

Time Dependent Quantum Chemistry
Professor Atanu Bhattacharya
Department of Inorganic and Physical Chemistry
Indian Institute of Science, Bengaluru
Module 03 Lecture 22
X-grid to k-grid

Welcome back to Python tutorial 3 of the course Time dependent Quantum Chemistry. So, far we have presented how to, what is the basic theory behind this construction of the k-grid and this is done based on Discrete Fourier transform, we are not getting into the details of those theories, we are just using the final solutions and moving forward for the numerical implementation.

(Refer Slide Time: 00:58)

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

Correspondence FFT (algorithm) based on DFT

$\xi = \frac{1}{N\Delta x} f$

 $f \rightarrow$ list (array) of spatial freq. components

$\alpha_i = [0, 1, 2, \dots, (\frac{N}{2}-1), (\frac{N}{2}), (\frac{N}{2}+1), \dots, -1]$ if N even

$= [0, 1, 2, \dots, (\frac{N-1}{2}), \frac{-(N-1)}{2}, (\frac{-(N-1)}{2}+1), \dots, -1]$ if N odd

$k = 2\pi\xi$

$\Delta k \Delta x = \frac{2\pi}{N}$

Time dependent Quantum Chemistry

So, what we have seen here is that FFT algorithm, whenever I use this FFT algorithm for the construction of the k-grid. When you will be constructing the k-grid, there is a particular restriction based on which the k-grid will be constructed,

$$\Delta k \Delta x = \frac{2\pi}{N}$$

And with this restriction this FFT algorithm will give me a frequency component, a set of frequency components, a list of frequency components.

And this list of frequency components are nothing but $\xi = \frac{1}{N\Delta x} f$, where $[0, 1, 2, \dots]$ these numbers and we see that the number contains 0, positive values and negative values, all these values are there. We will take examples and it will clarify more once we get this frequency

components, we will be able to get this k-grid points by $k = 2\pi\xi$. And the basic idea is that then we get this k-grid points. We can convert this X-grid points to this k-grid points following this procedure.

(Refer Slide Time: 02:10)

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

Correspondence

FFT algorithm through its implementation of DFT converts a finite sequence of equally spaced samples of real space domain into a same length sequence of equally spaced sample of Fourier domain

x-space (DC) x-space

Time dependent Quantum Chemistry

So, what we have understood so far is that the FFT algorithm, FFT algorithm, Fast Fourier Transform algorithm through its implementation of these Discrete Fourier Transform theory converts a finite sequence of equally spaced samples of real space domain. So, often this X position grid or position space is called the real space and the momentum space is called reciprocal space.

So, these are the common terminologies which we use in the Fourier transform problem. So, what you are seeing is that this FFT algorithm is going to convert the equally spaced samples of your space domain into a same length, which means number of elements should be the same, length sequence of equally spaced samples of Fourier domain. So, number of elements would be the same.

And what we have seen, the first element in the list in the k-grid point list what we see, the first element is going to be always 0, which means. What does it mean? It means that if it is 0 spatial frequency, so 0 which means it is related to the spatial frequency, spatial frequency is 0 which means that I do not have at all any oscillation. So, it is more like a DC component, that is why 0 oscillation.

Second element we have seen it is going to be 1, related to 1, so I am writing just spatial frequency components related to 1. So, I should not see this one is a frequency component.

So, this is f0, this is f1, what is f1, we see that it is a, this 1 means. So, this 1, how we are getting this, I will just show you, this is the frequency component 0, 1, 2, like these waves moving on.

So, what does it mean by those numbers that frequency component is that the first frequency component we are sampling is the DC component, which is 0, then second frequency components, which are sampling is frequency 1, so I have only one single frequency, one single oscillation in the entire space, this is the x space. This is again x space.

In the second one, we have seen that it is going to be f2, f2 equals 2. So, I have two complete oscillations, and so on, that is the way it is moving forward. So, this is the meaning of individual frequency components which are getting in the, when you are constructing the k-grid.

(Refer Slide Time: 06:15)

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

Example 1: X-Grid [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] 10 elements
even
N=10 ✓
Δx = 1 ✓

$$\xi = \frac{1}{N\Delta x} f \quad (\text{for even})$$

$$= \frac{1}{10 \times 1} \left[0, 1, 2, \dots, \left(\frac{10}{2}-1\right), \left(-\frac{10}{2}\right), \left(-\frac{10}{2}+1\right), \dots, -1 \right]$$

$$= 0.1 [0, 1, 2, 3, 4, -5, -4, -3, -2, -1] \quad (10)$$

$$\approx 6.286 \times [0, 0.1, 0.2, 0.3, 0.4, -0.5, -0.4, -0.3, -0.2, -0.1]$$

k-Grid $k = 2\pi\xi$

$$= 2\pi [0, 0.1, 0.2, 0.3, 0.4, -0.5, -0.4, -0.3, -0.2, -0.1]$$

$$\approx [0, 0.6286, 1.2571, \dots, -0.6286]$$

We will take an example. And this example will clarify many of our doubts, and it will clarify the procedure further. So, as I mentioned before, to construct the reciprocal space or Fourier space, we need to start with the X-Grid. So, let us say I have an X-Grid first, which starts with the 0, this is just an example, and ends at 9. So, it has [1, 2, 3, 4, 5, 6, 7, 8, 9]. So, this is the grid [0, 1, 2, 3, 4, 5, 6, 7, 8, 9].

I consider 9, after 9, because then I have 10 elements, so the window length is 10. So, N equals now 10 for this problem. And N equals 10, it means that it is the even number. So, I have shown that for even number, this f would be different for odd number, f could be

different. So, if it is even number, and then delta x, what we get is, is 1 in this x space. So, these are the characteristic of the x space, X-Grid.

First, we have to do is that in this entire procedure, first we have to do is that this FFT algorithm will give me this frequency components and this frequency components is going to be for even number, so that is why I will be able to write down as 1 by N is known now, N is known, delta x is known, I will write down this way, and then f components would be 0, 1, 2, like this. And then in $N/2$, $N/2 - 1$, then is going to be $-N/2$, then minus $N/2 + 1$ up to -1 as the we get.

So, we know that N equals 10. So, it is going to be 10 by 2. This is going to be 10. This is going to be 10. And N is here 10. And Δx is going to be 1. So, for the given problem, I have this situation where it is going to be then 0.1 multiplied by 0, 1, 2, 3. Now this part is going to be 5 minus 1 which is 4, that is all. Then I have these negative values minus 5, then minus 4, minus 3, minus 2, minus 1, up to minus 1 we have to go minus 1.

So, these are the frequency components we get after the FFT, Fast Fourier Transform, what we see here is that we have 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Again I have these 10 windows, the window remains to be the same. Here also I had 10 grid points in the x space. Now, I have frequency components also, 10x space. So, finally what I get is that I get 0, then this is the least what I get frequency component 0.1, then 0.2, then 0.3, then 0.4, then minus 0.5, then minus 0.4, then minus 0.3, then minus 0.2, then minus 0.1.

This is what we get. So, once we get this ξ , that is the spatial frequency components, once we get that, then k-grid points can be very easily obtained. Same k-grid points, will be able to obtain, this is the by multiplying by 2π , which means that 2π multiplied by this 0, 0.1, 0.2, 0.3, 0.4, minus 0.5, minus 0.4, minus 0.3, minus 0.2, minus 0.1.

So, if we multiply all and we get 0, and π , we have to consider the pi value, 2π is going to be 2 point, 2π is going to be 6.286. So, if we multiply, approximately, so if we multiply, then we get this 0.6286, then 1.2571. And one can try this final 1 is going to be minus 0.6286. So, this is what we get.

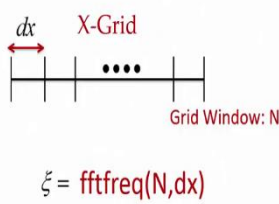
(Refer Slide Time: 11:51)

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

Example 1: Python Implementation

x-Grid
 $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$

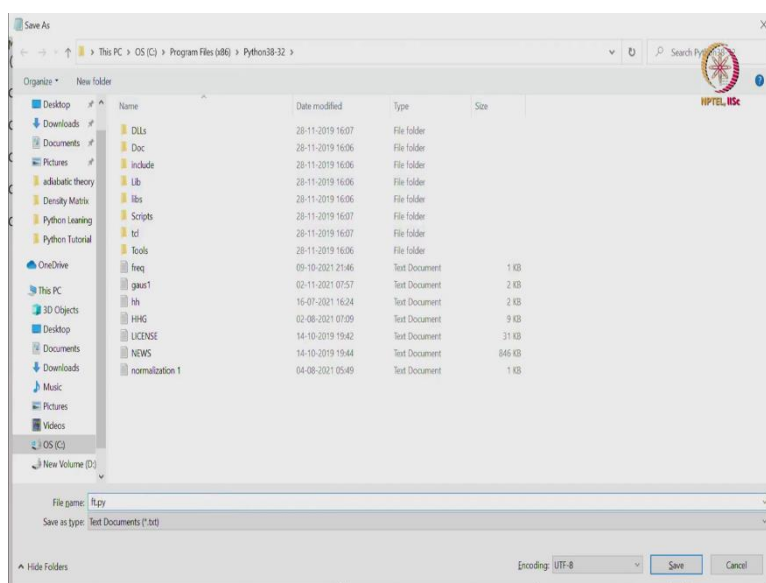


```
#Importing the Required Libraries
from scipy import arange
from scipy.fftpack import fftfreq
#Creating the x-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)
#Creating Fourier Grid (or k-grid)
N=len(x)
xi=fftfreq(N,dx)
print(xi)
```

Time dependent Quantum Chemistry

And we will see the implementation, Python implementation and we will confirm that this is the list what we create. So, we will go back to the laptop first, we will try to write down the program.

(Refer Slide Time: 12:06)



The screenshot shows a Windows File Explorer window with the following details:

- Path: This PC > OS (C:) > Program Files (x86) > Python38-32 >
- Left sidebar: Shows various folders like Desktop, Downloads, Documents, etc.
- Main pane: A table listing files and folders. The 'Tools' folder is selected.
- Table columns: Name, Date modified, Type, Size.
- Table content:

Name	Date modified	Type	Size
DLLs	28-11-2019 16:07	File folder	
Doc	28-11-2019 16:06	File folder	
include	28-11-2019 16:06	File folder	
Lib	28-11-2019 16:06	File folder	
libs	28-11-2019 16:06	File folder	
Scripts	28-11-2019 16:07	File folder	
tdt	28-11-2019 16:07	File folder	
Tools	28-11-2019 16:06	File folder	
freq	09-10-2021 21:46	Text Document	1 KB
gas1	02-11-2021 07:57	Text Document	2 KB
hh	16-07-2021 16:24	Text Document	2 KB
HHG	02-08-2021 07:39	Text Document	9 KB
LICENSE	14-10-2019 19:42	Text Document	31 KB
NEWS	14-10-2019 19:44	Text Document	846 KB
normalization 1	04-08-2021 05:49	Text Document	1 KB

Save As dialog box details:

- File name: f.py
- Save as type: Text Documents (*.txt)
- Encoding: UTF-8
- Buttons: Save, Cancel

```

Command Prompt
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Atanu>cd ..

C:\Users>cd ..

C:\>cd "Program Files (x86)"

C:\Program Files (x86)>cd Python38-32

C:\Program Files (x86)\Python38-32>python ft.py
[0 1 2 3 4 5 6 7 8 9]

C:\Program Files (x86)\Python38-32>

Notepad
File Edit Format View Help
#Importing the Required Libraries
from scipy import arange

#Creating the X-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)

```

First, we will import the required libraries. Importing the required libraries from, I will save it. I will save in the correct directory as ft.py, .py extension needs to be given for the Python programming. So, from SciPy import arange so that we need to first import then we are now creating the X-grid. We know that how to create that xminimum is going to be 0, xmaximum is now going to wait 10 and dx is going to be 1.

So, then x equals, I can write down arange, I have to use this xminimum, arange functionality does not include the stop which is x max 0.in the sequence. And that is why we are stopping at 10. If we are stopping at 10 with an interval, I will print it so that I can, I know what I am doing. So, if I run the program what I get is that [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]. That is the grid points. I come back to the slide now.

(Refer Slide Time: 14:00)

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

Example 1: Python Implementation

x-Grid
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] ✓

```

#Importing the Required Libraries
from scipy import arange
from scipy.fftpack import fftfreq

#Creating the x-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)

#Creating Fourier Grid (or k-grid)
N=len(x)
xi=fftfreq(N,dx)
print(xi)

```

length of an array
len(x)

$\Delta k \Delta x = \frac{2\pi}{N}$

Time dependent Quantum Chemistry

So, I have this X-Grid points, which is presented [1, 2, 3, 4, 5, 6, 7, 8, 9]. And this part is now representing the X-Grid with the help of importing arange functionality from SciPy. This is we are pretty clear to; this part is clear to us. And the grid window, we call it grid window is number of elements we have in the grid points, we have 10 grid points in the X-Grid. So, once we have done that, first thing we will do is that we have to create this frequency, frequency component xi frequency components.

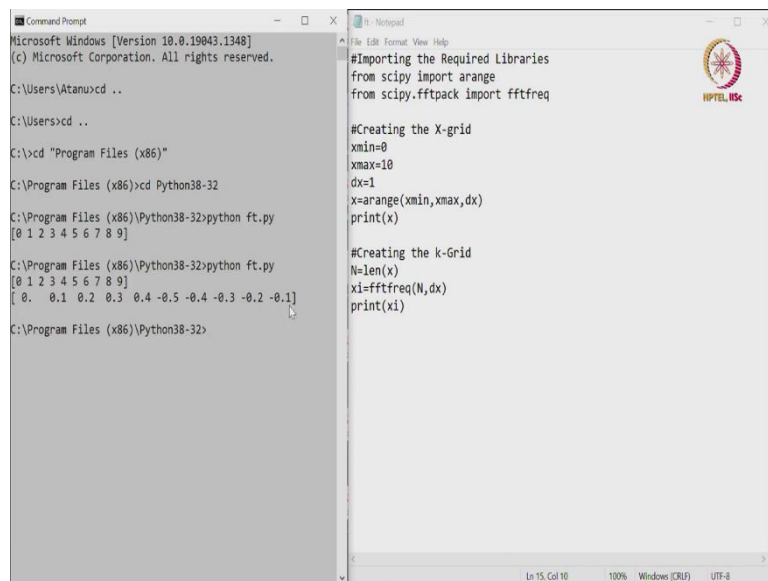
And one can create the Zia frequency components with the help of this fftfreq functionality, fftfreq functionality requires two inputs. The first input is that, what would be the grid window. And second input it records is that what was the delta x, it records delta x information because it will automatically take care of the entire algorithm for us to maintain this $\Delta k \Delta x = 2\pi / N$ restriction in the Fourier transform of the grid.

So, it will take care for us, we do not need to worry about anything, we have to use this fftfreq. So, fftfreq, I need two inputs, the first input is the, what is the window size that we can get it, by selecting this N equals length of X. So, this len functionality gives me the length of an array, length of an array. Previously, we have also used this functionality before, this is python's built-in functionality.

And I have to give the name of the array, that array name, here array name is x, this is the array, so I have to put x here. So, immediately, it will give me how many elements I have. And that is going to be 10, we know that we have 10 elements in this X-Grid, and another input it requires, the second input is going to be dx. What is the difference between the spacing between the adjacent X-Grid points?

So, these are the two information I need. Once I get that, then xi will be immediately created and the Zia will be created. And we can print it and we can check what is going on. So, this fftfreq this functionality is not available with Python, it has, it is available with SciPy.fftpack sub module of the SciPy library. So, first, so we will go back to the laptop.

(Refer Slide Time: 17:15)



```
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Atanu>cd ..

C:\Users>cd ..

C:\>cd "Program Files (x86)"

C:\Program Files (x86)>cd Python38-32

C:\Program Files (x86)\Python38-32>python ft.py
[0 1 2 3 4 5 6 7 8 9]

C:\Program Files (x86)\Python38-32>python ft.py
[0.  0.1  0.2  0.3  0.4 -0.5 -0.4 -0.3 -0.2 -0.1]

C:\Program Files (x86)\Python38-32>
```

```
#Importing the Required Libraries
from scipy import arange
from scipy.fftpack import fftfreq

#Creating the X-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)

#Creating the k-Grid
N=len(x)
xi=fftfreq(N,dx)
print(xi)
```

In order to use this `fftfreq`, I need to use the `SciPy.fftpack`. So, I will import the record functionality from `SciPy.fftpack import fftfreq`. So, I am going to import this one, the moment I import it, I will be able to now create the k-grid. So, I will create, creating the k-grid. When I am creating the k-grid, the first information first input I need is the, so I will be able to create the k-grid as I named this entire array, the k-grid array `xi` which is going to be now `fftfreq` within bracket `N` comma `dx`, `dx` is already given here, `dx` is going to be 1, so we know that `dx`.

But `N` is not given, `N` is the length of the X-Grid. That information I have to because the same length will be maintained. This is the only two inputs I need to create the frequency components `xi`, frequency components of the associated with the k-grid. And then I can print it to check what I am calculating here. I can print and then I can run the program, what I see is that the frequency components are following 0.1, 0.2, 0.3, 0.4, then minus 0.5, minus 0.4, minus 0.3, minus 0.2, and minus 0.1. That is exactly, we will go back to the slide now.

(Refer Slide Time: 19:17)

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

10 elements

even
N=10
Δx = 1

Example 1: X-Grid [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

$$\xi = \frac{1}{N\Delta x} f \quad (\text{for even})$$

$$= \frac{1}{10 \times 1} \left[0, 1, 2, \dots, \left(\frac{10}{2}-1\right), \left(\frac{10}{2}\right), \left(-\frac{10}{2}+1\right), \dots, -1 \right]$$

$$= 0.1 [0, 1, 2, 3, 4, -5, -4, -3, -2, -1]$$

2π
≈ 6.2832

$$= [0, 0.1, 0.2, 0.3, 0.4, -0.5, -0.4, -0.3, -0.2, -0.1]$$

k-Grid $k = 2\pi\xi$

$$= 2\pi [0, 0.1, 0.2, 0.3, 0.4, -0.5, -0.4, -0.3, -0.2, -0.1]$$

$$= [0, 0.6283, 1.2566, \dots, -0.6283]$$

Time dependent Quantum Chemistry

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

length of an array
 $\Delta k \Delta x = \frac{2\pi}{N}$
len(x)

Example 1: Python Implementation

x-Grid
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```

#Importing the Required Libraries
from scipy import arange
from scipy.fftpack import fftfreq

#Creating the x-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)

#Creating Fourier Grid (or k-grid)
N=len(x)
xi=fftfreq(N,dx)
print(xi)
    
```

Time dependent Quantum Chemistry

That is exactly what we have calculated here. The frequency component xi is given by 0 plus 0.1, 0.2, 0.3 like this. And once we know the xi, then one can calculate the k points very easily and we will show how to calculate the k points.

(Refer Slide Time: 19:39)

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

Example 1: Python Implementation

$x \rightarrow [0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9]$
 $y \rightarrow [0. \ 0.1 \ 0.2 \ 0.3 \ 0.4 \ -0.5 \ -0.4 \ -0.3 \ -0.2 \ -0.1]$
 $k \rightarrow 2\pi q$

```
#Importing the Required Libraries
from scipy import arange
from scipy.fftpack import fftfreq
#Creating the x-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)
#Creating Fourier Grid (or k-grid)
N=len(x)
xi=fftfreq(N,dx)
print(xi)
```

Time dependent Quantum Chemistry

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

Example 1: Python Implementation

$[0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9]$
 $[0 \ 0.62831853 \ 1.25663706 \ 1.88495559 \ 2.51327412 \ -3.14159265 \ -2.51327412 \ -1.88495559 \ -1.25663706 \ -0.62831853]$

```
#Importing the Required Libraries
from scipy import arange,pi
from scipy.fftpack import fftfreq
#Creating the x-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)
#Creating Fourier Grid (or k-grid)
N=len(x)
xi=fftfreq(N,dx)
k=2*pi*fftfreq(N,dx)
print(k)
```

Time dependent Quantum Chemistry

So, this is your x, this is the X-Grid, this is the X-Grid we have then we have xi components we have. And note that the same number of, the window length is the same. And now we have to find out the k, and k is nothing but 2π into xi. So, what we need to do is that, all we have to do is that we have to now just multiply this entire xi array by 2π , then we get this k values. And that is exactly what we are going to do right now. We are going to now move to the laptop and in the program.

(Refer Slide Time: 20:38)

```

Command Prompt
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Atanu>cd ..
C:\Users>cd ..
C:\>cd "Program Files (x86)"
C:\Program Files (x86)>cd Python38-32
C:\Program Files (x86)\Python38-32>python ft.py
[0 1 2 3 4 5 6 7 8 9]
C:\Program Files (x86)\Python38-32>python ft.py
[0 1 2 3 4 5 6 7 8 9]
[ 0.  0.1  0.2  0.3  0.4 -0.5 -0.4 -0.3 -0.2 -0.1]
C:\Program Files (x86)\Python38-32>python ft.py
[0 1 2 3 4 5 6 7 8 9]
[ 0.  0.62831853  1.25663706  1.88495559  2.51327412
 12 -3.14159265
-2.51327412 -1.88495559 -1.25663706 -0.62831853]
C:\Program Files (x86)\Python38-32>

ft - Notepad
#Importing the Required Libraries
from scipy import arange,pi
from scipy.fftpack import fftfreq

#Creating the X-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)

#Creating the k-Grid
N=len(x)
xi=fftfreq(N,dx)
k=2*pi*xi
print(xi)
print(k)
  
```

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

Example 1: Python Implementation

$\Delta x = 1$

$x\text{-grid}$ [0 1 2 3 4 5 6 7 8 9]

$k\text{-grid}$ [0 0.62831853 1.25663706 1.88495559 2.51327412 -3.14159265 -2.51327412 -1.88495559 -1.25663706 -0.62831853]

$\Delta k \approx 0.62832$

$N = 10$

$\Delta k \Delta x = \frac{2\pi}{N}$

```

#Importing the Required Libraries
from scipy import arange,pi
from scipy.fftpack import fftfreq
#Creating the x-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)
#Creating Fourier Grid (or k-grid)
N=len(x)
22*pi*fftfreq(N,dx)
print(k)
  
```

Time dependent Quantum Chemistry

So, we will write down this k is going to be now, let us keep this xi, then we will just write down k equals 2 multiplied by pi, but remember pi is not available with Python directly not an inbuilt functionality or inbuilt mathematical function that needs to be imported from SciPy. Built in arithmetic computation, which can be done with Python are just addition subtraction, multiplication, division and exponentiation.


But remaining part cos, trigonometric, pi those mathematical function needs to be imported from SciPy or NumPy. NumPy we are not at all using here we are consistently using only SciPy. So, k equals 2 pi multiplied by this xi and we can also print k. So, finally, what we get is we see that the first will go back to the, we have this k value, the first k value is 0, second k value is 0.62. So, the k-grid has been prepared.

So, this is the k-grid we have prepared and this was the x-grid. So, x-grid and k-grid has a correspondence, what is the correspondence we have this is x-grid, let us say. So, it is starting from 0 and then the difference is 1 in the X-Grid, this is the X-Grid, but if we look at the k-grid it is starting from 0, then it has positive values and negative values and the difference, so this is the k-grid and the difference the spacing is delta k. So, this is delta x equals to 1 and this is delta k equals 0.62832 approximately.

So, if we see that the FFT algorithm, this FFT frequency algorithm coming from fftpack of SciPy, it is actually taking care of this requirement delta k delta x is going to be $2\pi/N$. And that is exactly what we get, if we multiply this and 1, we get $2\pi/N$, N equals, we know it is going to be 10. So, it is taking care of that conversion, the requirement for the conversion. So, this is what we have to do, it is very simple, we can construct the reciprocal grid with the help of this script.

(Refer Slide Time: 24:03)

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid $\frac{1}{\Delta x}$ k-grid 

Example 2: X-Grid $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$ $N=11$

$$\xi = \frac{1}{N\Delta x} f = \frac{1}{N\Delta x} \left[0, 1, 2, \dots, \frac{(11-1)}{2}, \left(-\frac{(11-1)}{2}\right), \left(-\frac{(11-1)}{2}+1\right), \dots, -1 \right]$$

$$= 0.091 [0, 1, 2, 3, 4, 5, -5, -4, -3, -2, -1]$$

$$= [0, 0.091, 0.182, 0.273, \dots, -0.091]$$

\uparrow

k-Grid $k = 2\pi\xi$

$$= 2\pi [0, 0.091, 0.182, \dots, -0.091]$$

$$= [0, 0.572, 1.144, \dots, -0.572]$$

$\frac{2\pi}{6.283}$

Time dependent Quantum Chemistry



Construction of Fourier Grid/Reciprocal Grid/k-Grid

Example 1: Python Implementation

$\Delta x = 1/10$

x-grid: [0 1 2 3 4 5 6 7 8 9]

k-grid: [0 0.62831853 1.25663706 1.88495559 2.51327412 -3.14159265 -2.51327412 -1.88495559 -1.25663706 -0.62831853]

$\Delta k \approx 0.62832$

$\Delta k \Delta x = \frac{2\pi}{N}$

$N = 60$

```
#Importing the Required Libraries
from scipy import arange, pi
from scipy.fftpack import fftfreq
#Creating the x-grid
xmin=0
xmax=10
dx=1
x=arange(xmin,xmax,dx)
print(x)
#Creating Fourier Grid (or k-grid)
N=len(x)
k=2*pi*fftfreq(N,dx)
print(k)
```

Time dependent Quantum Chemistry

We will take another example, the example for the odd number of odd window where we have odd number of elements. Odd number of elements we have [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. So, we have 11 number of elements now, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. So, starting from, so this is your x-grid now [0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10]. So, total number is 11, it is odd, because it is odd, then this f function has slightly different, f is going to be now a list for the odd number of N and that list is given by $N \Delta x$.

Now, it is going to be 0, 1, 2 then N minus 1 by 2, then minus N minus 1 by 2, then minus N minus 1 by 2 plus 1 continue up to minus 1. So, we know that N is now 11 multiplied by delta x is still 1. So, we have 1, then I have this N value is going to be now 11, this N value is 11, this N value is 11. So, what I have is 0, 1, 2, 3, this part is going to be 5, so 4, then 5, then I have minus 5, then I have minus 4, minus 3, minus 2, minus 1, we have 11 number of frequency components.

And I have to now, multiply this 1 by 11 is going to be approximately 0.091. So, I have to multiply. So, in the N this frequency component xi we get 0 then 0.091, then 0.182, then 0.273 like this, and it will end at minus 0.091. We have both 0 and positive and negative frequency components. So, once we get the xi, we will be able to create the k-grid points by just multiplying this the list of xi frequency components which is 0.091, 0.182, then so on up to 0.091.

And if we multiply finally what we get 2π equals 6.286, 6.286 that is why we get a list of k-grid points or an array of k-grid points 0.572. Note that, in Python programming, when Python prints its array, it does not show the comma, we are showing the comma just to separate visual clarity.

But when Python is printing, as I can show you, Python has printed like this, it does not have any comma, it has only a space between two numbers. This is the convention Python follows. But when we are presenting it for visual clarity, we are giving this common. So, do not get confused by how Python is printing and how we are writing here. This is just for simplified part we are following. So, this second part is going to 1.144 like this, and it is going up to minus 0.572. So, plus 0.572 minus 0.572. Both are present. So, this is the x-grid points.

(Refer Slide Time: 28:32)

```

Command Prompt
C:\Users\Atanu>cd ..
C:\Users>cd ..
C:\>cd "Program Files (x86)"
C:\Program Files (x86)>cd Python38-32
C:\Program Files (x86)\Python38-32>python ft.py
[0 1 2 3 4 5 6 7 8 9]
C:\Program Files (x86)\Python38-32>python ft.py
[0 0.1 0.2 0.3 0.4 -0.5 -0.4 -0.3 -0.2 -0.1]
C:\Program Files (x86)\Python38-32>python ft.py
[0 1 2 3 4 5 6 7 8 9]
[0 0.1 0.2 0.3 0.4 -0.5 -0.4 -0.3 -0.2 -0.1]
[0 0.62831853 1.25663706 1.88495559 2.51327412 -3.14159265 -2.51327412 -1.88495559 -1.25663706 -0.62831853]
C:\Program Files (x86)\Python38-32>python ft.py
[0 1 2 3 4 5 6 7 8 9 10]
[0 0.09090909 0.18181818 0.27272727 0.36363636 0.45454545 -0.45454545 -0.36363636 -0.27272727 -0.18181818 -0.09090909]
[0 0.57119866 1.14239733 1.71359599 2.28479466 2.85599332 -2.85599332 -2.28479466 -1.71359599 -1.14239733 -0.57119866]

Notepad
#Importing the Required Libraries
from scipy import arange,pi
from scipy.fftpack import fftfreq

#Creating the X-grid
xmin=0
xmax=11
dx=1
x=arange(xmin,xmax,dx)
print(x)

#Creating the k-Grid
N=len(x)
xi=fftfreq(N,dx)
k=2*pi*xi
print(xi)
print(k)
  
```

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

$\Delta k \Delta x = \frac{2\pi}{N}$

Example 2: Python Implementation

$\Delta x = 1$

x-grid

```

#Importing the Required Libraries
from scipy import arange,pi
from scipy.fftpack import fftfreq
#Creating the x-grid
xmin=0
xmax=11
dx=1
x=arange(xmin,xmax,dx)
print(x)
#Creating Fourier Grid (or k-grid)
N=len(x)
xi=fftfreq(N,dx)
print(xi)
k=2*pi*xi
print(k)
  
```

$\Delta k = 0.57119$

k-grid

We will go over now. And this time we will be able to modify will change to laptop, and we will be able to modify the program accordingly. So, now we have xmax is going to be now 11. And we know that arrange functionality does not include in the sequence this maximum,

the stop point. So, if we use xmax equals 11, and it will stop at 10. And that is exactly what we want. And then we will be creating the X-grid points and remaining part is the same.

So, we will just run the program, if we run the program, we see that we have created 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. So, this is the X-grid point, we have created. Second one second list, representing 0, then 0.09, then 0.18, there is the list of the xi, the possible frequency components, once we get the possible frequency components, then we get the k-grid points as 0, 0.57, 1.14. So, we will go back to the slide right now.

And we see that we have been able to, so this is your x-grid point where it is starting from 0 then this is 1 and so on. And this is your k-grid, where it starting at 0 then 0.57119 approximately and it has also negative of that value also, the same value. So, we see again the delta k is given by 0.57119 And here delta x is given by 1. So, again, we have this delta k delta x equals 2 pi by N fulfilled and this is taken care by this fftfreq, this entire functionality of scipy.fftpack sub module.

(Refer Slide Time: 31:12)

Python Tutorial 3: Fourier Transform

Construction of Fourier Grid/Reciprocal Grid/k-Grid

$k = 2 \cdot \pi \cdot \text{fftfreq}(N, dx)$
 $\Delta k \Delta x = \frac{2\pi}{N}$

Time dependent Quantum Chemistry

So, what we see is that the Fast Fourier Transform, this Fast Fourier Transform pack, this scipy.fftpack sub module, one can use this sub module to convert the x space to k-grid. So, we are just simply converting the X-grid to k-grid, we have not converted the function yet. Function has not been converted it is just the grid point has been converted then function will be represented on those grid points with the help of a certain procedure.

So, what we have seen is that this fftfreq functionality of scipy.fftpack can actually convert X-grid to k-grid and is taking care of all the necessary requirement imposed by Discrete

Fourier Transform, DFT. Discrete Fourier Transform has certain requirements, doing the job in the background for me for us and is giving me the x-grid points.

So, this is the final equation to get the k-grid points and as an input what I needed an input, input is taken from the nature of decay grid point. The first nature is the N, which is the window length and dx is the spacing in this X-grid point. So, once we know this to include inputs will be able to construct k-grid points by with the help of this simple functionality, fftfreq functionality.

(Refer Slide Time: 33:11)

Python Tutorial 3: Fourier Transform

abs()

```

#Importing the Required Libraries
from scipy import arange, pi, cos
from scipy.fftpack import fftfreq, fft
from matplotlib.pyplot import plot, show, xlim

#Creating the x-grid
xmin=-100
xmax=100
dx=0.001
x=arange(xmin, xmax, dx)

#Defining a Cosine Function in Position Space Grid
y=cos(20*x)

#Creating Fourier Grid (or k-grid)
N=len(x)
k=2*pi*fftfreq(N, dx)

#Fourier Transforming the Cosine Function
y_k=fft(y)
plot(k, abs(y_k))
xlim(-30, 30)
show()
                
```

Python Implementation
through $\text{fft}(Y)$ of $\frac{?}{k}$
scipy.fftpack

function
on the x-grid
of scipy.fftpack

$\psi(k, t) \approx \int \psi(x, t) e^{-ikx} dx$

Complex $(a+ib)$
 $i = \sqrt{-1}$

$\sqrt{a^2 + b^2}$

Time dependent Quantum Chemistry

Will move on. And as I have previously pointed out that there are two steps in the Fourier transform, first step is that we have to get the k-grid, we have to construct the k-grid and then we have to convert the function. So, this program shows that, this is an example where we are going to now transform, Fourier transform a cosine function, a cosine function in the X-axis, if the cosine functions if I take on the X-Grid, it should have this oscillation.

And question is in the k-grid how it should look like, that is the thing we are going to take a look at. So, the first thing is that create the X-Grid, this is what we have done, creating the X-Grid, then we have to create the k-grid, this is the k-grid, we know that we are now familiar with this, then we have to define the function in the X-Grid. And that is the definition of the function you are giving.

We know that in a mathematical function when you use an array, we get back an array as a function values. So, Y is again an array of the function values. So, this is we are defining the

cosine function in the position space grid. Once we have defined a function in the position space grid, then the Fourier transforming the cosine function is very simple, we have to just use `fft` of `Y`. So, `fft` functionality, this is the function in the `X`-Grid, the function on the `X`-Grid, so on the `X` create the function I have defined.

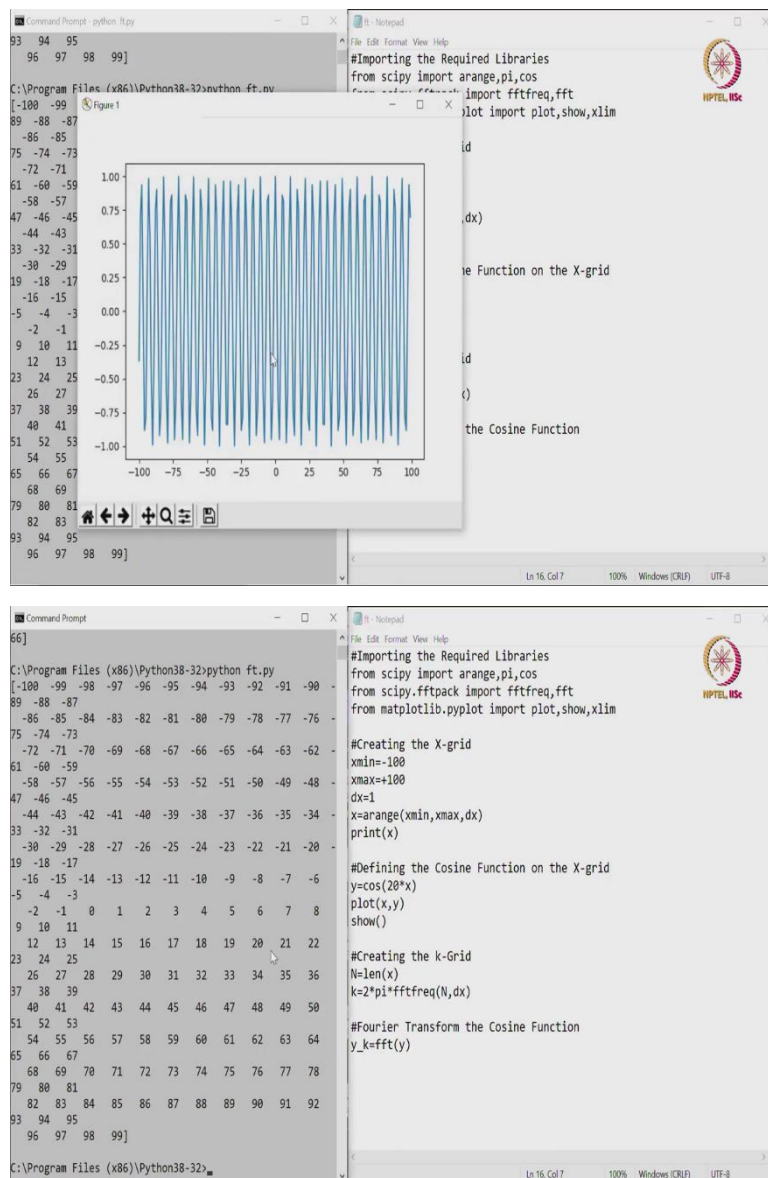
Here, the name of the function is `Y`, that is why we are giving `fft Y`, that is all. So, this is discretized function, and this FFT functionality of the same package `scipy.fftpack` can actually convert give me the discretized function in the `k` domain. So, that is going to be, I am giving the name as `y-k`. So, we will we will take a look at it. One more point which we would like to mention here is that when you do this FFT, the Fast Fourier Transform, Fast Fourier Transform when you are doing it always there is an complex part.

So, we are multiplying. So, basic idea is that if we look at the integration part, although this is based on Discrete Fourier Transform, we are not following this integration part, but integration is the general form. We have a function in the position space domain, we are multiplying by $\exp(-ikx)$. Some complex function will be multiplied always to get the Fourier transform, this is going to be $\phi(k)$.

So, because it is multiplied by a complex function, always this `y k` after the Fourier transforming space domain function to position momentum domain function, this function will be always a complex function. And because it is a complex function, which means that it will have always this form $a + ib$, i is square root of minus 1. So, because it is a complex function, we will try to plot taking its absolute value of the function, which means it will be plotting a square plus b square, square root of a square plus b square.

So, this part we have to remember, after we are transforming a function from position domain to the momentum domain always will get a complex function and that complex function, either we can plot the real and complex parts separately or to get an idea what kind of functions I have in the `k` domain now, it is going to be I can use the absolute value. Absolute value is the absolute magnitude which you can get from `abs` within bracket the function name and that is available in the python's built in library. So, one can use it very quickly. So, we will go back to laptop right now.

(Refer Slide Time: 38:19)



And we will try to implement this what we have learned scipack from scipy.fftpack will import fftfreq from the k-grid construction and also FFT functionality for converting the function. So, these are the two functions we will be importing. Now, here in this SciPy, we need another functionality, sorry, mathematical function, which is cos function because cos function is also not available with the, in the pythons built in library, we have to import it from SciPy. So, you are importing it because we have to define the cos function.

So, once we have now this X-Grid definition remains to be the same. Here, we will just change the values from minus 100 to plus 100, it is just convenience. Depending on what functions we are going to represent, we can create this X-Grid, then we are going to define the cosine function on the X-Grid, we are first defining it.

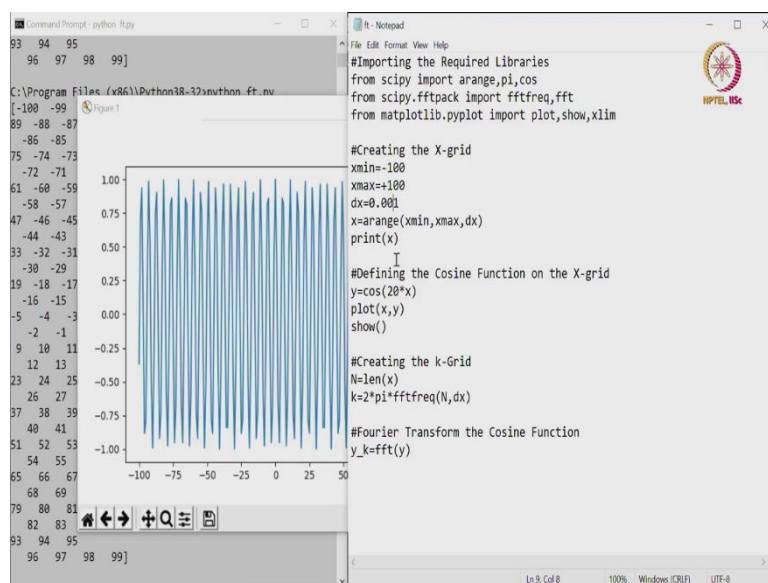
Definition is very simple y equals $\cos x$, the moment we take and then we will multiply by 20, why I will explain. $\cos 20x$, 20 multiplied by x . So, x is an array, which is the X-Grid. So, y is going to be again an array and our next step is to create the k-grid. So, k-grid has been created. I will now we do not need ξ always we need do need to show x_i . We can directly use k equals $2\pi\xi$ which is `fftfreq`. So, we have created the k-grid.

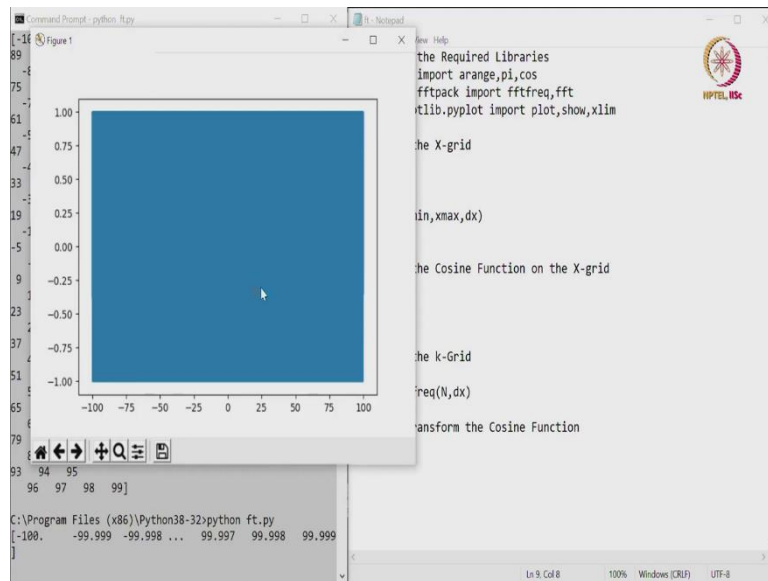
Now, we will do one thing, we will just Fourier transform the function, Fourier transform the cosine function. And the way we are going to Fourier transform is very simple, we are going to give the name y_k equals just `fft` within bracket, the discretized function in the x space, that is y and that is all. This will do the job. Next what we will do instead of printing we will just take a look at the, what we are getting.

So, first we will plot this X function and for plotting, we have to import again because the plot functionality does not come directly from Python in built library, we have to import it from `matplotlib.pyplot` sub module, this is we are now familiar with `matplotlib.pyplot` I have to import plot functionality, then show functionality and then I will make use of `x` limit functionality, I will just control the limit of that x .

So, first we will plot this x versus y . We will see what is going on, we will run the program and we will see what x versus y is giving me. We have to always, whenever we are plotting, we have to show the plot, otherwise it will not show anything. So, now I have this x versus y plot.

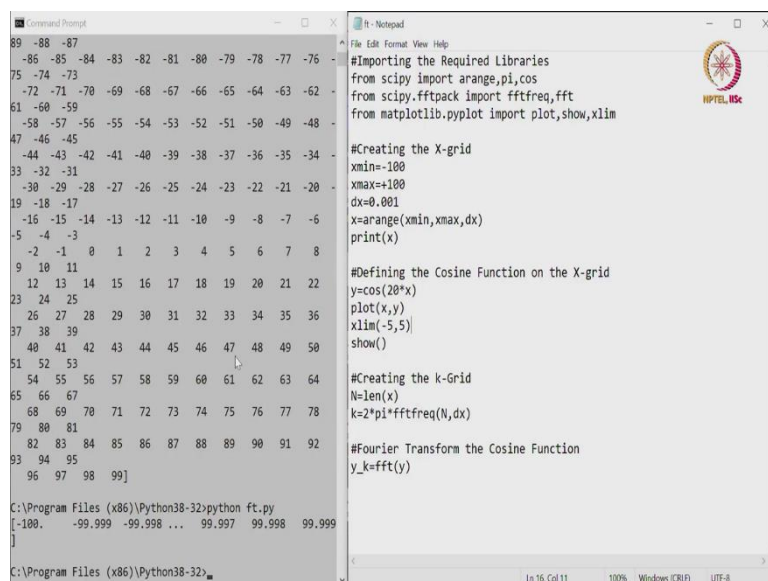
(Refer Slide Time: 42:28)

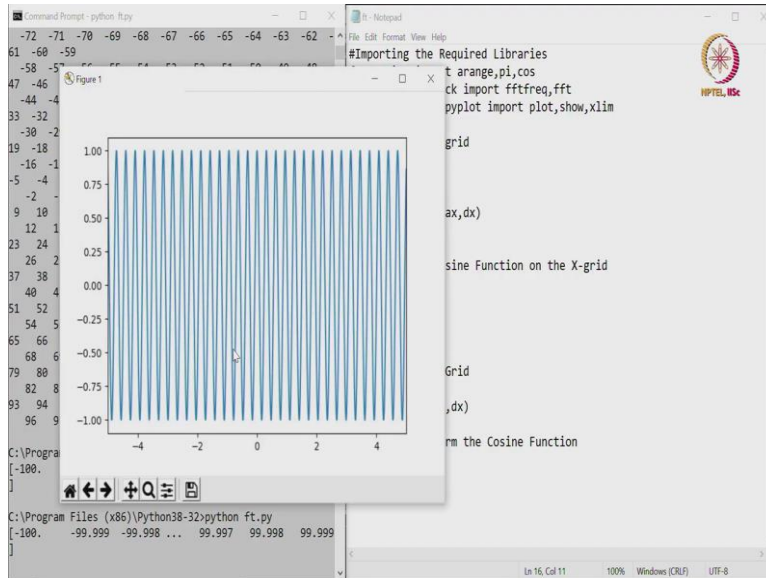




And dx also we have to change here, dx we are going to use 0.001, we will change it because then it will be, otherwise you see there is a slight change in the absolute amplitude. And that is because the sampling problem. So, if you reduce the spacing in the X-Grid, it will give you the values. So, what we see is almost nothing, we have to control the x limit.

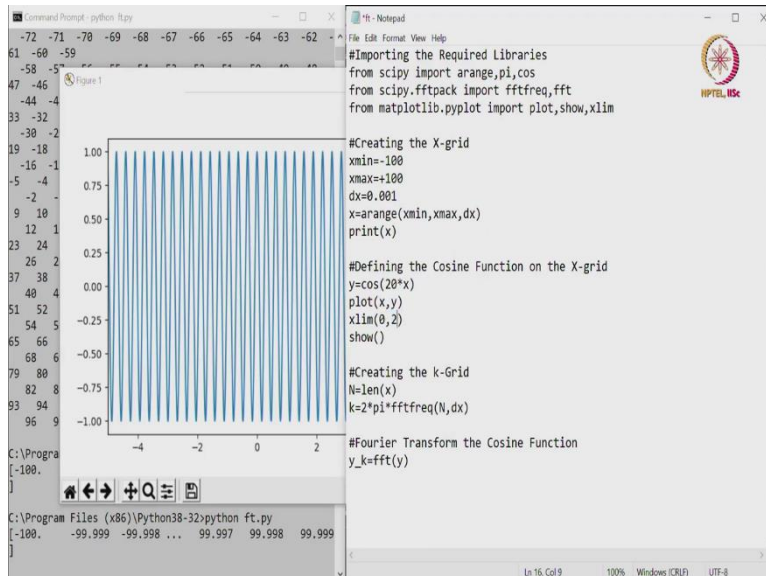
(Refer Slide Time: 42:59)

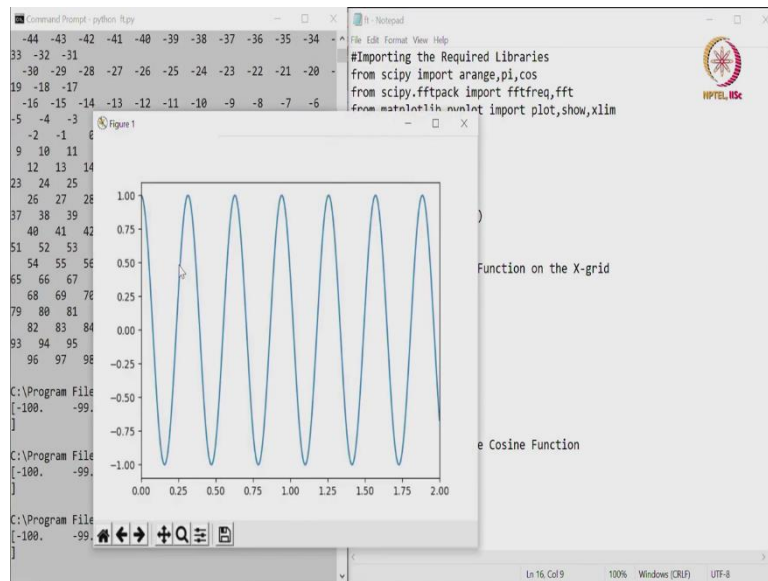




x limit equals, x limit, we place x limit to be minus 5 to plus 5. If we set the x limit, very small region we are trying to look at, then we see that the cosine function has been plotted.

(Refer Slide Time: 43:20)





```

Command Prompt
33 -32 -31
-30 -29 -28 -27 -26 -25 -24 -23 -22 -21 -20
19 -18 -17
-16 -15 -14 -13 -12 -11 -10 -9 -8 -7 -6
-5 -4 -3
-2 -1 0 1 2 3 4 5 6 7 8
9 10 11
12 13 14 15 16 17 18 19 20 21 22
23 24 25
26 27 28 29 30 31 32 33 34 35 36
37 38 39
40 41 42 43 44 45 46 47 48 49 50
51 52 53
54 55 56 57 58 59 60 61 62 63 64
65 66 67
68 69 70 71 72 73 74 75 76 77 78
79 80 81
82 83 84 85 86 87 88 89 90 91 92
93 94 95
96 97 98 99]

C:\Program Files (x86)\Python38-32\python ft.py
[-100. -99.999 -99.998 ... 99.997 99.998 99.999
]

C:\Program Files (x86)\Python38-32\python ft.py
[-100. -99.999 -99.998 ... 99.997 99.998 99.999
]

C:\Program Files (x86)\Python38-32\python ft.py
[-100. -99.999 -99.998 ... 99.997 99.998 99.999
]

C:\Program Files (x86)\Python38-32\
]

#Importing the Required Libraries
from scipy import arange,pi,cos
from scipy.fftpack import fftfreq,fft
from matplotlib.pyplot import plot,show,xlim

#Creating the X-grid
xmin=-100
xmax=100
dx=0.001
x=arange(xmin,xmax,dx)
print(x)

#Defining the Cosine Function on the X-grid
y=cos(20*x)

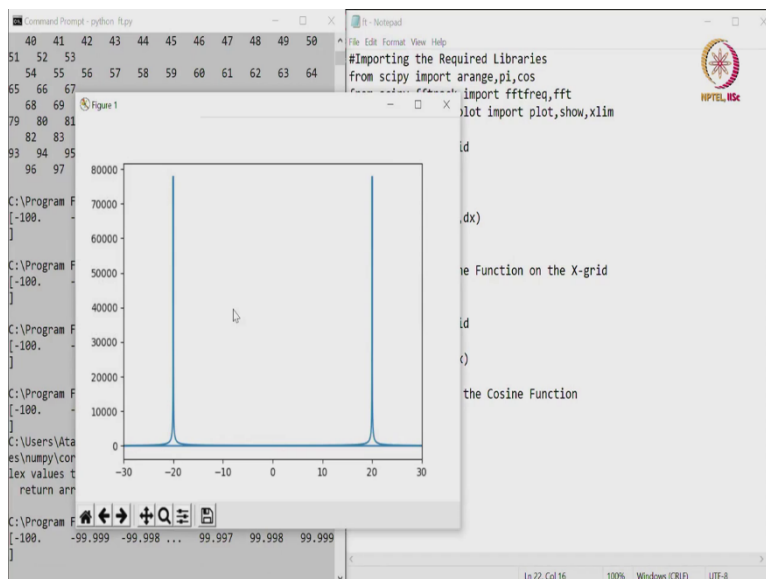
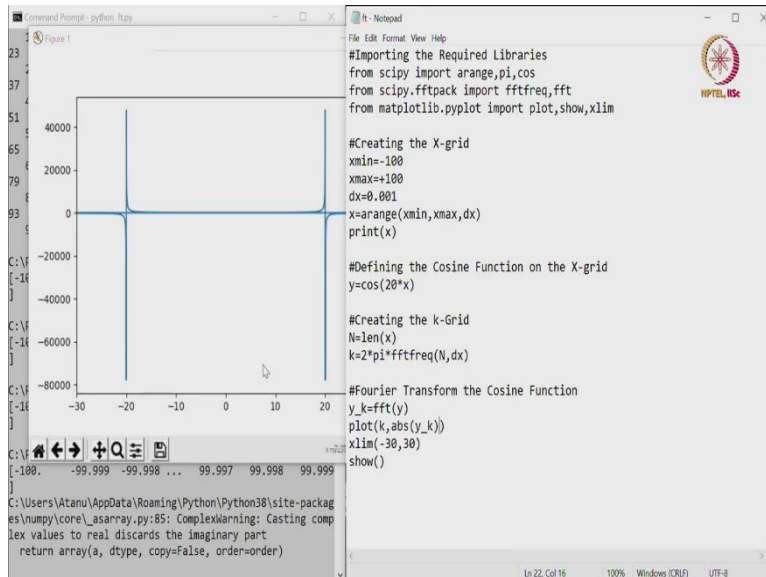
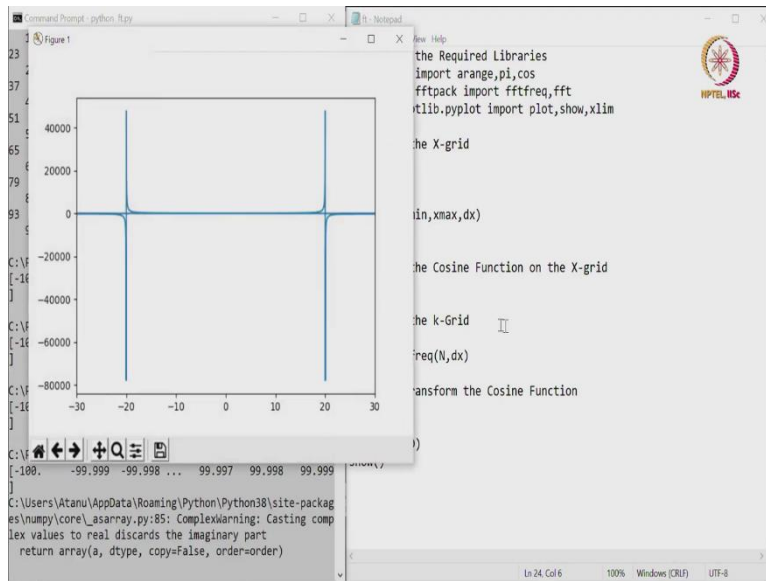
#Creating the k-Grid
N=len(x)
k=2*pi*fftfreq(N,dx)

#Fourier Transform the Cosine Function
y_k=fft(y)
plot(k,y_k)
xlim(-30,30)
show()

```

We can also change the x limit from 0 to, let us say, 2 that will be much clearly shown, then we see that the cosine function has been plotted on the X-Grid, and we have the X-Grid, this function. So, once we have confirmed, this part, I can now delete from here, we do not need it. And we can check after the Fourier transform what we are getting. So, now x is not, x here, it has to be now k, I am going to plot k versus y k. This is what I am going to plot right now. And x limit we will set to be minus 30. This is just for our convenience. So, conveniently one can select this x limits.

(Refer Slide Time: 44:12)



So, if we do that, then what we see here, we cannot plot this x_k because x_k has both real and the complex part. So, as we have mentioned it is going to be absolute value of y_k we are going to plot. So, if we plot the absolute value of y_k , what we see here is that, there are two peaks we see at the 20 position.

(Refer Slide Time: 44:40)

Python Tutorial 3: Fourier Transform

Fourier Transform a Cosine Function

abs()

```
#Importing the Required Libraries
from scipy import arange, pi, cos
from scipy.fftpack import fftfreq, fft
from matplotlib.pyplot import plot, show, xlim

#Creating the x-grid
xmin=-100
xmax=100
dx=0.001
x=arange(xmin, xmax, dx)

#Defining a Cosine Function in Position Space Grid
y=cos(20*x)

#Creating Fourier Grid (or k-grid)
N=len(x)
k=2*pi*fftfreq(N, dx)

#Fourier Transforming the Cosine Function
y_k=fft(y)
plot(k, abs(y_k))
xlim(-30, 30)
show()
```

$\sqrt{a^2+b^2}$

Python Implementation
through $\text{fft}(Y)$ of
`scipy.fftpack`

function ?

fft(Y) on the x-grid of scipy.fftpack

y-k

$\phi(k, t) \approx \int \psi(x, t) e^{-ikx} dx$

Complex (a+ib)
 $i = \sqrt{-1}$

Time dependent Quantum Chemistry

Python Tutorial 3: Fourier Transform

Fourier Transform a Cosine Function

Python Implementation
through $\text{fft}(Y)$ of
`scipy.fftpack`

(a)

(b)

Time dependent Quantum Chemistry

So, I will go back to the slide right now, and we will clarify some of the points here. We will stop here, and we will continue this session in the next class.