2k$ and $2k + 1$.

(Refer Slide time: 37:58)

Noting that $e^{-n\pi} = (-1)^n$, we get for even values

$$F_{2k} = \sum_{n=0}^{(N/2)-1} (f_n + f_{n+N/2}) e^{-i2\pi(2k)n/N}$$

$$= \sum_{n=0}^{(N/2)-1} (f_n + f_{n+N/2}) e^{-i2\pi kn/(N/2)}$$

(Refer Slide time: 38:40)

and for odd values

$$F_{2k+1} = \sum_{n=0}^{(N/2)-1} (f_n - f_{n+N/2}) e^{-i2\pi(2k+1)n/N}$$

$$= \sum_{n=0}^{(N/2)-1} (f_n - f_{n+N/2}) e^{-i2\pi n/N} e^{-i2\pi kn/(N/2)}$$

So then I can write that as the difference between these 2 functions. So I have 2 things quantities here. So I have a f_{2k} and a plus and a minus function there are $2k+1$. So the idea here is that I am writing this as for odd k values this 1 and even k values this. So to make sure that these odds I would write this as $2k$, I can change that $2k$ and say that this is for all k going from 0 to n by 2, n by 2 I can do this for n minus 1 by 2 I can do this for all k values for $2k$.

So I can do this up to that n by 2 minus 1 n by 2 minus 2 here it will be, so I will have that many values in the even thing and similarly, I can write this as f_{2k+1} again $2k$

and $2k$ plus 1, this is for the odd function and this is for even values I can write this as 2 different sums this is to make sure that this is $2k$.

(Refer Slide time: 40:30)

The chalkboard shows the following derivation:

$$F_k = \sum_{n=0}^{N/2-1} f_n W^{kn} + \sum_{n=N/2}^{N-1} f_n W^{kn}$$

$$= \sum_{n=0}^{N/2-1} \left(f_n + W^{kN/2} f_{n+N/2} \right) W^{kn}$$

where $W^{kN/2} = (-1)^k$.

$$F_{2k} = \sum_{n=0}^{N/2-1} (f_n - f_{n+N/2}) W^{(2k+1)n} \quad k = 0, N/2 - 1$$

$$F_{2k} = \sum_{n=0}^{N/2-1} (f_n + f_{n+N/2}) W^{2kn} \quad \text{Even } k.$$

(Refer Slide time: 41:02)

Now we have for the odd and even terms

$$F_{2k} = \sum_{n=0}^{(N/2)-1} (f_n + f_{n+N/2}) W^{2kn}$$

$$F_{2k+1} = \sum_{n=0}^{(N/2)-1} (f_n - f_{n+N/2}) W^n W^{2kn}$$

So I should write here also as $2k$ and I should write here as $2k$ plus n that is n , so I have 2 sums here 1 is $2k$ plus 1 and other is over $2k$. So that is for odd and the even functions, so that is what is here. So this is for the odd function, so w to the power $2k$ plus 1 is now here split into 2 parts 1 is $2\pi n$ by n and other is $2\pi kn$ by n by 2. So 2 parts I just, this is basically just w to the power $2k$ plus 1, so that is for the odd values for the even

values remember it is the sum here. So now that is the basic algorithm this algorithm now can be implemented.

So in this 2 sums can be implemented very, in a easy fashion by replacing the function values by its sum and its difference now that the whole trick here. So we have this f_{2k} that is for the, we have a series of f_n function values we have going from 0 to n by, we have 0 to n , so we have 0 to n minus 1 we have n function values now this n function values we take and we split that in to 2 halves n by 2 minus 1 and n by 2 to n and then what we do is we take the sum and the difference of these function values and replace these functions by its sum and difference.

So we will use a same array and then replace them by the same sum and difference. So that is what you are going to do, so you see that we wanted to compute for the odd and the even values of k , so 1 is replaced by the sum and the other is replaced by the difference and then we do the discrete Fourier transform of that.

So if you look at this expression here these 2 expressions, so you see this very clearly that this is the expression for the discrete Fourier transform of n by 2 functions n by 2 function values which is given by f_n plus f_n , n by 2 and this is our normal Fourier transform. So remember how did we write the discrete Fourier transform, so when you wrote the discrete Fourier transform we said that f_k is sigma n going from 0 to n by 2 minus 1 f_n plus f_n e to the power of i omega k . So that is the same thing here i omega k that is what we had written.

So now here the function to whose discrete Fourier transform we want to find is now f_n plus f_n plus n by 2 that is the sum of the 2 functions here and here the second part is the discrete Fourier transform of the difference of these 2 functions. So let me say that again, so following that we have this functional values here. So now what we do is we take the sum let us say now we are, I will write f_0 plus f_4 here and this and that I take the sum and I write that and I take the difference and I write that here f_0 minus f_4 and also multiply it by w to the power, in this case it will be w to the power n .

So that is what we would write here and then because you see the difference here is that for the even functions it is w to the power $2nk$ and for the odd functions we have the difference and there is also this extra w^n here. So I would write that also if I have to do the sum.

So I have to write this also here w to the power n and similarly I could take for all of them now this will be replaced by f_1 plus f_5 and this will be by f_2 plus f_6 and this will be f_3 by f_3 plus f_7 all the differences we would write here that is f_1 minus f_5 and f_2 minus f_6 and f_3 minus f_7 all of them multiplied my w^n . Okay and then we can do the discrete Fourier transform of each of them now you do the discrete Fourier transform of each of these values each of these function sets and then we would get the desired answer but now you see we know that this scaling, let us look at the scaling we said that the discrete Fourier transform as n squared operations.

(Refer Slide time: 45:56)

$$\begin{array}{l}
 f_0 \quad f_0+t_0 \\
 f_1 \quad f_1+t_1 \\
 f_2 \quad f_2+t_2 \\
 f_3 \quad f_3+t_3 \\
 \hline
 f_4 \quad (f_0-t_0)W^N \\
 f_5 \quad (f_1-t_1)W^N \\
 f_6 \quad (f_2-t_2)W^N \\
 f_7 \quad (f_3-t_3)W^N
 \end{array}$$

(Refer Slide time: 46:11)

If we define

$$g_n = f_n + f_{n+N/2} \quad \text{and} \quad h_n = (f_n - f_{n+N/2})W^n$$

we have

$$F_{2k} = \sum_{n=0}^{N-1} g_n W^{2nk} = G_k$$

$$F_{2k+1} = \sum_{n=0}^{N-1} h_n W^{2kn} = H_k$$

So we have n is equal to 8 here. So we will have about 64 operations here and then while this is 4 and this 4, so then we will have 16 plus 16 operations. So we have 30 operations. So by splitting this into 2 halves and then doing the discrete Fourier transform of each of them we are able to reduce the number of operations now we can split this into further. We can split this again further and do the discrete Fourier transform for each of them and that is the idea of the fast Fourier transform.

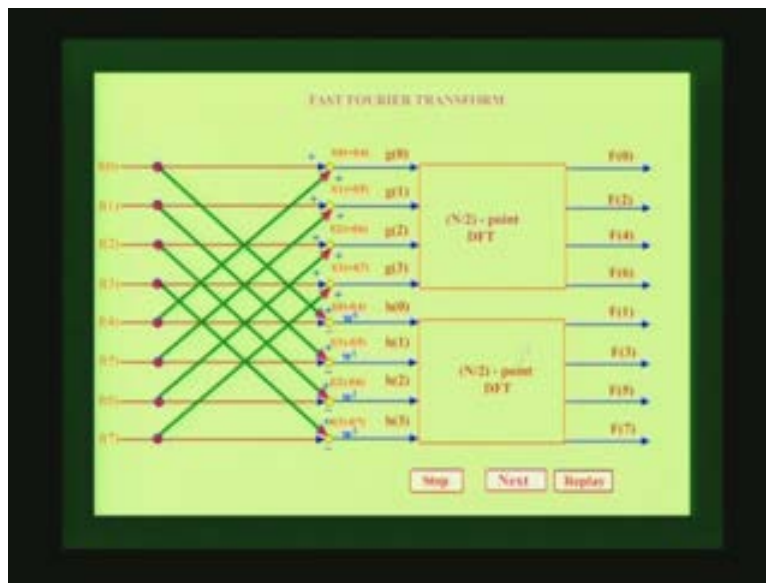
So graphically, we can show that here so before that let me summarize this. So what we are saying is we are defining new functions which is f_n plus n by 2 and f_n minus, f_n minus

f_n by 2 this is I am calling these functions would be called as g_0 . So let me call this in the notation we use this is g_0, g_1, g_2, g_3 and this we would call as h_0, h_1, h_2 and h_3 and we do the discrete Fourier transform for each of these functions here separately and that is what is written here. Okay, now these Fourier transforms would be called g_k and h_k remember now it is $2nk$ here.

So it is $2nk$ when you are multiplying when you do the discrete Fourier transform remember that this will be $2nk$ where ω here is e to the power of i times 2π by capital n into $2nk$, if you are doing the whole function Fourier transform, discrete Fourier transform we would be writing it as f_n . So remember if it is a full Fourier transform using discrete Fourier transform then we would be saying that f_k here, will be equal to $\sum_{n=0}^{n-1} f_n$, now this n is actually n by 2 this sum is reduced to half.

So, otherwise it will be $n-1 \sum_{n=0}^{n-1} g_n w^{nk}$ this is g_k function. So we have actually reduced the numbers into 2 halves here. Okay now that can be I can show this in a simple, graphically I can show this in this fashion. So I have this $f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7$ listed here then you take 2 functions f_0 and f_4 and make the sum and the difference and you write the sum here and the difference here and then you could do this for all the function values.

(Refer Slide time: 48:46)



So you do this for $f_0, f_4, f_1, f_5, f_2, f_6$ etcetera and then you get these 2 sets now this is what we call g_0, g_1, g_2, g_3 and this we call h_0, h_1, h_2, h_3 and then you do a n by 2 point discrete Fourier transform here and then n by 2 point discrete Fourier transform would give us the Fourier transform of all the k values which are even here and all the odd values here.

So let me repeat this again that is I take these function values I tabulate them here and then I take the sum for f_0 and f_4 . So this is the line which marks that f_0 and f_4 and I take the sum here and write that and then I take the difference between that f_0 and f_4 which is here that is the difference multiplied by w_0 , w to the power 0. So because I have written here remember I have to multiply it by this n .

So n here is 0, so w_0 and then in this case it will be w_1 and in this case it will be w_2 and in this case it will be w square and w cube etcetera. So that is what you have to write and that is what I have written here. So here let us replay this once again we have these functions here and you take these 2 and then you find the difference you have written here and the sum there and similarly, you find all the sums and differences for all for each of the pairs and then you take these functions here, the differences multiply them by the weight that is w to the power 0, w to the power 1, w_2 and w_3 and then tabulate them as g_0, g_1, g_2, g_3 and h_0, h_1, h_2 and h_3 and then you have a set of n by 2 points here and you have another set of n by 2 points and you do the discrete Fourier transform with that.

Let us say we stop it here we can actually do this again. So let us continue this operation but let us first look at this I can do a n by 2 point discrete Fourier transform here, now what do you get is here the f or the Fourier transform of this if this whole thing was a black box, you do not know what is going on we put in these functions here and we got the Fourier transforms out here, but when you put in f_0 , you get the corresponding Fourier transform capital F_0 and you put in f_7 and you get the corresponding value transform of f seven f this for k equal to seven and this for k equal to 0 but the f_1 will now be replaced by f_2 and f_2 will be replaced by f_4 and f_3 by f_6 .

So now, so when you put in here the function values into this black box, you put in the function values in this order the Fourier transform, we get in a slightly jumbled up order. So we are going to get it as f_0 and then we are going to get f_2 for k equal 2 k equal to 4 k equal to 6 and then k equal to 1 3 5 and seven because we have segregated the even and the odd Fourier transforms. So now we can do this continue this again we can take this box here, which has now n by 2 points in this case there are 4 points and these 4 points can be split further up. So we see that to start with we had this 8 points and then we made them into groups of 4 and then we did a 4 point dft here.

So as I said we will further divide these 4 points and then continues this process. So let us look at that in the next slide. So we have here this 8 points, that is 0 to 7 and then you had this now we had grouped them into 4 and 4 and then we said, that we could do a 4 point Fourier transform here. So let us do a 4 point fast Fourier transform again here. So that is, we could do a 4 point discrete Fourier transform or we could do a 4 point fast Fourier transform.

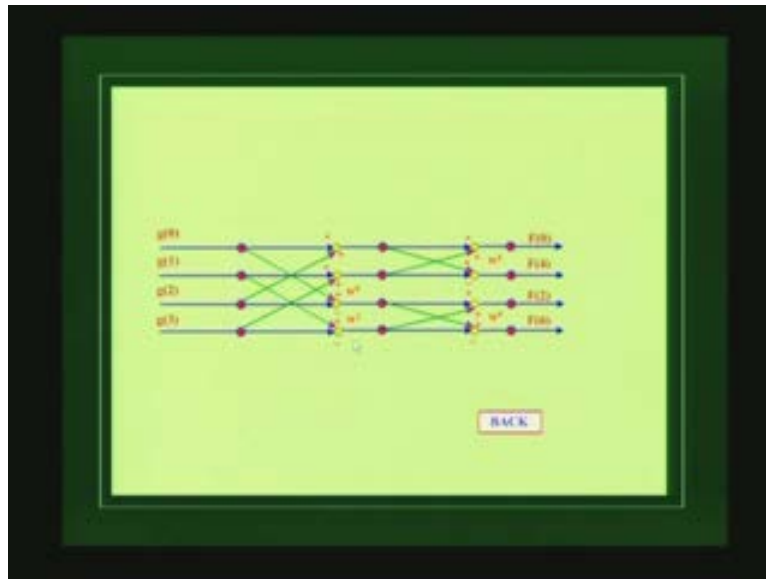
So let us do this fast Fourier transform, so that means we will again do what we did on these points here again that is, we took this 4 and 4 and we added f_0 to f_4 and replace f_0 with that sum and we took the difference between f_0 and f_4 and replace f_4 with that difference multiply it by w to the power 0 and we took f_1 the difference between the sum

of f_1 and f_5 and replace f_1 with that sum and we took the difference between f_1 and f_5 and multiply it by w to the power 1 and replace f_5 with that and so on.

So we will do exactly the same here now the input to this was remember g_0, g_1 let us just go back and see that again so we have the input as g_0, g_1, g_2, g_3 and here the input we call as h_0, h_1, h_2 and h_3 . So when we do a_n by 2 point fast Fourier transform instead of dft, we will take g_0 and g_2 find the sum replace g_0 with that and then g_1 and g_3 and take the sum and replace g_1 with that and take the difference between g_0 and g_2 and multiply it by w to the power 0 and replace g_2 with that and take the difference between g_1 and g_3 multiplied by w to the power 2 and replace g_3 with that.

So notice that difference that is when we take the difference here, we go away w_0, w_1, w_2 w square w_3 etcetera but when you do that here, then we take g_0 minus g_2 multiplied by w to the power 0 while g_0 minus g_3 multiplied by w to the power 2 similarly, here we replace h_2 by h_0 minus h_2 into w to the power 0 and h_1 minus h_3 into w to the power 2, that is what we do.

(Refer Slide time: 55:57)



So when we do that what we get is starting from these values so we will get, we have these n by 2 points and we this g_0, g_1, g_2, g_3 here and h_0, h_1, h_2 here and we do a again a Fourier transform here, fast Fourier transform by doing this split. So let us look at that here. So that is the 4 points g_0, g_1, g_2 and g_3 and then we can take the sum and the difference of g_0 and g_2 and replace g_0 and g_2 by the sum and the difference, okay. So again note that when I multiply g_2 and g_3 with w 's it is w_0 and w_2, w square w_2 , I mean w square.

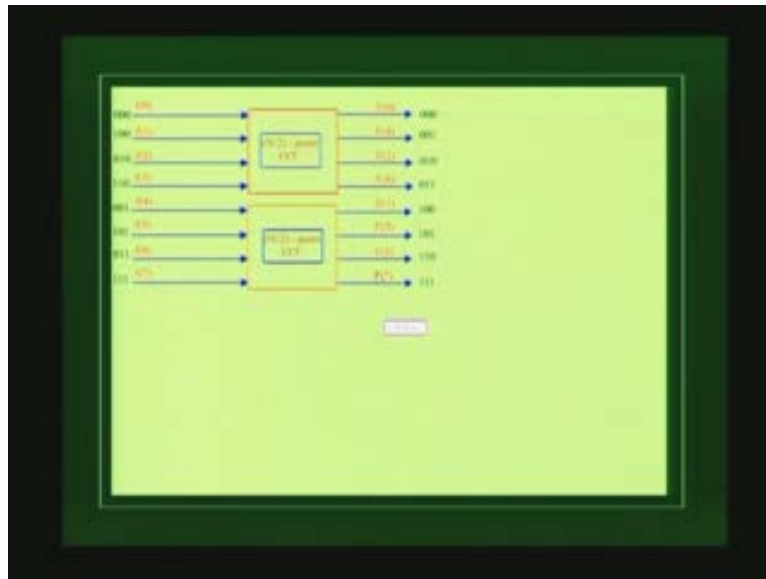
So then now we have this set of 2 points here and the set of 2 points here, now I can again do the continue this transform again by taking these 2 points and then these points as these 2 difference set. So we had 8, we just split into 4 and then that is further split into

2 each and then that too we will use and take the sum and difference and write now the whatever we got here.

Okay so we take that the sum of these 2 and replace this point with the sum and take the difference between these 2 multiply it by w to the power 0 and replace that is just the difference and we replace this point with that similarly we take the sum between these 2 points and replace this point with the sum and take the difference and replace this point with the difference. So now that gives us the Fourier transforms, so final answer when you do this cascading similarly we can do for h also.

So we could do the same thing for h , so we have the h_0, h_1, h_2, h_3 here we take the sum between h_0 and h_2 and replace h_0 with that take the difference between h_0 and h_2 and then replace h_2 with that and take the sum between h_1 and h_3 replace h_1 with that sum and take the difference between h_1 and h_3 multiply it by w to the power 2 and replace h_3 with that and now we take these 2 points take the sum, replace the first point with that and take the difference replace the second point with that and similarly here.

(Refer Slide time: 58:00)



So that leads to the full Fourier transform when you do that here. So the end result is that we have the Fourier transforms and in this order, so when you are do this thing it is as we are taking the sum and the difference in replacing those numbers with those sum and the difference multiplied by the appropriate weights what we find is that we actually jumble up the order of the Fourier transform.

So we start with an array which contains the function values at 0 in an array, let us say in this case dimension from 0 to 7 and in this case and finally what we get is the Fourier transforms which are again goes from 0 to 7 but not in the same order as we have got here. So it is not in the same order that we get here. So that is the highest value and the lowest value will be the same for the k values.

So f_0 and f_7 are but the in between values get jumbled up you can see that after f_0 the next array point that is what we had the function in the f_1 is now replaced by the Fourier transform for k equal to 4 and f_2 is say the same as 2 but f_3 is now replaced by the Fourier transform of k equal to 6. So what we see is that we start with an array which contain function values from 0 to 7, we started with that and then we are ending with Fourier transforms in the same array, the same array the functions are now replaced by its Fourier transform but not in the same order.

So now what order it comes in and how we can decipher which k value, which function which functional values replaced by the Fourier transform is what k values are they replaced for we will see that in the next class.