Python Class 10: Quantum Dynamics, or, Making Wavefunctions Wave

1. Introduction

Up to this point we have not discussed how quantum wavefunctions evolve in time. That silence ends today. In what follows I want you to keep in mind the Python exercise in which you animated the motion of a classical string. The goal in that exercise was to animate a triangle-shaped string by superposing many sinusoidal modes with appropriately chosen coefficients. You made the string evolve in time by including the time dependence appropriate for each of the normal modes.

In the PHYS 212E Supplementary Reading, Chapter 4, I introduced some sets of *special-case* quantum wavefunctions. They are *special* because they are solutions of Schrödinger's equation, and such solutions correspond to *states of definite energy*. These special-case wavefunctions are analogous to the classical normal modes on a string, in that they have simple time dependence. If at time $t = 0$ the wavefunction is just one of these special-case functions, that is,

$$\Psi(x, t = 0) = \psi_n(x), \tag{1}$$

then at a later time the wavefunction is simply

$$\Psi(x, t) = e^{-iE_n t/\hbar}\psi_n(x). \tag{2}$$

Remembering that complex exponentials can be written as a complex combination of sines and cosines we see something that is sort of like the time dependence of classical normal modes:

$$\Psi(x, t) = \cos(-E_n t/\hbar)\,\psi_n(x) - i\sin(-E_n t/\hbar)\,\psi_n(x), \tag{3}$$

except for the fact that there is an 'extra' imaginary term.

In the case of Eq. (2), which is appropriate for a special-case initial wavefunction, the factor with exponential time dependence does not affect the probability density, because

$$\begin{aligned}
\rho(x, t) &= |\Psi(x, t)|^2 \\
&= \Psi(x, t)^*\Psi(x, t) \\
&= e^{iE_n t/\hbar}\psi_n(x)\, e^{-iE_n t/\hbar}\psi_n(x) \qquad (4)\\
&= \psi_n(x)^2, \qquad (5)
\end{aligned}$$

which is independent of time. But if the initial state is a linear combination of special-case wavefunctions, we will see that the probability density does depend on the time, because not all the exponential factors will disappear in the product $\Psi(x, t)^*\,\Psi(x, t)$.

2. Linear combination of particle-in-a-box states

The special-case wavefunctions for a particle in a box are given by

$$\psi_n^{\text{PIB}}(x) = \begin{cases} \sqrt{\frac{2}{L}}\sin\left(\frac{n\pi x}{L}\right) & \text{for } 0 \le x \le L \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

with energies given by

$$E_n^{\text{PIB}} = n^2 \times \frac{h^2}{8mL^2}, \tag{7}$$

In this exercise we will work with dimensionless variables like we did in Exercise #9:

$$\begin{aligned} x' &\equiv \frac{x}{L}, \\ E' &\equiv \frac{E}{\frac{h^2}{8mL^2}}, \quad \text{and} \\ t' &\equiv \frac{t}{\frac{4mL^2}{h\pi}} \end{aligned}$$

In these variables $E' = n^2$, and $e^{-iEt/\hbar} = e^{-i2\pi E't'}$, which means that h is effectively 1 in these variables.

Let's start with the initial state

$$\Psi(x,0) = \frac{1}{\sqrt{2}}\,\psi_1^{\text{PIB}}(x) + \frac{1}{\sqrt{2}}\,\psi_2^{\text{PIB}}(x). \tag{8}$$

- Find your program that animated the motion of a classical string..
- Make a function that calculates special-case wavefunctions for a particle in a box, something like:

```
def psiPIB(n,x):
    return sp.sqrt(2.)*sp.sin(n*sp.pi*x)
```

- Make another function that gives the time-dependent *total* wavefunction:

```
def psiTotal(x,t):
    return (sp.exp(-1j*2*sp.pi*1*t)*psiPIB(1,x)
                + sp.exp(-1j*2*sp.pi*4.*t)*psiPIB(2,x))/sp.sqrt(2.)
```

  Notice that the imaginary unit i is written as `1j`, (with no space between the `1` and the `j`).

- Check to make sure that you can graph $|\Psi(x,t)|^2$ at various times that you choose, like $t = 0$, $t = 0.5$, $t = 1.0$, etc. This will require you to use the `abs()` function of python. As a simple example, the following code

```
z = 3 + 1j*4
abs(z)
```

  will return 5, which is the value of $\sqrt{3^2 + 5^2}$. To get $|z|^2$, we just need to use `abs(z)**2`.

- Animate your plot. (If you don't remember how to do this, just ask.)
- If not already there, add this line in your program within your animation `while` loop:

```
plt.title("t={0:.2f}".format(t))
```

  This will provide a "clock" at the top of your graph.

- Determine the period of the "sloshing" of the probability density, and then the frequency. Compare this with the frequency of light emitted when an electron makes a transition from the $n = 2$ to the $n = 1$ state. Recall that $hf = \Delta E$, and in our variables this becomes $1/T' = \Delta E'$.

3. Linear combination of harmonic oscillator states – I

The first few special-case harmonic oscillator wavefunctions are given by

$$\psi_0^{\text{HO}}(x) = \frac{1}{(a^2\pi)^{1/4}}\, e^{-\frac{1}{2}\frac{x^2}{a^2}} \tag{9}$$

$$\psi_1^{\text{HO}}(x) = \sqrt{2}\,\frac{x}{a}\left[\frac{1}{(a^2\pi)^{1/4}}\, e^{-\frac{1}{2}\frac{x^2}{a^2}}\right] \tag{10}$$

$$\psi_2^{\text{HO}}(x) = \frac{1}{\sqrt{2}}\left(2\frac{x^2}{a^2} - 1\right)\left[\frac{1}{(a^2\pi)^{1/4}}\, e^{-\frac{1}{2}\frac{x^2}{a^2}}\right]. \tag{11}$$

For higher values of n, the wavefunctions are messier, but Python can handle them, as you will see below. The general form for these special-case wavefunctions is

$$\psi_n^{\text{HO}}(x) = \frac{1}{\sqrt{2^n n!}\, a\sqrt[4]{\pi}}\, e^{-\frac{x^2}{2a^2}}\, H_n\left(\frac{x}{a}\right), \tag{12}$$

where $a \equiv \sqrt{\hbar/(m\omega)}$, and the symbol $H_n(x)$ refers to the n^{th} *Hermite polynomial*. The set of Hermite polynomials is extremely well studied — so well studied that there are routines to evaluate them in Python, as well as most other numerical computation packages.

The definite energies for the harmonic oscillator states are given by

$$E_n^{\text{HO}} = \left(n + \frac{1}{2}\right)\hbar\omega. \tag{13}$$

In this exercise we will work with dimensionless variables

$$\begin{aligned} x' &\equiv \frac{x}{a}, \\ E' &\equiv \frac{E}{\hbar\omega}, \quad \text{and} \\ t' &\equiv \frac{t}{T}, \end{aligned}$$

where ω and T are the classical angular frequency and period of the oscillator. In these dimensionless variables, $E' = (n + \frac{1}{2})$, and $e^{-iEt/\hbar} = e^{-i2\pi E' t'}$, which again means that h is effectively 1 in these variables.

Let's start with the initial state

$$\Psi(x,0) = \frac{1}{\sqrt{2}}\,\psi_0^{\text{HO}}(x) + \frac{1}{\sqrt{2}}\,\psi_1^{\text{HO}}(x). \tag{14}$$

- Start a new version of your graphing program.
- At the top of your program include the line

```
from scipy import special
```

- Make a function that calculates special-case wavefunctions for a particle in a box:

```
def psiHO(n,x):
    return special.eval_hermite(n, x)*sp.exp(-x**2/2.)/ \
           sp.sqrt(2**n*special.factorial(n)*sp.sqrt(sp.pi))
```

Note that I have used a \ character to break my `return` statement over two lines to fit onto the page. You may not need to do this in your notebook.

- Make another function that gives the time dependent total wavefunction:

```
def psiTotal(x,t):
    return (exp(-1j*0.5*t)*psiHO(0,x) + exp(-1j*1.5*t)*psiHO(1,x))/sqrt(2.)
```

- Check to make sure that you can graph $\Psi(x,t)$ at various times that you choose, like $t = 0$, $t = 0.5$, $t = 1.0$, etc. **NOTE:** You will have to adjust the range of values include for the variable `x` in order to see the whole wavefunction.
- Animate your plot.
- Determine the period of the "sloshing" of the probability density density, and then the frequency. Compare this with the frequency of light emitted when an electron makes a transition from the $n = 1$ to the $n = 0$ state. (Again, recall that $hf = \Delta E$, and in our variables this becomes $1/T' = \Delta E'$.)
- How does the frequency you found compare with the frequency of the classical analog to your quantum oscillator?

4. Linear combination of harmonic oscillator states – II

Try this combination of harmonic oscillator states:

```
def psiTotal(x,t):
    alpha = 2.
    m = 40
    total = 0

    for n in range(m):
        total += sp.exp(-1j*(0.5+n)*t)*alpha**n*psiHO(n,x)/ \
                 sp.sqrt(sp.special.factorial(n))
    return sp.exp(-alpha**2/2)*total
```

This should be very classical!