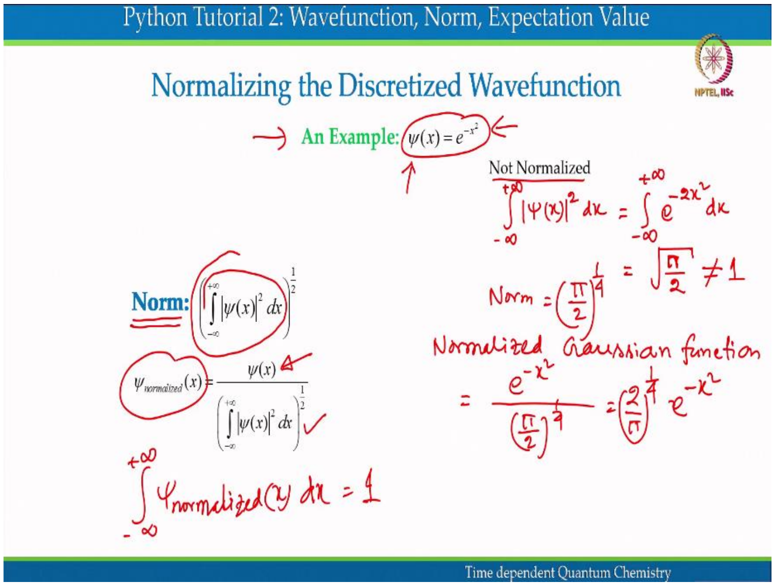
# Time Dependent Quantum Chemistry
## Professor. Atanu Bhattacharya
## Department of Inorganic and Physical Chemistry
## Indian Institute of Science, Bengaluru
## Lecture 15
## Normalizing the Discretized Wavefunction and Finding Expectation Value

Welcome back to python tutorial 2, we have presented how to represent a wave function, how to discretize a wave function or how to represent a wave function on a position grid.

(Refer Slide Time: 0:44)



Python Tutorial 2: Wavefunction, Norm, Expectation Value

Normalizing the Discretized Wavefunction

An Example: $\psi(x) = e^{-x^2}$

Not Normalized

$$\int_{-\infty}^{+\infty} |\psi(x)|^2 dx = \int_{-\infty}^{+\infty} e^{-2x^2} dx = \sqrt{\frac{\pi}{2}} \neq 1$$

$$\text{Norm} = \left(\frac{\pi}{2}\right)^{\frac{1}{4}}$$

Normalized Gaussian function

$$= \frac{e^{-x^2}}{\left(\frac{\pi}{2}\right)^{\frac{1}{4}}} = \left(\frac{2}{\pi}\right)^{\frac{1}{4}} e^{-x^2}$$

Norm: $\left(\int_{-\infty}^{+\infty} |\psi(x)|^2 dx\right)^{\frac{1}{2}}$

$$\psi_{normalized}(x) = \frac{\psi(x)}{\left(\int_{-\infty}^{+\infty} |\psi(x)|^2 dx\right)^{\frac{1}{2}}}$$

$$\int_{-\infty}^{+\infty} \psi_{normalized}(x)\, dx = 1$$

Time dependent Quantum Chemistry

And next what we will do, we will take that discretized wave function and we will normalize it and we will check how to normalize it. The wave function which we have taken, this is the form of the wave function we have taken, clearly it is not a normalized wave function I can prove that and in order to prove that what I need to do is that I have to take this integration,

$$\psi(x) = e^{-x^2}$$

$$\int_{-\infty}^{\infty} |\psi(x)|^2 \, dx = \int_{-\infty}^{\infty} e^{-2x^2} dx$$

$$\int_{-\infty}^{\infty} |\psi(x)|^2 \, dx = \sqrt{\frac{\pi}{2}} \neq 1$$

So we have to take the square of absolute value of this wave function and then take the integration of it. So, normalized, if it is a normalized wave function, then this integration should have given me 1 but because it is not giving me 1 it is not normalized.

So, I have to make it normalized and for the normalization what we will do is we have to find out the norm. Norm is defined by the square root of this entire integration, the square root of this integration

$$Norm = \left( \int_{-\infty}^{\infty} |\psi(x,t)|^2 \, dx \right)^{\frac{1}{2}}$$

and once we get the norm then we divide the wave function non-normalized wave function which is not normalized yet that way function has to be divided by its own norm to get the normalized wave function. And then if I do

$$\psi_{normalized}(x) = \frac{\psi(x)}{\left( \int_{-\infty}^{\infty} |\psi(x,t)|^2 \, dx \right)^{\frac{1}{2}}}$$

$$\int_{-\infty}^{\infty} |\psi_{normalized}(x)|^2 \, dx = 1$$

so that is the procedure for doing normalization.

So, let us look at this for this wave function we can easily find out the norm. Norm is going to be square root of this

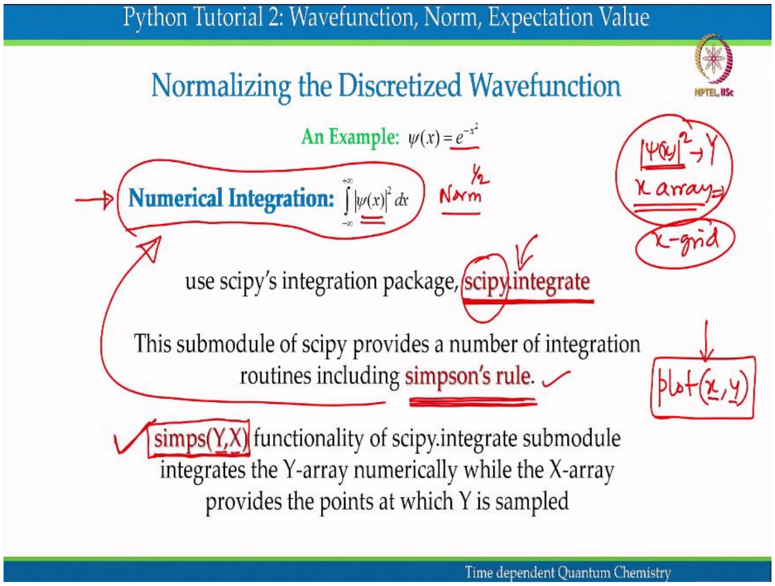
$$Norm = \left( \frac{\pi}{2} \right)^{\frac{1}{4}}$$

So, analytical form of normalized Gaussian function,

$$\psi_{normalized}(x) = \frac{e^{-x^2}}{\left( \frac{\pi}{2} \right)^{\frac{1}{4}}} = \left( \frac{2}{\pi} \right)^{\frac{1}{4}} e^{-x^2}$$

And what is our task here in this tutorial is that we normalize the discretized wave function this wave function has been discretized already, we have seen how to discretize this wave function we will use that discretized wave function and we will numerically normalize it.

(Refer Slide Time: 4:26)



Python Tutorial 2: Wavefunction, Norm, Expectation Value

## Normalizing the Discretized Wavefunction

An Example: $\psi(x) = e^{-x^2}$

Numerical Integration: $\int_{-\infty}^{+\infty} |\psi(x)|^2 \, dx$

use scipy's integration package, scipy.integrate

This submodule of scipy provides a number of integration routines including simpson's rule.

simps(Y,X) functionality of scipy.integrate submodule integrates the Y-array numerically while the X-array provides the points at which Y is sampled

Time dependent Quantum Chemistry

So, in order to do this normalization clearly we have to do this numerical integration because this integration has to be now performed numerically. This integration will give me the norm and once I get the norm I can take square root of that norm and divide this wave function by the square root of norm I get the normalized wave function. So ultimately, we have to depend on numerical integration of this absolute square of absolute value of the wave function.

Now there are many ways one can  perform numerical integration. Numerical methodology can construct a full-fledged course and in this course we are not actually going over numerical methodologies, we are always using making use of scipy all the functionalities which is implemented in scipy because scipy has been developed keeping scientific community in mind and there are many protocols which is already packaged in scipy which can be very useful for the numerical analysis or numerical solutions.

So, will make use of a highly optimized integration package of scipy it is called scipyintegrate this package provides many options to do a numerical integration. So, it has a pre-written code in it and we are going to use that code directly to do the to perform this numerical integration. The scipy integrate sub module of scipy provides a number of integration techniques including simpson’s rule, now what is the simpson’s rule? We will not go over in details in this course. It is a numerical technique which can enable one to get a numerical integration term.

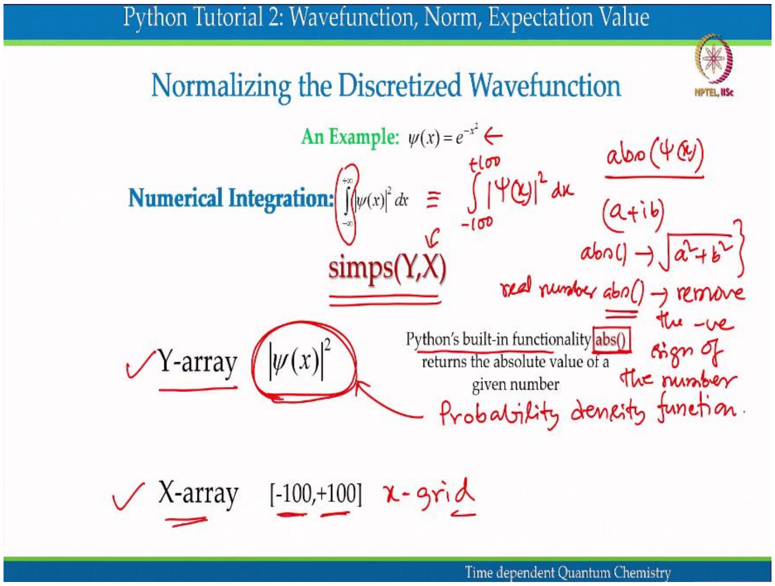
So, we will use this simpson's rule there are many other techniques which can be used which are already included in this scipy integrate package, but we are just selecting simpson's rule. And the construct which we have to use in order to implement this simpson’s rule for the numerical integration is called is following it is simps (Y,X)functionality where y would be an array and x would be also an array both would be an array.

And we have already got those arrays we in instead of y we have now x that is instead of y we have now this $|\psi(x)|^2$ this is going to be my y array and x array is already the x grid we have prepared. So, already we have both arrays in hand and we have to use that under this construct simps(Y,X) just remind we remind ourselves that when you have used plot functionality of pi plot we have used x, y plot(x,y) construct and here when you are doing in numerical integration imported from the module input imported from fi integrate I have seems y, x so y comes first and then x.

So, this is the this is something which may confuse certainly somebody if you are doing this python programming for the first time. Just remember that we have already used plot functionality of pyplot that has a construct (x,y) but here I have a construct (y,x) so do not get confused or do not shuffle these two arrays or do not write like simps(x,y) do not write it like this way. So, this is just something which may we may do something wrong in the in the coding if we write it.

So, the simps y x functionality of psi by integrate sub module integrates the y array numerically while x array provides the points at which y has to be sampled to compute the  the simpson’s rule of the numerical integration. So, we will use this, it is a very simple thing and it is a pre-written code it has, so once you invoke this simps functionality it is actually using that pre-written code in python scipy integrate sub module and it will give me the result.

(Refer Slide Time: 10:09)



Python Tutorial 2: Wavefunction, Norm, Expectation Value

## Normalizing the Discretized Wavefunction

NPTEL, IISc

An Example: $\psi(x) = e^{-x^2}$ ←

Numerical Integration: $\int_{-\infty}^{+\infty} |\psi(x)|^2\, dx \equiv \int_{-100}^{+100} |\psi(x)|^2\, dx$

simps(Y,X)

abs($\psi(x)$)

(a+ib)

abs() → $\sqrt{a^2+b^2}$

real number abs() → remove the -ve sign of the number

Python's built-in functionality abs() returns the absolute value of a given number

✓ Y-array $|\psi(x)|^2$ ← Probability density function.

✓ X-array [-100,+100] x-grid

Time dependent Quantum Chemistry

So, what we will do here in our present problem square of absolute value of y function, this is the y array as I have told before that this is going to be my y array is a square of absolute value of  the wave function. This is nothing but the probability density this is the probability density function we are familiar with is called probability density function which is square of absolute value of the wave function and this y array has to be integrated.

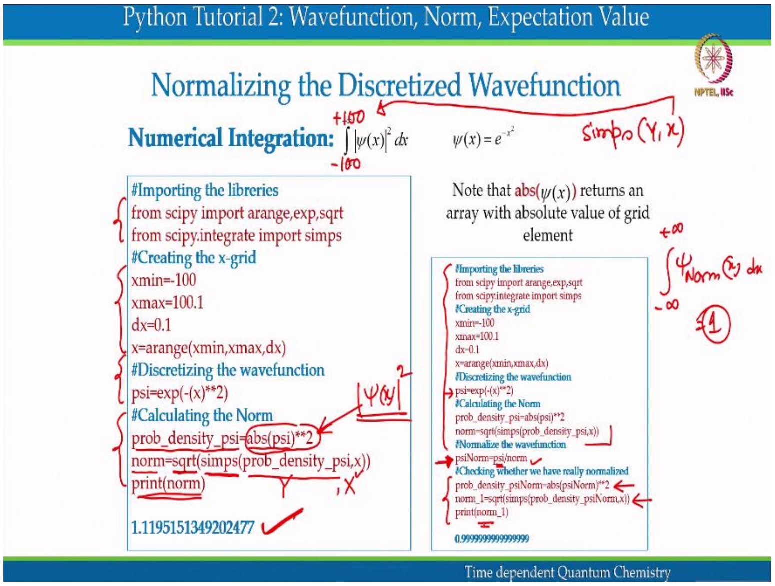
The probability density function is sampled at the points present in the x array and x array we know that it is we have defined as -100 to +100, so it is the x grid and because we are taking x grid to be -100 to +100 remember in this integration we have limit minus infinity to plus infinity and will not be able to use this limit clearly, what we will use the moment we use this simps functionality of this scipy integrate sub module, it will adopt the limit of the x grid which is minus 100 to plus 100.

So now, this is my practical numerical calculation which is going to be performed with the help of simpson's rule. Now python's built in functionality this abs absolute value abs will give me returns absolute value of a given number and that exactly what we need because this is absolute square of absolute value of the wave function and I can get this absolute value of a function by this abs which means that I have to write abs within bracket $\psi(x)$, then I get the absolute value.

So, what does it mean by absolute value? If the number is a complex number, let us say a+ib if it is a complex number, then abs will return its magnitude which is $\sqrt{a^2+b^2}$, I will get this value. And if the number is a real number if it is a real number for a real number abs functionality this functionality is actually present in pythons built in function library it does not need to be imported from other module.

So, this if it is a real number, then it will remove the negative sign of the number and that is the way it will return. So, what we need to do is that we have to use this absolute value of $\psi(x)$ and then we have to do the integration make it square and then do the integration.

(Refer Slide Time: 14:00)



Python Tutorial 2: Wavefunction, Norm, Expectation Value

## Normalizing the Discretized Wavefunction

**Numerical Integration:** $\int_{-100}^{+100} |\psi(x)|^2 \, dx$  $\psi(x) = e^{-x^2}$  simps(Y, X)

```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Discretizing the wavefunction
psi=exp(-(x)**2)
#Calculating the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x))
print(norm)

1.1195151349202477
```

$|\psi(x)|^2$

Y , X

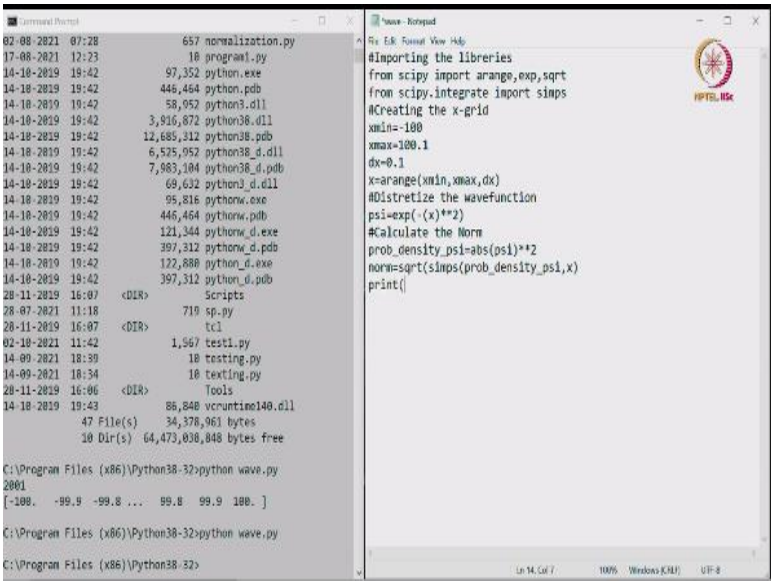
Note that abs($\psi(x)$) returns an array with absolute value of grid element

```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Discretizing the wavefunction
psi=exp(-(x)**2)
#Calculating the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x))
#Normalize the wavefunction
psiNorm=psi/norm
#Checking whether we have really normalized
prob_density_psiNorm=abs(psiNorm)**2
norm_1=sqrt(simps(prob_density_psiNorm,x))
print(norm_1)

0.9999999999999999
```

$\int_{-\infty}^{+\infty} \psi_{Norm}(x) \, dx = 1$

Time dependent Quantum Chemistry



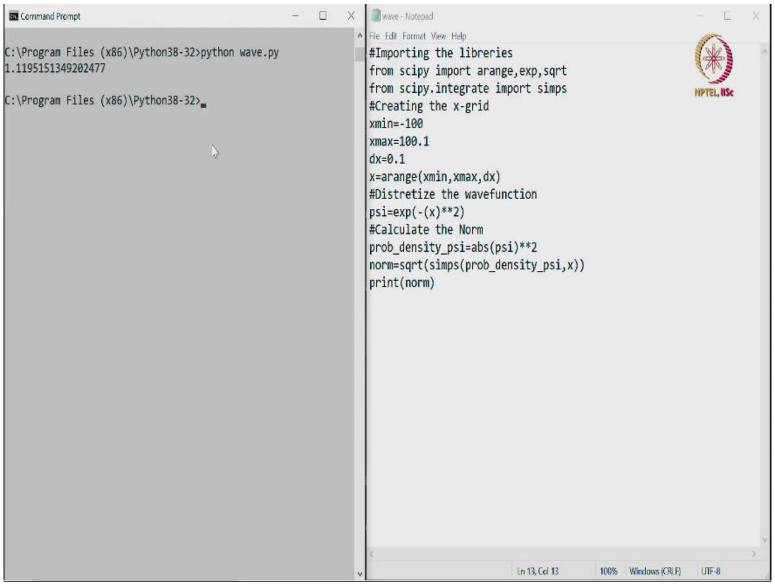
```
02-08-2021  07:28               657 normalization.py
17-08-2021  12:23                10 program1.py
14-10-2019  19:42            97,352 python.exe
14-10-2019  19:42           446,464 python.pdb
14-10-2019  19:42            58,952 python3.dll
14-10-2019  19:42         3,916,872 python38.dll
14-10-2019  19:42        12,685,312 python38.pdb
14-10-2019  19:42         6,525,952 python38_d.dll
14-10-2019  19:42         7,983,104 python38_d.pdb
14-10-2019  19:42            69,632 python3_d.dll
14-10-2019  19:42            95,816 pythonw.exe
14-10-2019  19:42           446,464 pythonw.pdb
14-10-2019  19:42           121,344 pythonw_d.exe
14-10-2019  19:42           397,312 pythonw_d.pdb
14-10-2019  19:42           122,880 python_d.exe
14-10-2019  19:42           397,312 python_d.pdb
28-11-2019  16:07    <DIR>          Scripts
28-07-2021  11:18               719 sp.py
28-11-2019  16:07    <DIR>          tcl
02-10-2021  11:42             1,567 test1.py
14-09-2021  18:39                10 testing.py
14-09-2021  18:34                10 texting.py
28-11-2019  16:06    <DIR>          Tools
14-10-2019  19:43            86,840 vcruntime140.dll
              47 File(s)     34,378,961 bytes
              10 Dir(s)  64,473,038,848 bytes free

C:\Program Files (x86)\Python38-32>python wave.py
2001
[-100.   -99.9  -99.8 ...   99.8   99.9  100. ]

C:\Program Files (x86)\Python38-32>python wave.py

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

```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Distretize the wavefunction
psi=exp(-(x)**2)
#Calculate the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x)
print(
```



```
C:\Program Files (x86)\Python38-32>python wave.py
1.1195151349202477

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

```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Distretize the wavefunction
psi=exp(-(x)**2)
#Calculate the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x))
print(norm)
```

So, we will now numerically implement this idea. Clearly minus infinity to plus infinity is not the limit, for us it is going to be -100 to +100 limit the limit has been will be automatically selected in the simps if I use sims functionality this is y then x then x grid limit would be selected as the limit for the integration.

So, we will see that the first two lines are importing the libraries then we have to definitely create the x grid once we have created the x grid we have to discretize the wave function we have already seen how to do that, then these are the new steps for the normalization what and this is very simple three steps normalization steps.

First, we will define the probability density of $\psi(x)$, how we are going to define? It is nothing but absolute value of $\psi(x)$ square, so it so this this part is nothing but psi x square, that is the probability density and then we have to find out the norm, norm is going to be square root of this integration of this probability density versus x.

So, we have we have defined this is my y array this is my x array. So, I have to first integrate it and after the integration I have to take the value square root of that value and then we are going to print the norm we will just double check what is the norm we are getting. So, we will move to laptop and we have these libraries importing the libraries we are going to keep it as it is but the plotting part we do not need because we are not going to plot it anymore instead of plot we have to import the simps functionality from scipy integrate sub module.

So, I will write down from scipy dot integrate import simps. So, that I can use that later, so this part remaining to be the same because we are creating the x grid first, then we are discretizing the wave function then this plot part we do not need anymore, we have to now after discretizing the wave function we have to now calculate the norm, and how do I calculate it? First I have to get the probability density of $\psi(x)$.

So, probability density (probe_density) just giving a name with underscores so that anybody can understand it when is reading it later stage, so using a good name of a particular function can help understand what is the meaning of that function, so that's the reason I am explicitly writing like this way otherwise, one can use also y to represent the probability density function. So, probability density function, what does it mean?

It is the absolute value, first I  have to take the absolute value of psi so this is going to give me the absolute value of psi which is another array basically it will take absolute value of each point each element of psi and then this needs to be I  have to take the square of that value each value. So, probability density gives me another array where each element where we have taken absolute value of each element in the array and then we have squared it.

So, norm would be defined by square root but we have not imported square root, square root is not a mathematic this mathematical function squared mathematical function is not available with pythons built-in library it has to be imported from scipy. So, you will be using sqrt remember many of these functionality can be imported from numpy as well but we are deliberately not using numpy at all we do not want to mix up the libraries we want to just keep things simple that is why we are using always this scipy module because it is more optimized and will keep using the same module for different purpose.

So, norm, I have to define by square root of the integration of this function, so I will use simps then I have to use y,x, x is already defined and then I have to use print to check what is the norm I get. So, we have to check when you are using many brackets we have to always think we have to double check whether the brackets are in proper order or not we have two brackets here, so we have two brackets here also and that exactly what we need should always double check.

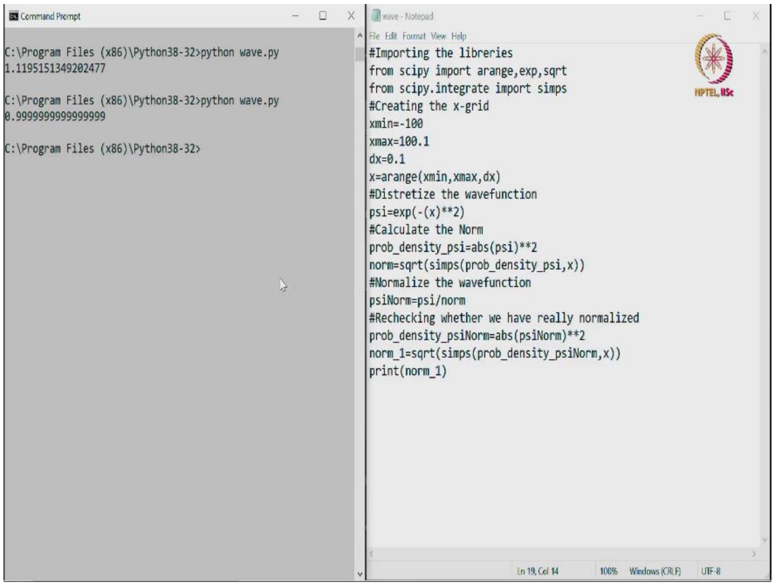
So, if we run the program right now, then we get back the norm, norm is 1.119, we go back to the slides we get this value for the norm, this is nothing but the value which we have used before we analytically solve the problem and we have found that the norm has to be $\left(\frac{\pi}{2}\right)^{\frac{1}{4}}$. So, this is the value which we get here numerically. We will move on and we will once we have understood that we have been able to get the norm remaining part is very simple we have to just divide the wave function by the norm and we get the normalized wave function.

So, this part for the normalization is quite understandable up to this far we have understood how to get the norm and then we have to name another we are giving a name psi_norm which is normalized wave function which is nothing but the earlier wave function psi this is the earlier wave function which was not normalized, I am now dividing by the norm and I am getting I should get the normalized wave function.

So, what we have done is that to make sure that we have normalized it, we can always double check by taking this integration one more time psi if it is a normalized wave function, then this is going to be 1. So, we are doing one more time we are taking this as a wave function normalized wave function and checking whether we are getting norm to be 1 or not. So, these 3 steps has been written to recheck whether we have got the normalized wave function.

So, for that we have to again take the probability density, then the integration and then square root of that integration is going to be norm and I am going to print norm.

(Refer Slide Time: 22:39)



```
C:\Program Files (x86)\Python38-32>python wave.py
1.1195151349202477

C:\Program Files (x86)\Python38-32>python wave.py
0.9999999999999999

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

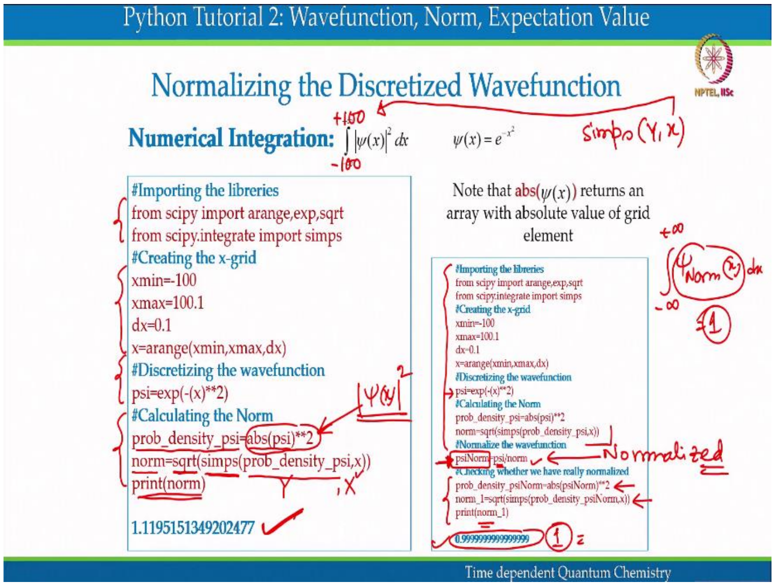
```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Distretize the wavefunction
psi=exp(-(x)**2)
#Calculate the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x))
#Normalize the wavefunction
psiNorm=psi/norm
#Rechecking whether we have really normalized
prob_density_psiNorm=abs(psiNorm)**2
norm_1=sqrt(simps(prob_density_psiNorm,x))
print(norm_1)
```

So, we will move to laptop and we will just we do not need to print the norm anymore we will just keep the norm and then we will just normalize the wave function we can normalize the wave function by writing psiNorm, this is a new name we are giving which is nothing but psi divided by norm.

So, I have this normalized wave function remember psi norm still an array it has 2001 number of points and this is representing an array. So, psi divided by norm it means that each element of psi has been has been divided by the norm value. Once we get that, then we will recheck whether we have really normalized the wave function for that will define the probability density of this normalized wave function we are giving the pattern of naming we are keeping similar probability density psiNorm.

So, this is going to be absolute value of this normalized wave function, what it does? It will take the absolute value for each element of this array and then square it so this is again another array we have prepared and then I will write norm 1 this is going to be then square root of here psi norm. So, y array is going to be psi probability density of psi norm and then if I want to print I will write down so I execute the program what I get back is 0.999 which is equivalent to 1.

(Refer Slide Time: 25:21)



Python Tutorial 2: Wavefunction, Norm, Expectation Value

## Normalizing the Discretized Wavefunction

**Numerical Integration:** $\int |\psi(x)|^2 dx$ $\psi(x) = e^{-x^2}$

```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Discretizing the wavefunction
psi=exp(-(x)**2)
#Calculating the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x))
print(norm)

1.11195151349202477
```

Note that abs($\psi(x)$) returns an array with absolute value of grid element

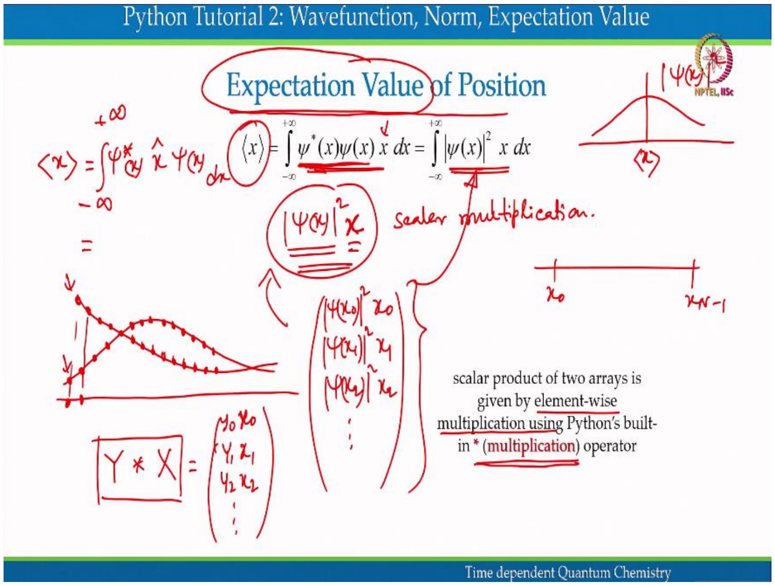
```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Discretizing the wavefunction
psi=exp(-(x)**2)
#Calculating the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x))
#Normalize the wavefunction
psiNorm=psi/norm
#Checking whether we have really normalized
prob_density_psiNorm=abs(psiNorm)**2
norm_1=sqrt(simps(prob_density_psiNorm,x))
print(norm_1)

0.9999999999999999
```

Time dependent Quantum Chemistry

So, I will go back to the slides, so I have this value which is equivalent to 1. So, this kind of floating point value will make it depending on because it is a numerical integration we may not get 1 but 0.999 is equivalent to 1. So, because it is 1 the norm is getting 1 it means that the wave function which I have right now here the psi norm this psi norm is now a normalized wave function that we have proved and always in quantum dynamics we have to always use a normalized wave function this is something which we have learned in this course.

(Refer Slide Time: 26:14)



Python Tutorial 2: Wavefunction, Norm, Expectation Value

## Expectation Value of Position

$$\langle x \rangle = \int_{-\infty}^{+\infty} \psi^*(x)\psi(x)\, x\, dx = \int_{-\infty}^{+\infty} |\psi(x)|^2\, x\, dx$$

scalar product of two arrays is given by element-wise multiplication using Python's built-in * (multiplication) operator

Time dependent Quantum Chemistry

We will move on and we will look at the expectation value of position. Now expectation value of position, what does it mean? I just remind if I have a probability density distribution which is psi x absolute square if I have this, then question is what is the mean of this or average of this distribution. So, clearly for Gaussian probability density distribution the mean would be here this is called the mean value, and what does it mean?

Experimentally it means that is the average of many repetitive experiments that is the average value. So, expectation value has a direct connection to the experimental outcome because in the experiment we repeat the experiment many times and after repeating it, if we take the average of the outcome we get the value that value should correspond the x correspond to this expectation value which can also be calculated from the known wave function of the system, so that is the basic idea.

Expectation value of position is given by this in general it is given by if x is an operator, then expectation value is going to be

$$\langle x \rangle = \int_{-\infty}^{\infty} \psi^*(x)\hat{x}\psi(x)dx = \int_{-\infty}^{\infty} |\psi(x)|^2\, xdx$$

x,s position does not mean anything this is multiplication, multiplication does not follow any order it can be multiplied by this way it can be multiplied by this way both side is perfectly fine.

So, but operators in general operators does not work that way, if we consider the derivative operator if I have y z, then this derivative operator is acting only on z it is not acting on y, so operator in general has a particular direction but position operator because it is a multiplication operator it does not have any direction so one can position it anywhere so I have position it here so that I can club this size star side together and $|\psi(x,t)|^2$ ,the absolute value.

So, in the end expectation value if we want to calculate all we need to do is that we have to construct another function, what is that function? The first function is the probability function which is psi x absolute value square of absolute value of $\psi(x)$, this is one function it has to be multiplied by x the x is another function. So, these two functions needs to be multiplied.

Now if we think about this probability density distribution looks like this and x has x function this function has certain way let us say this is the way x is working and if we multiply two functions let us say these two functions are multiplying in terms of matrix or in terms of this discretized on a grid it is nothing but multiplying each element if it is a simple multiplication scalar multiplication then it is multiplying each element to get the final form of the total function.

So, if I multiply so this element has to be multiplied by this element and each element will be defined by its own grid point. So, for corresponding to each grid point I have two elements for coming from two functions and they needs to be multiplied, so each element has to be multiplied so it is a element wise multiplication because it is a scalar multiplication.

And scalar multiplication so it is very easy to do and so in the end this is an array which will be represented by leak this way

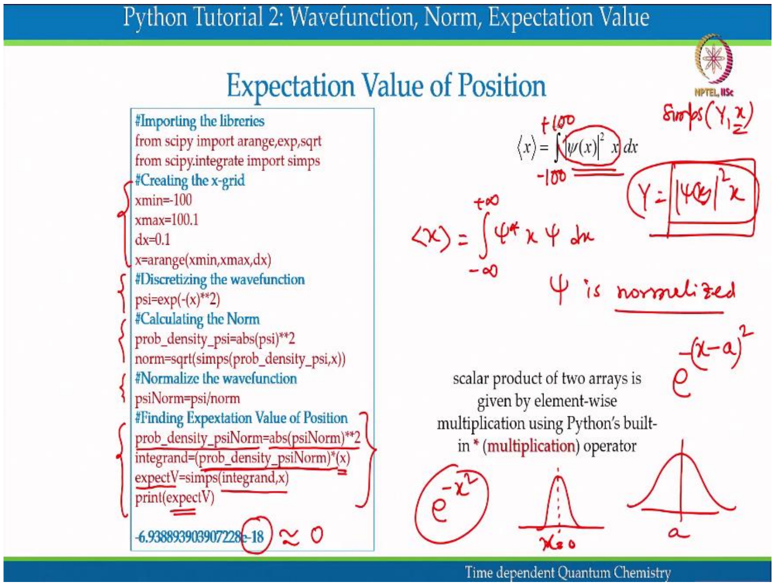
$$\begin{pmatrix} |\psi(x_0)|^2 \, x_0 \\ |\psi(x_1)|^2 \, x_1 \\ |\psi(x_2)|^2 \, x_2 \\ . \\ . \end{pmatrix}$$

So these are the points we get in the probability density distribution. So, this is the array we create by this multiplication and this is the array which will be using for the integration following the simpson’s method as an y array.

So, but the question is in python programming the scalar product of two arrays is given by element wise multiplication and element wise multiplication can be done by just multiplication operator which is available in python's built-in library. ,So, if I have y array and x array if I multiply then what will happen it will do element wise multiplication each element will be multiplied by the corresponding other element. So, y1 will be multiplied by x1 y0 will be multiplied by x0, then y2 will be multiplied by x2 and so on and then it will give me another array.

$$Y * X = \begin{pmatrix} y_0 x_0 \\ y_1 x_1 \\ y_2 x_2 \\ . \\ . \end{pmatrix}$$

(Refer Slide Time: 32:40)



Python Tutorial 2: Wavefunction, Norm, Expectation Value

## Expectation Value of Position

```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Discretizing the wavefunction
psi=exp(-(x)**2)
#Calculating the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x))
#Normalize the wavefunction
psiNorm=psi/norm
#Finding Expextation Value of Position
prob_density_psiNorm=abs(psiNorm)**2
integrand=(prob_density_psiNorm)*(x)
expectV=simps(integrand,x)
print(expectV)

-6.938893903907228e-18
```

$\approx 0$

$$\langle x \rangle = \int_{-100}^{+100} |\psi(x)|^2 \, x \, dx$$

Simps(Y, x)

$$Y = |\psi(x)|^2 x$$

$$\langle x \rangle = \int_{-\infty}^{+\infty} \psi^* x \psi \, dx$$

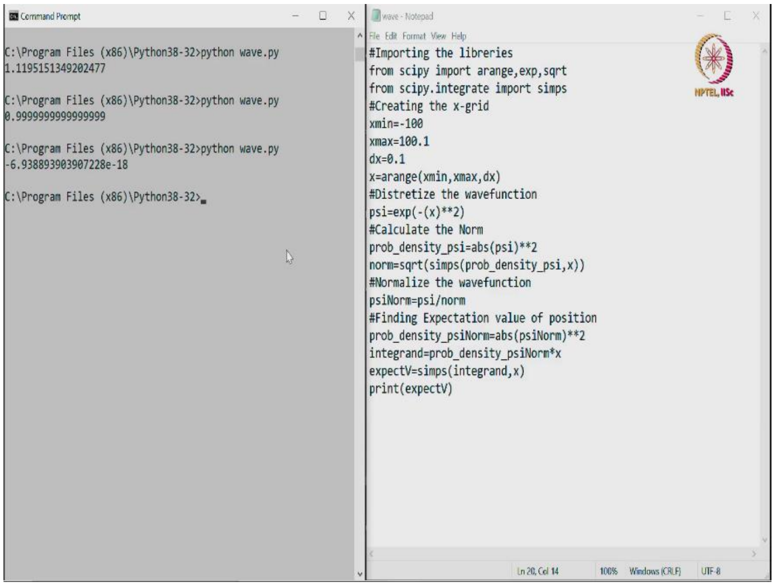
$\psi$ is normalized

$e^{-(x-a)^2}$

scalar product of two arrays is given by element-wise multiplication using Python's built-in * (multiplication) operator

$e^{-x^2}$

Time dependent Quantum Chemistry



```
C:\Program Files (x86)\Python38-32>python wave.py
1.1195151349202477

C:\Program Files (x86)\Python38-32>python wave.py
0.9999999999999999

C:\Program Files (x86)\Python38-32>python wave.py
-6.938893903907228e-18

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

```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Distretize the wavefunction
psi=exp(-(x)**2)
#Calculate the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x))
#Normalize the wavefunction
psiNorm=psi/norm
#Finding Expectation value of position
prob_density_psiNorm=abs(psiNorm)**2
integrand=prob_density_psiNorm*x
expectV=simps(integrand,x)
print(expectV)
```

So, it is very simple, so we will move forward and we will see how to get this integration done so that i can get the expectation value this is the integration and again practically we cannot do this minus infinity to plus infinity integration this is going to be done with the help of Simpson rule and for that it will just pick up the integration limit as the x grid limit, so this is going to be our integration this entire part is going to be y.

So, in the Simpson rule I have to specify y and y is going to be this part and x is the x grid over the x grid it will be sampling the y values and then integrate it this x grid creation part is known to us now discretization of the function is known to us then calculating the norm we have to first normalize the wave function because this expectation value this when we write down

$$\langle x \rangle = \int_{-\infty}^{\infty} \psi^*(x)\hat{x}\psi(x)dx = \int_{-\infty}^{\infty} |\psi(x)|^2 \, xdx$$

The expectation value we assume that psi is normalized.

So, we have to first normalize the wave function to get the right expectation value so we normalized it this part is also known. And then, little bit tricky we have made here first we have to get the probability density of the normalized wave function that is known to us now. Then the integrand part we will just write down this is the integrand part, so what we have done here this probability density will be multiplied by x we get the integrand and then expectation value is going to give me the integration part where y is this entire integrand and then we will print the expected value expectation value for this.

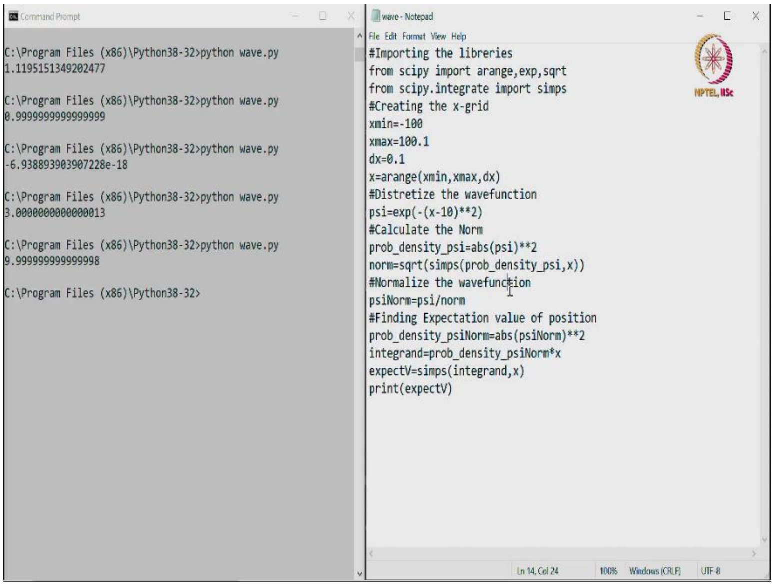
So, this part is little new others are known to us, so move to the laptop and will write down the program we have everything remaining to be the same then except for this this is x grid we have prepared then discretize the wave function, then we have to normalize the wave function this far we are done and then we have to find out the expectation value of the position.

So, this part will be deleted and will write down finding expectation value of position where first we have to write down this probability density psiNorm which we need to do because after we normalize it we have to find out the probability density and then this probability density has to be multiplied by x so that I can create another we multiply by x.

And then expectation value is expectV which is represented representing the expectation value expectation value we get from this integration and in this integration y has to be this integrand and then we are going to print the expectation value. So, if we execute the program we get minus we go back to the slide we get -6*10$^{-18}$ which is almost equal to 0, it is the 0 position and that should be because e to the power our wave function was e to the power minus x square this is a Gaussian function which is centered at x equals 0.

A Gaussian function which is centered at x equals 0 is having this form but if I want to change the Gaussian function to be centered at different place then I have to write x minus a whole square if I do that then Gaussian will be centered at position a and that will prove here.

(Refer Slide Time: 38:14)



```
C:\Program Files (x86)\Python38-32>python wave.py
1.1195151349202477

C:\Program Files (x86)\Python38-32>python wave.py
0.9999999999999999

C:\Program Files (x86)\Python38-32>python wave.py
-6.938893903907228e-18

C:\Program Files (x86)\Python38-32>python wave.py
3.0000000000000013

C:\Program Files (x86)\Python38-32>python wave.py
9.999999999999998

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

```
#Importing the libreries
from scipy import arange,exp,sqrt
from scipy.integrate import simps
#Creating the x-grid
xmin=-100
xmax=100.1
dx=0.1
x=arange(xmin,xmax,dx)
#Distretize the wavefunction
psi=exp(-(x-10)**2)
#Calculate the Norm
prob_density_psi=abs(psi)**2
norm=sqrt(simps(prob_density_psi,x))
#Normalize the wavefunction
psiNorm=psi/norm
#Finding Expectation value of position
prob_density_psiNorm=abs(psiNorm)**2
integrand=prob_density_psiNorm*x
expectV=simps(integrand,x)
print(expectV)
```

We will move to laptop I will just change this the definition of this Gaussian function, according to this definition which I have used right now, this is centered at x equals 0 now I will center it at x equals let us say 3, x minus 3 that is why I am writing and once we do that then we will see that will check what is the expectation value we have, expectation value we get 3.000 which is close to 3. Numerically we get some error always and we have to deal with this kind of error but this is 3 we are getting.

So basically, what we are seeing is that center of the Gaussian is representing the expectation value of the position and that is now 3. If I change 3 to 10 I will see that expectation value I will get 10 instead of 10 I get 9.999 which is equivalent to 10 numerically. So, what we see here is that the for a Gaussian function the center of the Gaussian is actually represented by the expectation value of position that we have seen before also. And one can do analytically one can one can perform analytical integration for this with the help of this kind of function to check whether really expectation value would be at this value this the at this parameter which we are using.

So, we have come to end of this of this course, end of this module this python tutorial 2 where we have learned how to represent the wave function in the computer programming with the help of python and then how to normalize the wave function that is very important step because in every quantum dynamics problem will be normalizing it and first we have to normalize it and then we have to move forward and then we have learned how to get the expectation value. We will stop here and we will continue learning time dependent quantum chemistry in this course in the next module and next tutorials.