particle_in_a_box

March 14, 2024

1 Particle in a Box

Particle-in-a-box:

$$\hat{H} = \frac{-\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x)$$

where $0 < x < L = V$, and $x = \infty$ everywhere else. Solve to find:

$$\psi(x) = \sqrt{\frac{2}{L}}\sin(\frac{n\pi}{L}x) \tag{1}$$

$$E_n = \frac{n^2\hbar^2\pi^2}{2mL^2} \tag{2}$$

If we include the time-dependent phase factor:

$$\Psi(x,t) = \psi(x)e^{-iEt/\hbar}$$

Constants: - reduced Planck's constant: $(\hbar) = 1.05457182 * 10^{-34}$ $\mathrm{m^2 kg\ s^{-1}}$ - mass of electron: $m_e = 9.1093837 * 10^{-31}$ kg - angstrom: $\mathrm{\AA} = 10^{-10}$ m

[ ]:
```
import matplotlib
# Enable interactive plot
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import numpy as np

A = 1e-10               # m; one angstrom
me = 9.1093837e-31      # kg; mass of electron
hbar = 1.05457182e-34   # m^2 kg / s
n_grid = 100

# L = 10*A

#
L = 1
hbar = 1
```

```
x = np.linspace(0, L, n_grid)
delta_x = x[1]-x[0]


def energy(n, m=me, L=L):
    return (n * hbar * np.pi / L)**2 / (2 * m)

def norm(x, p):
    return (p.conj()*p).real

def psi(x, n, t=0, m=me, L=L):
    psi_over_time = np.sqrt(2 / L) * np.sin(np.pi * n * x / L) * np.exp(-1j*␣
 ↪energy(n,m,L) * t / hbar)
    return psi_over_time
```

2 Plot

Now let's plot the wavefunction

[ ]:
```
import plotly.express as px
import pandas as pd


# t = np.linspace(0, 1e-15, 100)
t = np.linspace(0, 10, 100)

data = []
# m = me
m = 1

ki = 2 # this is the eigenstate

print(" Energy is: ", energy(1, m=m, L=L))

for ti in t:
    nt = 0
    for xi in x:
        p = psi(xi, ki, t=ti, m=m)

        data.append({'x':xi, 'Real':p.real, 'Imag':p.imag, 'Dens':norm(x,p),␣
 ↪'t':ti})

df = pd.DataFrame(data)

ymax = np.max(np.abs(df.Dens))
```

```
fig = px.scatter(df, x='x', y=['Imag', 'Real', 'Dens'], animation_frame='t',␣
 ↪range_y=[-ymax,ymax])

print(np.sum(df[df.t==0]["Dens"])*delta_x)

fig.show()
```

```
 Energy is:  4.934802200544679
1.0
```

3 Non-stationary state

Now let's plot the time evolution of the system in a superposition of different energy eigenstates:

[ ]:
```
data = []
m = 1
k = [1,2,3,4,5,6,7,8,9,10]
k = [1,2]

print(" Energy is: ", energy(1, m=m, L=L))

for ti in t:
    nt = 0
    for xi in x:
        p = 0
        for ki in k:
            p += psi(xi, ki, t=ti, m=m)
        p = p / np.sqrt(len(k))

        data.append({'x':xi, 'Real':p.real, 'Imag':p.imag, 'Dens':norm(x,p),␣
 ↪'t':ti})

df = pd.DataFrame(data)

ymax = np.max(np.abs(df.Dens))

fig = px.scatter(df, x='x', y=['Imag', 'Real', 'Dens'], animation_frame='t',␣
 ↪range_y=[-ymax,ymax])


fig.show()
```

```
 Energy is:  4.934802200544679
```

[ ]:
```
fig = px.line_3d(df, x='x', y='Imag', z='Real', labels={'x':'x', 'y':'Real',␣
 ↪'z':'Imag'}, animation_frame="t", range_y=[-ymax,ymax], range_z=[-ymax,ymax])
fig.show()
```

[ ]:

[ ]: