DOING PHYSICS WITH PYTHON

Ian Cooper

matlabvisualphysics@gmail.com

QUANTUM MECHANICS
A NUMERICAL APPROACH
OPERATORS: EXPECTATION VALUES AND THE HEISENBERG UNCERTAINTY PRINCIPLE

DOWNLOAD DIRECTORIES FOR MATLAB SCRIPTS

qm002.PY

Google drive

GitHub

INTRODUCTION

The state of a system can be represented by a vector and operators by matrices. So, it is very convenient to use Python for simulating quantum systems.

State vectors in Dirac notation:

column vector ket $|\psi\rangle$

row vector bra $\langle\psi|$

Differential operators:

matrices (ket-bras) acting on those vectors

Dirac introduced bra-ket notation for state vectors and operators in 1930. This notation emphasized and clarified the role of inner products and linear function spaces and his idea of the bra-ket notation is fundamental to our understanding of quantum mechanics.

Two key concepts underpinning quantum physics are the Schrodinger equation and the Born probability equation.

The Schrodinger equation tells us how the state of a system evolves in time. The **[1D] time dependent Schrodinger equation** is

$$(1) \quad i\hbar \frac{\partial}{\partial t} \left| \Psi(x,t) \right\rangle = \hat{H} \left| \Psi(x,t) \right\rangle$$

where $\left| \Psi(x,t) \right\rangle$ is the ket vector representing the state of the system (particle) and $\hat{H}$ the Hamiltonian operator.

In our approach, the state vector evolves in time, and operators corresponding to observables (and hence their eigenbasis of vectors) are time independent and the Hamiltonian is time-independent.

When a measurement of a physical quantity A is made on a particle initially in the state $\left| \psi(x,t) \right\rangle$, the **Born equation** provides a way to calculate the probability $P(A_0)$ that a particular result A_0 is obtained from the measurement.

$$(2) \quad P(A_0) \sim \left\| \left\langle A_0 \middle| \Psi \right\rangle \right\|^2 \qquad \left\langle A_0 \right| = \left| A_0 \right\rangle^*$$

where $\left| A_0 \right\rangle$ is the state vector corresponding to the particular result A_0 having been measured, and $\left\langle A_0 \middle| \Psi \right\rangle$ is called the inner product.

EXPECTATION VALUES OF OBSERVABLES

The wavefunction $\Psi(x,t)$ itself has no definite physical meaning, however, by performing mathematical operations on the wavefunction we can predict the mean values (**expectation values**) and their uncertainties of physical quantities such as position, momentum and energy.

The **expectation value** of an observable quantity A is the quantum-mechanical prediction for the mean value of <A>.

We will consider a particle to be an electron and that we know the wavefunction $\Psi(x,t)$ for the system of the single electron. Since an electron does exist within the system, the probability of finding the electron is 1.

(1) $$\int_{-\infty}^{\infty} \Psi(x,t)^* \Psi(x,t) dx = 1$$

The quantity $\Psi(x,t)^*\ \Psi(x,t)$ is called the **probability density function**.

The probability of finding the electron in the region from x_1 to x_2 is

(2) $$prob\left(x_1 \le x \le x_2\right) = \int_{x_1}^{x_2} \Psi(x,t)^*\ \Psi(x,t)\ dx$$

Consider *N* identical systems, each containing an electron. We make *N* identical measurements on each system of the physical parameter *A*. In a quantum system, each measurement is different. From our *N* measurements, we can calculate the mean value $\langle A \rangle$ and the standard deviation ΔA of the parameter *A*. Since we do not have a complete knowledge of the system, we only can estimate probabilities.

The mean value of an observable quantity *A* is found by calculating its **expectation value** $\langle A \rangle$ by evaluating the integral

$$\langle A \rangle = \int_{-\infty}^{\infty} \Psi(x,t)^* \, \hat{A} \, \Psi(x,t) \, dx \tag{3}$$

where $\hat{A}$ is the quantum-mechanical operator corresponding to the observable quantity *A*, and its **uncertainty** is the standard deviation of *A*. The uncertainty ΔA is given by

$$\Delta A = \left[\langle A^2 \rangle - \langle A \rangle^2 \right]^{1/2} \tag{4}$$

$$\langle A^2 \rangle = \int_{-\infty}^{\infty} \Psi(x,t)^* \, \hat{A}\hat{A} \, \Psi(x,t) \, dx \tag{5}$$

The quantity $\Psi(x,t)^* \, \hat{A} \, \Psi(x,t)$ represents the **probability distribution** for the observable *A* and is often shown as a probability density vs position graph. Probability distributions

summarize the extent to which quantum mechanics can predict the likely results of measurements. The probability distribution is characterized by two measures – its **expectation value** which is the mean value of the distribution and its **uncertainty** which is the represents the spread in values about the mean and is given by the **standard deviation**. The R.H.S. of the integrals (3) and (5) are known as **sandwich integrals**.

For our [1D] system, a particle in any state must have an uncertainty in position Δx and uncertainty in momentum Δp that obeys the inequality called the **Heisenberg Uncertainty Principle**

$$(6) \qquad \Delta x \, \Delta p \geq \frac{\hbar}{2}$$

The Heisenberg Uncertainty Principle tells us that it is impossible to find a state in which a particle can has definite values in both position and momentum. Hence, the classical view of a particle following a well-defined trajectory is demolished by the ideas of quantum mechanics.

Table 1 gives a summary of the most important operators for [1D] quantum systems. Expectation values maybe time dependent.

Table 1. Observables, Operators and Expectation values

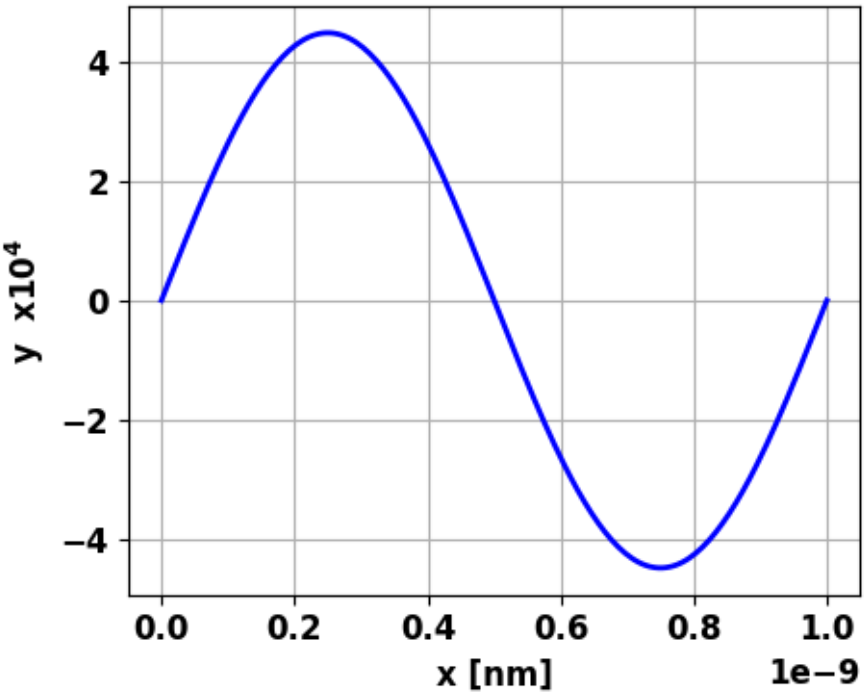
Observable	Operator	Expectation Value
probability	1	$\int_{-\infty}^{\infty} \Psi(x,t)^* \; \Psi(x,t)\, dx = 1$
position	x	$\langle x \rangle = \int_{-\infty}^{\infty} \Psi(x,t)^* \; x \; \Psi(x,t)\, dx$
x^2	x^2	$\langle x^2 \rangle = \int_{-\infty}^{\infty} \Psi^*(x,t) \; x^2 \; \Psi(x,t)\, dx$
momentum	$-i\hbar \frac{\partial}{\partial x}$	$\langle p \rangle = -i\hbar \int_{-\infty}^{\infty} \Psi(x,t)^* \; \frac{\partial \Psi(x,t)}{\partial x}\, dx$
p^2	$-\hbar^2 \frac{\partial^2}{\partial x^2}$	$\langle p^2 \rangle = -\hbar^2 \int_{-\infty}^{\infty} \Psi(x,t)^* \; \frac{\partial^2 \Psi(x,t)}{\partial x^2}\, dx$
Potential energy	U	$\langle U \rangle = \int_{-\infty}^{\infty} \Psi(x,t)^* \; U(x) \; \Psi(x,t) \quad dx$
Kinetic energy	K	$\langle K \rangle = -\frac{\hbar^2}{2m} \int_{-\infty}^{\infty} \Psi(x,t) \; \frac{\partial^2 \Psi(x,t)}{\partial x^2}\, dx$

EXAMPLE

The eigenfunctions for an electron confined to an infinite potential well of width *a* are

$$y_n = \sqrt{\frac{2}{L}} \sin\left(\frac{n\pi x}{a}\right) \quad n = 1, 2, 3, \ldots$$

The potential energy *U* is zero within the well and infinite outside the well. Consider the case when $n = 2$ (first excited state) and the well width is 1.00 nm. The wavefunction for the eigenstate $n = 2$ is shown in the figure.



Knowing the wavefunction *y*, it is a straight forward procedure to calculate expectation values of key physical quantities and test the uncertainty principle using Python. The code **qm004.py** was used to

make the plot and do the calculations. A summary of the calculations are displayed in the Console Window.

prob = 1.000

xavg = 0.500 nm

deltax = 2.658e-10 m

deltap = 6.626e-25 N.s

Delta = 1.761e-34 J.s

<K> = 1.504e+00 eV

lambda = 1.000 nm

p = 6.626e-25 N.s

K = 1.504e+00 eV

The probability of finding the electron is 1 and the average position of the electron is at the centre of the well ($< x >$ = 0.500 nm). The uncertainty principle is satisfied as

$$Delta = \Delta x \Delta p = 1.761 \times 10^{-34} > \hbar / 2 = 5.27 \times 10^{-35}$$

The total energy E is equal to the sum of the kinetic energy K and the potential energy U

$E = K + U = K$ since $U = 0$ inside the well

The theoretical eigenstate total energy is

$$p = \frac{h}{\lambda} \quad E = \frac{p^2}{2m_e} = \frac{h^2}{2m_e \lambda^2} = 1.504 \text{ eV}$$

which agrees with the expectation calculation of the total energy $< E >$ = 1.504 eV.

```
# -*- coding: utf-8 -*-
"""
# qm004.py       April 2024
# Ian Cooper        matlabvisualphysics@gmail.com
# QUANTUM MECHANICS
#  EXPECTATION VALUES AND THE UNCERTAINTY PRINCIPLE
# Website: https://d-arora.github.io/Doing-Physics-With-Matlab/
# Documentation: https://d-arora.github.io/Doing-Physics-With-
Matlab/pyDocs/qm002.pdf
"""

import time
import math
import numpy as np
import pylab as py
from matplotlib import pyplot as plt
from numpy import linspace,sin,cos,exp, zeros, ones, pi, diag, sqrt,
real, imag
from scipy.integrate import odeint, quad, dblquad, simps

from matplotlib.ticker import (MultipleLocator,
                               FormatStrFormatter,
                               AutoMinorLocator)

#%% FUNCTIONS

def firstDer(N,dx):
    v  = ones(N-1)
    M1 = diag(-v,-1)
    M2 = diag(v,1)
    M = M1+M2
```

```
    M[0,0] = -2; M[0,1] = 2; M[N-1,N-2] = -2; M[N-1,N-1] = 2
    MF = M/(2*dx)
    return MF

def secondDer(N,dx):
    v = -2*ones(N)
    M1 = np.diag(v)
    v = np.ones(N-1)
    M2 = np.diag(v,1)
    M3 = np.diag(v,-1)
    M = M1+M2+M3
    M[0,0] = 1; M[0,1] = -2; M[0,2] = 1
    M[N-1,N-3] = 1; M[N-1,N-2] = -2; M[N-1,N-1]=1
    MS = M/(dx**2)
    return MS


# INPUTS
# Quantum number
n = 2
# Grid points
N = 199
# Well width
a = 1e-9
# m to nm
L = 1e9

# SETUP
hbar = 1.054571817e-34; me = 9.11e-31;
e = 1.602e-19; h = 6.626e-34
# X grid
```

```
x1 = 0; x2 = a; x = linspace(x1,x2,N); dx = x[2] - x[1]
# Wavefunction
y = sqrt(2/a)*sin(n*pi*x/a)

wL = 2*a/n
p = h/wL
K = p**2/(2*me)

# Probability
fn = y*y
prob = simps(fn,x)

# Expectation value & Uncertainty x
fn = y*x*y
xavg = simps(fn,x)
fn = y*x**2*y
x2avg = simps(fn,x)
deltax = sqrt(x2avg - xavg**2)

# Expectation value & Uncertainty p
y1dash = firstDer(N,dx)@y
fn = y*y1dash
pavg = -1j*hbar*simps(fn,x)

y2dash = secondDer(N,dx)@y
fn = y*y2dash
p2avg = -hbar**2*simps(fn,x)
deltap = sqrt(p2avg - imag(pavg)**2)

delta = deltax*deltap
```

```
# Expectation KE
Kavg = -hbar**2*simps(fn,x)/(2*me)

# Console output
v = prob; print('prob =  %2.3f'  % v)
v = xavg*L; print('xavg =  %2.3f  nm'  % v)
v = deltax; print('deltax =  %2.3e  m'   % v)
v = deltap; print('deltap =  %2.3e  N.s'   % v)
v = delta; print('Delta =  %2.3e J.s'   % v)
v = Kavg/e; print('<K> =  %2.3e eV'   % v)
v = wL*L; print('lambda =  %2.3f nm'   % v)
v = p; print('p =  %2.3e N.s'   % v)
v = K/e; print('K =  %2.3e eV'   % v)

#%% GRAPHICS
plt.rcParams['font.size'] = 12
plt.rcParams["figure.figsize"] = (5,4)

fig, ax = plt.subplots(1)
ax.xaxis.grid()
ax.yaxis.grid()
ax.set_ylabel('y  x10$^4$ ',color= 'black')
ax.set_xlabel('x [nm]',color = 'black')
fig.tight_layout()
ax.plot(x,y/1e4,'b', lw = 2)


# fig.savefig('a1.png')
```