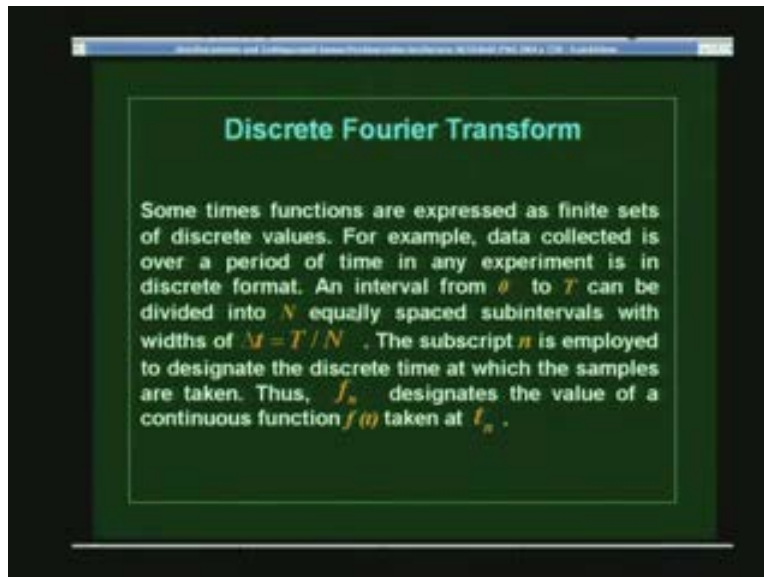


Numerical Methods and Programming
P. B. Sunil Kumar
Department of Physics
Indian Institute of Technology, Madras
Lecture - 38
Fast Fourier Transforms

In the last class, we have been looking at computing the Fourier transform of a function using some numerical method and we look two methods, we looked at are discrete Fourier transform and the fast Fourier transform. So that is, if you have the function values tabulated at certain points and then how do we compute the Fourier transform that was a problem.

So today what we will do is, we will actually look at the implementation of these techniques that is fast Fourier transform and the discrete Fourier transform for a set of values, for a set of function values which we know of that is we know the functional form and so hence we know the Fourier transform or what it should look like and then we can compute them using this numerical techniques which we looked at in the last class.

(Refer Slide time: 02:15)



Okay so we have the function values tabulated at equal intervals and we call them as f_0 , f_1 , f_2 etcetera and let us take case 8, 8 such functions that is we have to tabulate them at 8 points. So we have f_3 , f_4 , f_5 , f_6 and f_7 . So we have the function values tabulated at these 8 points and you want to compute the Fourier transform of this. So this had tabulated at equal intervals.

So we set the delta x the distance between let us say this is f of x , now we are tabulating.

So f of Δx is now t the whole time interval, whatever interval between these two is divided by n , let us say l divided by n in the case of spatial variable.

Okay we divide this to n , where n is now 8 here and then we have these tabulated function values. So the notation we were using was that this f_0 will be now be written as f subscript 0 and this as f_1, f_2, f_3, f_4 . So we use this notation f_n as the function value at n , at n th point. So that is the notation which we were using, we see that here. So we divide the total interval into n equal intervals and denote this function by f_n , that is what we looked at and then we want to compute the Fourier transform as that.

(Refer Slide time: 03:54)

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

$$F_k = \sum_{n=0}^{N-1} f_n e^{-ik\varpi_0 n} \quad \text{for } k = 0 \text{ to } N-1$$

and the inverse fourier transform as

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} F_k e^{ik\varpi_0 n} \quad \text{for } n = 0 \text{ to } N-1$$

where $\varpi_0 = 2\pi/N$

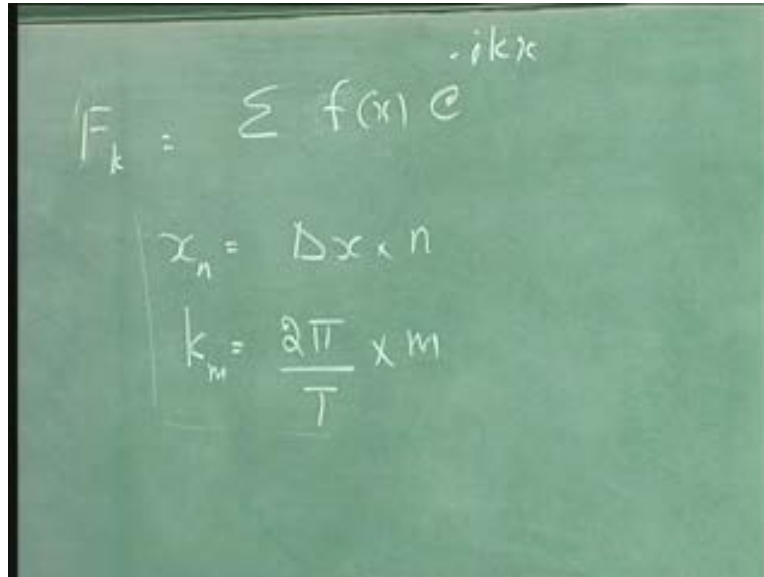
DFT requires N^2 complex operations.

So remember the Fourier transform which you are going to compute, is computed with this expression, right. So the idea I remember was that, we want to actually compute the transform and then we have, so we know that the discrete fourier transform can be written in this as f_k , we normally write it as sigma e to the power of, so we say f of x right, f of k is f of x , e to the power of minus ikx . Okay that is what we will do for all x values, discrete values of x , we would sum over all x values.

Okay so whatever limit are that is what the actual discrete form is but now we have replaced the x , the x is now replaced by, so now we are going to write x as some Δx times n that is our x and similarly, our k value is actually 2π divided by whatever be the interval is in this case t is the total interval and then, this total interval multiplied by some m is our k_m .

So we have a x_n , okay so we are going to replace this continuous variable x by a discrete variable x_n which is Δx into n and k_m by some, this the smallest k vector which we can get which is 2π by the total interval divided by, multiplied by some number integer m . So we have these two discrete values and that is what we are going to substitute there.

(Refer Slide time: 05:40)



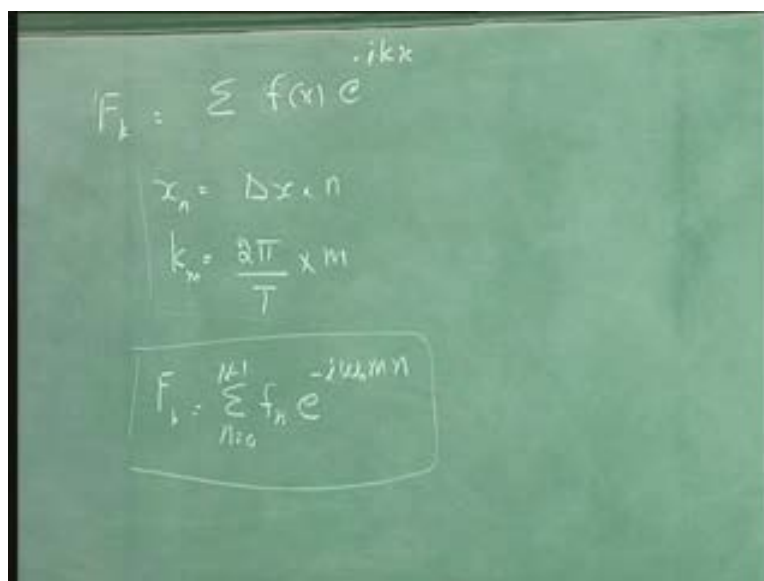
Handwritten equations on a green chalkboard:

$$F_k = \sum f(x) e^{-ikx}$$
$$x_n = \Delta x \cdot n$$
$$k_m = \frac{2\pi}{T} \times m$$

So now when you substitute that in this we get the expression which I just now written, so I just say that f_k is now become $\sum f_n$ that is our function value at x_n into e to the power of I will say minus i omega 0, we set m times n.

So now omega 0 is this value and omega 0 times n, that is what we are going to, that is the expression which we are going to write and now this sum goes from n equal to 0 to n minus 1 where n is the, so n is the total number of points. So if it is 8, it goes from 0 to 7 that is the sum which we are going to write.

(Refer Slide time: 06:28)



Handwritten equations on a green chalkboard:

$$F_k = \sum f(x) e^{-ikx}$$
$$x_n = \Delta x \cdot n$$
$$k_m = \frac{2\pi}{T} \times m$$
$$F_k = \sum_{n=0}^{N-1} f_n e^{-i\omega_0 n}$$

So that is the that is basically the sum which we want to compute. Now let us try to use this function here and this form of the, this expression for the discrete fourier transform first and compute the fourier transform of a function of this form that is a gaussian, let us take a gaussian. So we have a function of this form. So now I am going to write I have a function which is of this form.

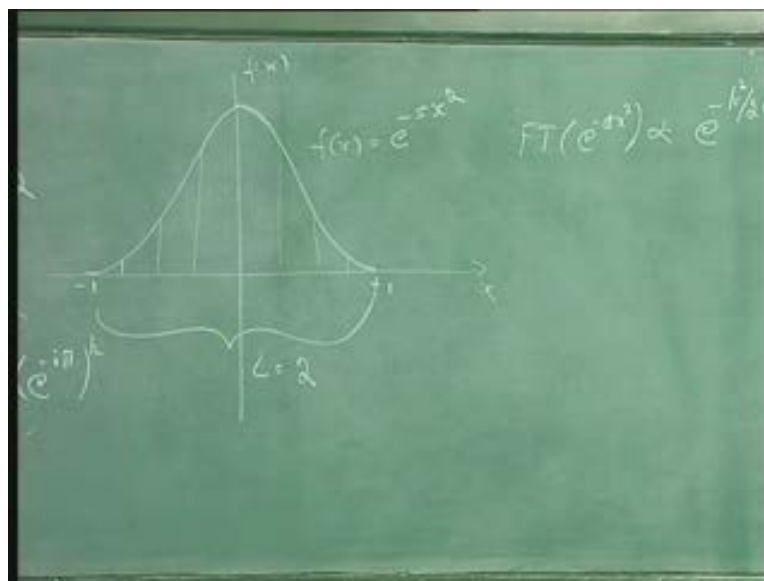
So that is my $f(x)$ and this is x let us goes from, it goes from now for specific example we take it from minus 1 to plus 1. So it goes to 0 by that almost to 0 by that time and actually the function form which I am going to use in the program is e to the power of minus 5 times x square, I know what the fourier transform of that is, so this will be the fourier transform of this would be some factor which is a normalizing factor multiplied by e to the power of minus k square by 20.

So let us take this function and let us descriptize this into certain number of points like this. So one is this and then some equal intervals we will descriptize this function and we will call them as f_0, f_1, f_2 etcetera. So, care should be taken that we are going to call this as $f_0, f_1, f_2, f_3, f_4, f_5, f_6$ and f_7 like that.

Okay, so because of that my f_0 value, so x_0 value should be minus 1 not 0. So I should had x_n as Δx times n minus my total interval divided by 2 that is the x_n I should actually substitute here and then this would be now, 1 is 2 the total interval okay that should be my form.

So my x_n interval x_n is given by Δx times n minus 1 by 2 because I start from here. So 1 is my 1 is this total interval 1 which is equal to 2. So now that is what I have and then if I do this this form which is given normally in any text book which you look at for the discrete fourier transform what you see is this.

(Refer Slide time: 09:37)

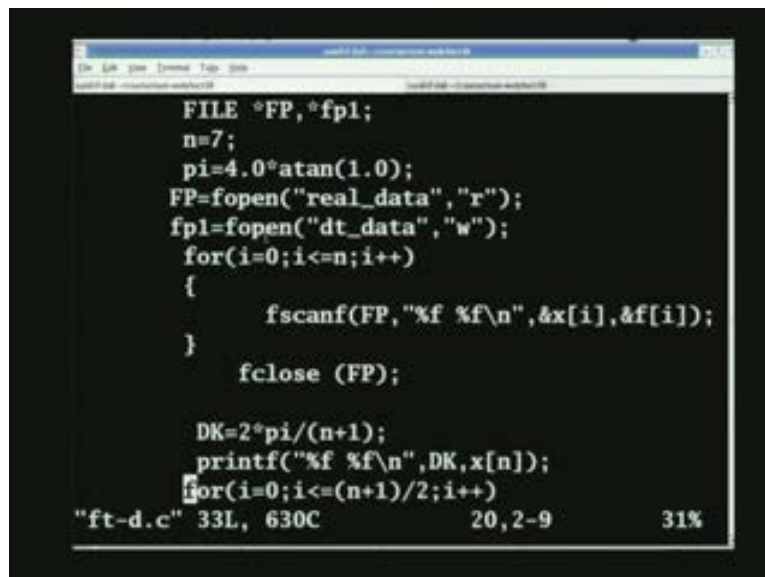


Okay so now that gets modified a little bit, now that will get e to the power of, an additional term e to the power of minus i ϕ to the power k . Okay so you will find this this should be k here, k is the integer. So you will get an extra factor here which is multiplying this and it would change the sign of all the odd k values. So that is something you should be careful about that is, if you are going for symmetric interval and then this x_n is actually having an additional factor here which would show up in the fourier transform in the discrete fourier transform by this quantity which you can see by substituting this into this expression straightaway.

So actually what you would be computing is this quantity just to get that correct and that is our program which you will look at now. So we look at a program which implements this. So here is the program, okay this program is a discrete Fourier transform. Okay so in this I am reading off the, I have a file called real data which is actually the data which contains this function.

Okay so we can actually plot that and then see first. So we have, so we will plot this using, here is my plot of this function I guess you could see this that I have a gaussian function here which I have descriptized at 1, 2, 3, 4, 5, 6, 7, 8 points. So I have the 8 points this is gaussian when descriptized.

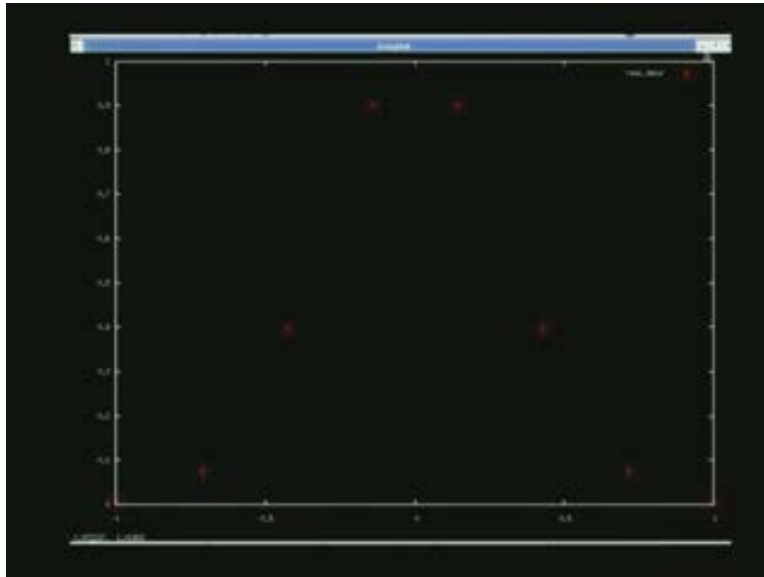
(Refer Slide time: 10:04)



```
FILE *FP,*fp1;
n=7;
pi=4.0*atan(1.0);
FP=fopen("real_data","r");
fp1=fopen("dt_data","w");
for(i=0;i<=n;i++)
{
    fscanf(FP,"%f %f\n",&x[i],&f[i]);
}
fclose (FP);

DK=2*pi/(n+1);
printf("%f %f\n",DK,x[n]);
for(i=0;i<=(n+1)/2;i++)
"ft-d.c" 33L, 630C          20, 2-9          31%
```

(Refer Slide time: 11:14)



I will use a different symbol, so you will see it better. Okay so here you can see it better here is the my gaussian function which is being discretized now at 8 points now that is the function which you are going to use to compute the, this is the the Fourier transform of this is what we are going to compute using the discrete Fourier transform. So let us look at that, this real data contains that 8 data points. So I have 8 data points, so n that is 0 to 7.

So I put n equal to 7 here the value goes from 0 to 7. So starting from 0 to 7 it will read off that x and f values actually as you see that when you compute the discrete fourier transform actually where the x values for which this function is evaluated, that information is not here. So that is what I am warning you about this one.

Okay so when you look at a standard text book and see that formula here we do not have the information about where the x values are as long as it is equally spaced this function should work, but to get these signs correct the sign of the Fourier terms, the Fourier terms transform correct we need to also multiply it by e to the power of minus i ϕ by k depending upon what your interval actually is where you start from. Of course you do not have to do this, if the if your origin was here that is if this whole thing is shifted then you do not have to use that here there is a shifting factor there which you have to be careful about.

Okay so now you have the function read off here. So we will read the function values into this we are going to use only the f we are not going to use this x_1 values at all, we are going to use only the f values here now this is our what I call in this thing as k my basic omega naught. So that is 2π by n plus 1 capital n is n plus 1 here capital n is 8.

Okay so I am reading from 0 to 7 so n is 7 but number of points is 8. So that the smallest k value I have is 2π by n plus 1 that is 2π by 8 that is my omega naught. Okay so

then for each of these k values, now I substitute I find this transform now we look at this here again that I want to find this sum.

So now to find this sum this is a complex number. So now many of the languages like `c` does not handle complex number. So I split this into two and I would write this as $\sum_{n=0}^{n-1} f_n$ now this would be written as \cos and \sin I will write it as \cos of $i \omega$ naught, \cos of ω naught k_n plus ϕk that is what I would have and then one that is first step and then I will say that this plus i times minus i times \sin of ω naught k_n plus ϕk .

Okay so that is the sum, now I split this into 2. Okay so that I can evaluate this term and this term. So that is the imaginary part and that is the real part. So, I do the real part and the imaginary part separately here. So i times, so I have the imaginary and the real part and the imaginary part separately, okay so that is what I have in this program here. Okay so I have the real part here and which is \cos of the ω naught multiplied by the j and the i here.

So i is my the k values, so i is the 1 which multiplies the ϕ also that is my k on the board what I have written this is k and this is j value is the n , when we say sum over so when we say j value goes over the function is evaluated at different j values and we sum over all j values that is 0 to n .

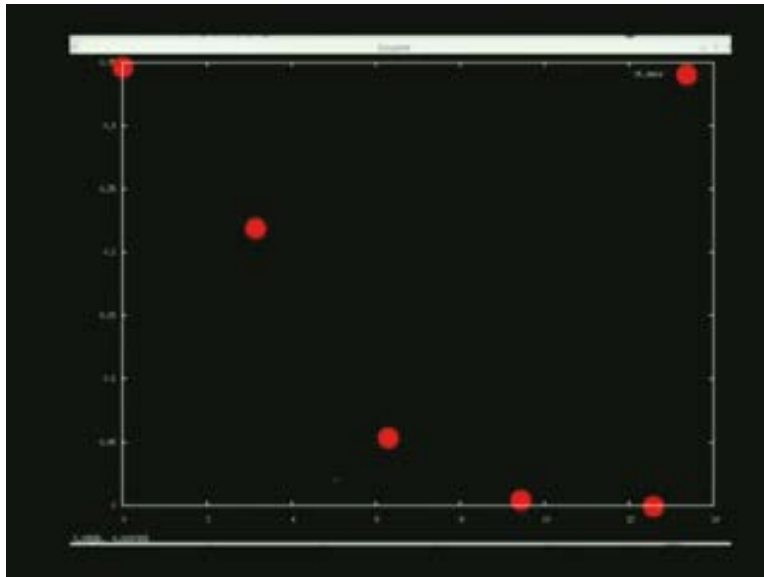
So n little n which is 7 in this case, so I sum over all that i initialize the sums for the real and the imaginary part and I do the sum so then I have the Fourier transform. So now I can do this sum explicitly and I would get the Fourier transform for each of these i values up to n plus 1 by 2 that is what I am doing.

So I take the first $4k$ values that is n is 7, so n plus 1 by 2 is 4. So I get the first $4k$ values here, $5k$ values and that is what I am going to print out into this file which is called which I call as `dt_underscore_data`, so discrete transform data. I am going to put that into that file, I am going to write the function values the real part which is sum the real part and sum one which is the imaginary part.

Okay so I will write these two into this file and I will also write the k value, now the k value is ϕ times i that is my k value which I have, so I will write that also. Okay we will remember the largest the smallest k value in term in the real units in the smallest k value is 2ϕ by this distance. So that is the smallest k value I have, so 2ϕ by that distance that distance is 2 the smallest k value is actually the k small smallest k value is actually 2ϕ by 2, so that is ϕ .

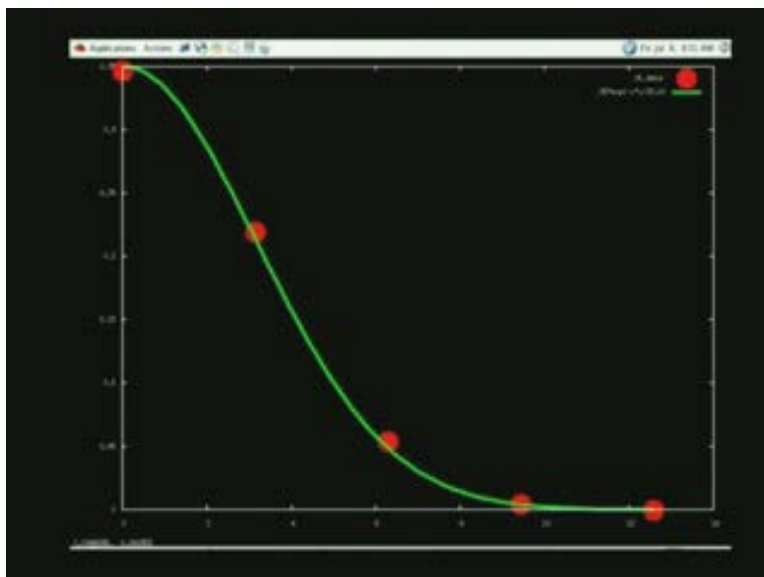
So that is what I am trying then ϕ multiplied by the integer multiples of that ϕ are the other k values, so I am writing that here. So I have the different k values and the real part correspond for the Fourier transform and the imaginary part of the Fourier transform that is what we have. So now this file I will plot this file now this file is I called `dt_data`, again we will use the same symbol and so now that is the Fourier transform. So I have done only one part of the thing it is symmetric.

(Refer Slide time: 18:32)



So as I said the Fourier transform of the gaussian is a gaussian, so now it is symmetric. So I plot only one part of the Fourier transform now which starts from 0 to 12 here. So this is one part of the Fourier transform and the other part is symmetric. So I did not compute that, so that is why I said only the first 5 of the fourier transform which we have computed here. So now we see that this is a pretty good gaussian and it actually fits in with exponential minus k square by 20, it actually fits into that I can show you that here we can see that it fits in very well through that curve let us increase the line width, the width.

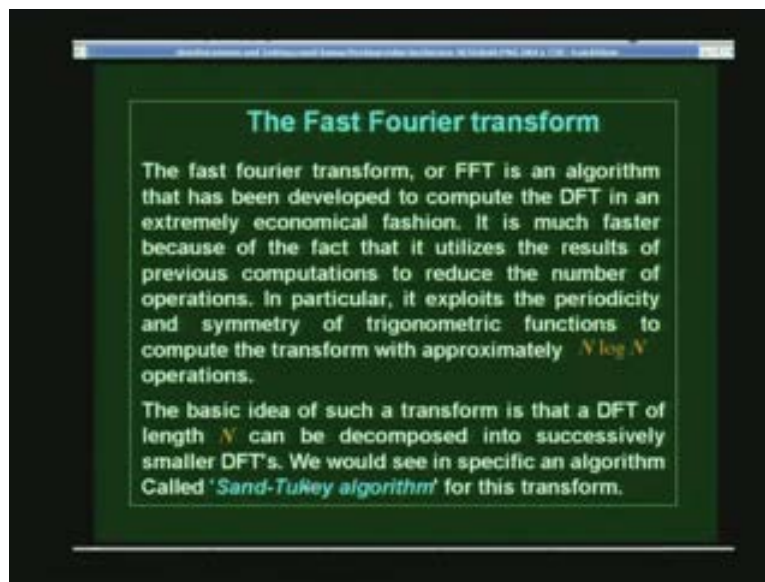
(Refer Slide time: 19:50)



So you can see this function this is the actual function, this green curve here is the function which I would have got if I had done the fourier transform analytically and I did an 8 point discrete Fourier transform and I get numbers which are pretty good. You can see that they actually go almost on to this into this line and considering the fact that we do not use 8 points to compute this, this is a reasonably good result that is a discrete Fourier transform implementation very simple to implement but as I said in the last class this requires n to the power of 2 operations. So it is not computationally very efficient.

So we need n square complex operations, so instead of this n square complex operation we have done 2 times n square real operations by splitting this fourier transform here into two different that is a cosine and a sine transform and our calculation here the imaginary part of this is 0, in this particular, for this particular function. So we see that this is easy to implement and we can compute and we can obtain a reasonably good accuracy.

(Refer Slide time: 21:06)



So now we look at the fast Fourier transform implementation as we said in the last class that this fast Fourier transform technique reduces the number of operations to $n \log n$. So hence it is the popular the most used method for computing discrete Fourier transforms. So what was the technique the technique was that we would have the function values and we would split them into 2 halves and then take the sum between these 2 and then replace this function by the sum and take the difference between these 2 and replace this function by the difference multiplied by w to the power 2, w being e to the power of minus i omega naught, where omega naught is the same as just found out 2π by n that was the technique. So we just briefly go through that.

(Refer Slide time: 22:07)

The Sand-Tukey Algorithm

We proceed by assuming that N is an integral power of 2,

where $N = 2^M$ M is an integer.

In a DFT we recall there are N^2 number of complex operations required.

The equation can also be expressed as

$$F_k = \sum_{n=0}^{N-1} f_n W^{nk} \quad (1)$$

So we write the fourier transform in this case, the discrete Fourier transform as now f_n to the power nk that is what we write and then this w here remember is e to the power exponential minus i 2π by capital n , where n is number of points. So that is exactly this is the same as what we used in the discrete Fourier transform case, so we use this and then we just write that into 2 halves. So it is n equal to 0 to n by 2 minus 1 and n starting from n by 2 to n minus 1 that is what I just now said.

(Refer Slide time: 22:43)

where W is a complex valued function defined as $W = e^{-i(2\pi/N)}$. Now we divide the sample in half and express equation (1) in terms of the first and last $N/2$ points:

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

We split this whole interval into two intervals and write this as two different sums and then we notice that I can change a variable here and then make the interval, the range of the sum the same.

So I will replace this n by m minus n by 2 and then write the sum as m equal to 0, n equal to 0 to n by 2 and m equal to 0 to n by 2 minus 1, where here now f instead of n here now it is f_m plus n by 2.

(Refer Slide time: 23:16)

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 that is the difference and now we have 2π by n times k instead of n , it is m plus n by 2 , we have a change of variable here. So instead of the sum going from n equal to n by 2 to n minus 1 we have made it to 0 to n by 2 minus 1 . So that these two limits of the sum are the same and now I can combine these 2 and write it in this fashion and then I can write the Fourier transform, discrete Fourier transform now as f_n plus e to the power of minus $i\pi k$ $f_{n+N/2}$ e to the power of minus $i2\pi kn$ by N and that is what ω_0 $2\pi/N$ is our ω_0 and e to the power of $i\omega_0 n$ is w .

So this part is actually w to the power kn that is what we say. So now we notice that this simply means that for all the odd k 's this is the plus sign it is a minus sign this is for odd k it is a minus sign for even k it is a plus sign. So we can separate out the odd and even sum that is the key observation for the implementation of fast Fourier transform. So this odd and even k 's can be separated here that is what we do and then we will write for the even function as plus, right even function now is plus.

(Refer Slide time: 22:44)

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

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

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

So now k is replaced by 2k, so k goes from 0 to n by 2 and then we have all the even values computed here and the even values it is plus because e to the power of i 2 phi k is minus 1 to the power k and it is plus 1 for even k and for odd k values, we know that it is minus sign here right. So because e to the power of i phi k is minus 1 it is negative for an even, for odd functions, odd values of k.

(Refer Slide time: 25:18)

and for odd values

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

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

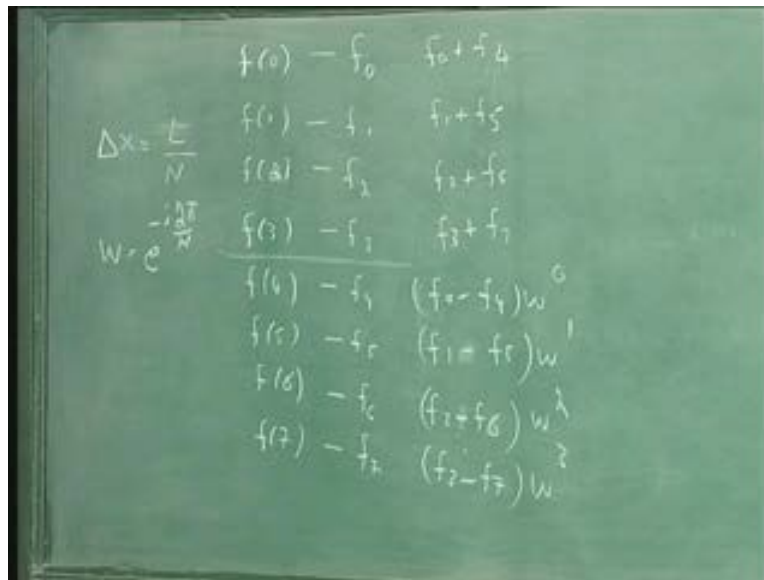
for $k=0,1,2,\dots,(N/2)-1$.

So we have the even and odd separated out. So let us write that here and that is what we are going to use in the program now, so and we will do that for the same function again so we are going to write this now as f_0 minus f_1 and f_0 minus f_4 and here we are going to

write f_0 plus f_4 , f_0 minus f_4 into w_0 and here we are going to write f_1 plus f_2 , f_5 and we are going to write here f_1 minus f_5 w_1 and here f_2 plus, sorry I made a mistake here it is f_0 minus f_4 into w_0 and here it is f_1 plus f_5 into w_1 and then here we have f_2 plus f_6 and here we have f_2 plus f_6 at w square and here we have f_2 minus f_2 plus, f_3 plus f_7 and here we have f_3 minus f_7 times w cube.

So that is what we have all the differences, the difference between the functions multiplied by the weights where, w remember is e to the power of $i 2 \pi$ by n that is w , okay so n is the total number of points.

(Refer Slide time: 26:52)



So then this is the fourier transform of the, this is the first step of this thing and then we take this we call this as g_0, g_1, g_2 and g_3 and we call this as h_0, h_1, h_2 and h_3 that is what we did and then we continue this process again. So now we did this and we said that here I will replace by g_0 plus g_2 and this I will replace by g_1 plus g_3 and this is g_0 minus g_2 multiplied by w power 0 and this is g_0, g_2, g_1 minus g_3 multiplied by w to the power 2 notice that it is 2 and here, it is h_0 plus h_1 and here h_0 plus h_3 h_2 and here it is h_1 plus h_3 and this is h_0 minus h_2 w power 0 and here h_1 minus h_3 w power 2.

We can do one more step, we can take the difference between now we have these quantities now we can take the, so we have divided this here 4 into 2 and now this has become 4, we have 4 divisions, 4 sets now we can this further here and take the sum and the difference and write that again.

(Refer Slide time: 28:32)

$$\Delta x = \frac{T}{N}$$

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

$f(0) = f_0$	$f_0 + f_4 = g_0$	$g_0 + g_2$
$f(1) = f_1$	$f_1 + f_5 = g_1$	$g_1 + g_3$
$f(2) = f_2$	$f_2 + f_6 = g_2$	$(f_0 - f_4)W^0$
$f(3) = f_3$	$f_3 + f_7 = g_3$	$(f_1 - f_5)W^1$
$f(4) = f_4$	$(f_0 + f_4)W^0$	$h_0 = h_0 + h_4$
$f(5) = f_5$	$(f_1 + f_5)W^1$	$h_1 = h_1 + h_3$
$f(6) = f_6$	$(f_2 + f_6)W^2$	$h_2 = (h_0 - h_2)W^0$
$f(7) = f_7$	$(f_3 + f_7)W^3$	$h_3 = (h_1 - h_3)W^1$

Okay so now that is the key, way of doing the discrete Fourier transforms. So at the end of it we have the Fourier transforms listed here. So that is the observation, so we have f_2 k and then we continue this and writing its as the discrete Fourier transform and that is what we did.

(Refer Slide time: 28:53)

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 let us write it here in this fashion, so we start from here we want to take this as two sets. So look at that each step I go forward I divide the interval into 2 to the power of that step intervals, it is the first step I do 2 to the power of 1 interval I have the whole set divided into 2 to the power of 1 interval that is 2 intervals and the next step I have 2 to the

power of 2 intervals that is 4 intervals that is 1, 2, 3 and 4 and the next step I should have to do 2 to the power of 3 that is 8 there are 1, 2, 3, 4, 5, 6, 7, 8,. So it will be I have 8 intervals and that is the n because I have only eight points

So that means that the number of points to start with should be a power of 2, otherwise I cannot do this now that is the restriction we have with fast Fourier transform. So number of points you start with should be always a power of 2 in this case we have taken 2 to the power of 3.

So it is a restriction on this, so now if you do this now if you take the sum and difference between these 2 functions and that will give us. So the sum and difference the sum between these 2 will give us f_0 and the difference between these two will give us f_4 etcetera. Okay that is what it is and we also saw that what we get at the end is a jumbled up version of what we started from here now the f_0 will be replaced by the fourier transform k equal to 0 but f_1 will be replaced by the fourier transform for k equal to 4 but f_2 will be replaced by the Fourier transform for f equal to 2, sorry that is f_0 and that will be f_4 and that will be f_2 and this will be f_6 .

So that is how it goes so f_3 will now be replaced by f_6 etcetera. So the way to see is the following that I have the function values written in this order now this is in binary 01, this is the least significant bit and that is the most significant bit. So I start with 0,1 this is 2 and that is 2 plus 1, 3 and this is 0 plus 0 plus 2 to the power of 2 that is 4 and this is 5, 6 and 7.

So that is what we write it as in this in binary and the result we finally get would be in this form this will be replaced by the k value 0 itself and this will be replaced by 0 0 1. So we interchange the most and least significant bits, so that is what was 1 become 4 and what is 0, what is 2 will become 2 again.

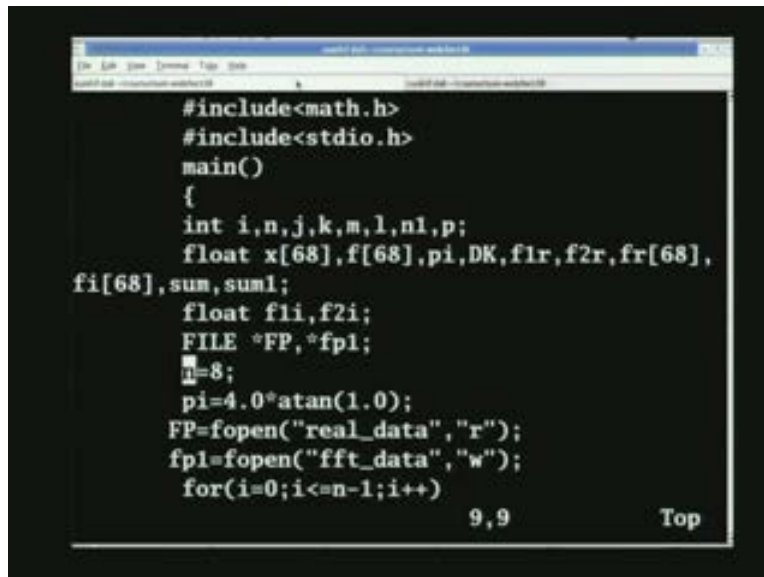
(Refer Slide time: 33:17)

0	000	—————	000	0
1	100	—————	001	4
2	010	—————	010	2
3	110	—————	011	6
4	001	—————	100	1
5	101	—————	101	5
6	011	—————	110	3
7	111	—————	111	7

So 0 1 0 and this will now go to 0 1 1, so that is why it is 6. Okay so what was 3 will now become 6 now this goes to 1 0 0. So that is we had here as 4 now that became 1, so 4 becomes 1 and then this will again remain as 1 0 1 because it is symmetric now this becomes 1 1 0 and this will become 1 1 1. So we had here to start with 0, 1, 2, 3, 4, 5, 6 and 7 and the Fourier transform order will be again 0 and now this is 4 and this is 2 and this is 6 and that is 1 and this is 5 and this is 3, that is 7.

So that is the order in which it will come so it is not difficult to decipher what it is so it will come as the Fourier transform, if you pack the array with the functions in this form in this given as f 0, 1, 2, 3, 4 etcetera those functions by this algorithms which we just looked at Sand-Tukey algorithm, if you use that those functions would be the that array elements will be replaced by the fourier transform but for k values it is pointing to 0, 4, 2, 6, 1, 5, 3 and 7.

(Refer Slide time: 34:57)



```

#include<math.h>
#include<stdio.h>
main()
{
int i,n,j,k,m,l,n1,p;
float x[68],f[68],pi,DK,f1r,f2r,fr[68],
fi[68],sun,sum1;
float f1i,f2i;
FILE *FP,*fp1;
n=8;
pi=4.0*atan(1.0);
FP=fopen("real_data","r");
fp1=fopen("fft_data","w");
for(i=0;i<=n-1;i++)

```

So that is what we will get that is one points which you should keep in mind that first point is that we need a 2 to the power of n points to implement this algorithm and the second is that the array elements of the functional value array elements now replaced by its Fourier transform but in this order 0, 4, 2, 6, 1, 5, 3 and 7 that is obtained by reversing the bits, the most interchanging the most and least significant bits in the in this order that is what we will see now.

We will see an implementation of how we implement such a algorithm in a program this is what we are going to see now. This is little more complicated than the discrete Fourier transform for obvious reasons but much faster and once we have implemented it is of course, it is much more easy to execute. Okay so let us see this now, we have the same thing that we have the function values evaluated here now one more problem with this

thing is again that these w 's here which I have written remember they are complex numbers.

So when I say w to the power 1, 2 etcetera these are complex numbers. So since languages like c does not handle complex numbers we have to handle the real and the imaginary part of each of them separately for each of these functions we have to have real and imaginary part handle separately. So I do that with starting from here itself, so this function has a real and imaginary part in principle even the input data could be a complex number. So we say this as a real and imaginary part that is what I am going to call f_r and f_i .

So my array f_r and f_i actually handles that real and imaginary part of the function, we are again doing an 8 point fast Fourier transform. Our input in this particular case is real, so I am only reading off f_r , f_i is put equal to 0 it is a real function. So that is what the input value is, so now once we have the function value just print that out here not necessary actually.

So we can remove this here and then we have now the question is how many steps do we have to go, so I said that we have to go in this steps. So till we reach step everything is only one function number of divisions is equal to the number of points which we have, so in this case we have 8 at the end the number of divisions should be 8.

So that is what we have to do so we have gone 2 to 4 to 8 that is what we will do, so that many steps you have to do in this case it is 3 steps. So once we know that the n can be written as 2 to the power of some m and that is the number of steps which you have to go so that we have to compute this m that is the first step so first step if you are given n we need to compute what the number of steps we have to go and that is given by $\log m$ by $\log 2$.

So that will be in this case it will be 3, in general it will be $\log m$ by $\log 2$ because we have to write our n as m to the power of 2 to the power of m . So I have to start with I have said that I have introduced a new integer m and I have put m equal to n and then I compute the number of steps which I require with this case I will be 3 remember that and then that means I have to through 3 steps.

So this is the loop this loop here, this particular loop here is the one which would do that. So this loop goes 1 times, so that means it goes through step 1, step 2 and step 3 in this case so that is the outer most loop so that loop is to go step 1, step 2 and step 3. Okay so this division it will do. So in the first division that is when k equal to 0 in the first loop, we will divide m equal to we will divide m at by half remember m was n . So we divide that into 2.

So that whole full interval, so n_1 tells you the number of points in that interval, in each interval how much is that number of points start with the whole thing was one interval, so n_1 is same as m , m is same as n .

So take a minute to notice this, so I need to divide my interval, I mean to divide my whole set into several intervals and I should have 3 such divisions in this case because there are 8 points. So in the first set I have the whole thing as one interval that is n_1 is m and then where m is n which is 8 and then I divide that into 2, so m is m by 2. So that is what I did.

So this is basically saying that I take this whole set as one unit and then I divide that into 2, now this is my n_1 and this is m , m is the number of points inside this and then I have my k as 2ϕ by n_1 , n_1 is the interval and I know this is the smallest k possible.

So now there is a reason why one writes it like 2ϕ by n_1 because notice that what we multiply here is w to the power 0, w to the power 1 and w power 2 and w power 3 that is in the first step when you go from here to here the multiplication is by 1 power 1, 2, 3 but in the next step the multiplication is by 0 and 2. So we do not want to, that is difficult to implement in an algorithm. So what we do is we always multiply it by 0, 1, 2 etcetera in all intervals but we change the w itself, the w itself we changed by dividing multiplying this power by 2.

Okay so our omega naught in this case the way have written here the omega naught is the same it is 2ϕ by n here 2ϕ by n here. So what I do is I write w to the power 1, w power 2, w power 3 in this step using omega naught as 2ϕ by n and in the next step instead of writing w_2 , I will write w_1 w to the power 1 but I change my omega naught to omega naught into 2 that is this will become ϕ by n by 2.

So that is what the reason why I write n_1 here. So in the first step it is a full interval and in the next step next step when I go for k equal to 1, you can see n_1 will be replaced by m value, m value was half that. So that means I will actually double my omega naught, so that serves the purpose instead of, for change in the power here to 2, I will change the omega naught I will double the omega naught. So that is what both are same, so now then I have to do that sum and difference and that is what has been done here.

Okay so now what have we done we just replace this into the first the whole interval replace into two halves and then I find the sum and the difference here that is what this point will be doing now these two loops are basically doing that so how many times I should do that, that is depends on which interval you are now that is done here.

So n_1 is the size of your interval, so now this interval to start with n_1 is same as the whole n so it is only one interval so I am just going to take the whole thing. So this loop in the first k value this loop runs only once j equal to 0 it will only run because n_1 is the same as n which will be then greater than n minus 1, if you do that. Okay so the first step I go here and I just take the sum and difference here.

So okay I replace, I store $f_1 r$ as the real part of a function and $f_1 i$ as the real part of function at i and $f_2 r$ and $f_2 i$ are the real part of the i plus m and the real and imaginary part of i plus m . So what am I doing, I am saying that when I go in the step, I have to do this divide this to 2 and I have to take the sum it has real and imaginary part.

So that is i and that is i plus m where m is the number of points in that interval which is n by 2 to start with and I replace the real and imaginary part of this f_0 by the sum that is what I have done here which is my g_0 actually, the real and imaginary part of g_0 and I take the difference between these 2 multiply it by w_0 and write the real and imaginary part and store it the real and imaginary part of f_4 .

So I do that here and then in the next step when I go, I have to do this twice for each interval now the next step when I go, I have to do this twice because now here I have to do it only once from here to here but here I have to do between these two and then between these 2 right. I have to take g_0, g_2, g_1, g_3 and then h_0, h_2 and h_1, h_2, h_3 and that is what my j loop is doing here.

So this j loop is basically doing that this is telling that how many times you have to do it, the first time you have to do it only once because n_1 is just n . So this loop will be only for j equal to 0 but when we go to the next loop the next time that is k the next k value n_1 is going to half, so that means I have to do this twice. So this j loop will run for 2 values j equal to 0 and j equal to 1.

So that is what it will do that, so that is the basic idea here. So I have to do this replace I have to do this twice in that particular case. So the j value goes then not 0 to 1 sorry 0 to and then next 1 would be n_1 would be now 4. So this j value will be 0 and 4, so that means that I will do 0 to 3, 1 set and next j value is 4. So I start from 4 to 7 in another 1 because I have that functions here replaced by 4 will be this h_0 and 7 would be h_3 for f_7 it will be h_3 .

So in the first set I do 0 to 4 and in the next set I will do 0 to sorry 4 to 7, 0 to 3 and 4 to 7 sorry that is what you will be doing here and that is what it is saying here saying I should run from j value which was 0 in the second step first j value is 0 and to m minus 1 which will be three and the next j value is j plus n_1 which is 4.

So i starts from four and goes all the way to 7, okay so that is the sum and the difference again we will replace. So now you keep doing this loop till we finish that, that is we will reach 1, 1 is 3 here. So we will do it in 3 loops and at the end we have the function values now only thing one more thing which you have to note here is that when I do the difference I do f_r and f_i and f_r i plus n and f_r i minus n sorry f_i i plus m .

So now i plus m is replaced by this the difference right. So now there are 2 points here, so I have the real and the imaginary part. So the real part has 2 contributions one comes from the, so you have the this w to the power one now that is written as \cos minus i sine. So we have this w_0 .

So remember w to the power n is nothing but e to the power of minus i 2 ϕ by n into n , now that has 2 parts that is \cos of 2 ϕ by n into n and minus i sine of 2 ϕ by n into n . So when I compute the, this itself is a complex number f_0 and f_4 , f_0 minus f_4 is a complex number and now that I am going to multiply by another complex number.

So I had to pick up the real the two contributions to real and imaginary parts. So that is what I have done here one comes from the real part of $f_1 r$ minus $f_2 r$ multiplied by the real part of w to the power n and then the imaginary part of f_1 and f_2 multiplied by the imaginary part of w that is the real part contribution and similarly for the imaginary part and with appropriate signs.

So that is the whole process so then we can now write this function again we have to divide this by n the number of inter points which we use. So final value which you get here at the end divided by n is the Fourier transform and then we can compare that Fourier transform which you obtained with the Fourier transform which you obtained using the discrete function. So that is, so now that is been plotted here. So the value which if you run that what we get is what is, so that is the value which we get.

So now this is we can see is now we have to compare this with, now let us compare it with this now you look at this and this value here I have not written the k value I have just numbered it as 0, 1, 2, 3, 4, 5, 6, 7. So this is the order in which the Fourier transform is going to come out and this is the real part and this is the imaginary part and the one which I am showing here at the bottom and this is the again this is the real part and the imaginary part of the first $5k$ values, the first $5k$ values which we have obtained using the discrete Fourier transform and you can see that the numbers exactly match but the order is completely different.

So we have the same number but the order is completely different. Okay you can see that 0 is the same as 0 and now what is one is now here what was one here has now gone to 0, 1, 2, 3, 4 here. So what is 4 here has become 1 there that is what we expect remember what was. So let us look at this way, so what is 0 here is the same as 0 there, now what has come out as one in the fast Fourier transform is actually 5 here, 4 here 0, 1, 2, 3, 4 and that is what we see here we said that what will come out as one the first, this 0 and what comes out as one is actually 4 in the element, the array number one in the of the data will be replaced by the Fourier transform corresponding to k equal to 4 and now array number 2, the array element 2 that is f of 2 will be replaced by Fourier transform 2.

So the discrete Fourier transform and the fast Fourier transform should match at 2 that is what we would see so this is discrete Fourier transform's k equal to 2 and matches with the fast Fourier transform's k equal to 2. So ignore the signs here there is something more to the signs which I will just tell you here right now, we just look at the magnitudes and so here we get the same matching. Okay so the discrete Fourier transforms 2 matches with the fast Fourier transforms 2 matches with that of the discrete Fourier transform but remember the element number one here matches with the element number 4 in the discrete Fourier transform. So similarly, we can see other numbers also.

So they match exactly like this. So now there is a problem with the sign now that problem with the sign comes from the fact that we had this e to the power of $i\phi$ which we do not take care in this particular Fourier transform. Okay so now if you have this code and in fact if you have this particular code and then we have one more part to be written which I

did not show here that is to reorder the Fourier coefficients. So now we have to I have one more part to this program which would actually reorder this fourier coefficients and give us as f_0, f_1, f_2, f_3, f_4 etcetera.

So in that order, so we have to have an algorithm which would actually do this bit reversing and give us the correct sequences. So that would be the end of this section, so that is what we would want to look at here. So now we could actually add one more part to this and that is what we will see next that is how to actually reorder these coefficients such that we get the same order as the function value which has been put in.

So to get the fourier transform into this of this array of functions in the correct order, we need to do a bit reversal and that bit reversal is done in this loop, the following loop. So remember those f_r 's and f_i stores the fourier transform and we saw that in the 8 point fast fourier transform which we had done we had the order as 0 and then we got it as 4 and then 2 and then 6 etcetera.

So we want it as 0, 1, 2, 3. So now to get that we do the following thing that is we start with an integer j as n by 2 that is 4 and then we run a loop from i to n minus 2 in steps of one. So whenever i is less than j we do we interchange between i and j , so there is no, I am not going to the details of how this algorithm works, the algorithm is arrived at but this algorithm works. So here that is start with the j which is n by 2 always it is and then I start from i equal to 1 and reverse i and j .

So the second point the first point is 0, we do not need to change anything that is f of 0 is same is f_k equal to 0. So and then the next one is instead of 1 it comes as n by 2. So now in our 8 point fourier transform we got instead of one we got 4.

So if it is a 16 point instead of one you would get 8. So we need to reverse that it is done here i and j are interchanged by storing first the i value into a temporary array for both real and imaginary part and then we would flip between the i and the j values and then we would go again. We will decrease k by n by 2 and if k we will run a loop till k is less than j plus 1.

So remember we said n was j was 4, now k is 4 here. So k is not k is equal to k is less than j plus 1 j plus 1 is 5, so we go here and change j to j minus k and k to k plus half etcetera. Okay so now you do this and whenever you come back into this loop here and whenever i is less than j you flip this thing. Okay so there are three four loops in this case one is start from j equal to n by 2 and then we go from i from 1 to n minus 2 and if i is less than j we make the flip and then we make it k equal to n by 2 and then say that you run find a new j and k value such that k is less than j plus 1.

So by doing j equal to j minus k and k equal to k by 2. So you have to go through this loop till k is less than j plus 1 and then say j equal to j plus k and go back here again. So in the first case j was 4, i is 1 if you flipped j and 4 and 4 it came here now k is 4. So k is less than j plus 1 so we went here, so now j is 4 and k is four so j becomes 0 and k

became 2. So k is not less than j so we so k is 2 here and j is 0, so we go back from here so j is put equal to 2 because k is 2 and j is 0.

So we go back here and now i is 1 but then and the j is 2. So that is what we will interchange now, so we keep doing this till we get everything reversed. So remember to conclude here it is start with j equal to n by 2 and run a loop from one to n minus 2, i less than n minus 2 in steps of one and reverse and interchange i and j when i is less than j and then the next loop is starting from k equal to n minus 2 and then do this loop where j equal to j minus k and k equal to k by 2 till k is less than j plus 1 and then we come out put j equal to j plus k go back and check i is less than j or not if it is again interchange between i and j that is the loop.

So what we get from this is the following if you do this and if I plot the Fourier transform, so if I plot the Fourier transform what we get is f , ft data. So that is the Fourier transform we get, so now it comes down like this here and then goes up again here. So you see this, so it comes down and it goes up again as you number goes we know that our real function was a gaussian.

So now what we are getting here we know the fourier transform, of the gaussian should be a gaussian but that is not what we get we get it goes up like this and that is because the way f , ft works to get the correct fourier transform we have to cut it at the middle that is at the fourth point here and then paste it on the other side. So finally what we have to do is that the fourier transform comes in this array this is actually folded.

So if you have 8 points in the fourier transform and after doing the bit reversal it comes from 0, 1, 2, 3, 4 correctly and then the other part we have to cut and then paste in the other side. So that is if I can do that here, so it comes like this. So it comes so this part it comes correct up to this and then it is going up again here. So actually we have to make a cut here and take this part and paste it on this side. So with that we can conclude the fourier transform part of this course.