PH2255 Course:
Quantum Harmonic Oscillators

Thomas Bass

27 March 2021

Abstract

TODO

1 Hooke's Law and the Quantum Harmonic Oscillator

To begin an analysis of quantum harmonic oscillators, we begin with Hooke's law. While this law is in the classical regime, later it can be proved - with Bohr's correspondence principle - that we can apply the same potentials described by Hooke's spring model to the forces experienced by an atom in equilibrium. Just as Hooke's law describes the force on a mass displaced from its equilibrium, here we use it to describe the potential gained by a quantum object in terms of its angular frequency. By differentiating Hooke's law we can obtain the work done by displacing the mass, which we can then rewrite in terms of angular frequency:

$$W.D. = V(x) = \frac{1}{2}kx^2 = \frac{1}{2}m\omega^2x^2 \tag{1}$$

Then, by substituting this potential into the 1D Schrödinger Equation, we obtain the equation for our quantum harmonic oscillator:

$$\frac{-\hbar^2}{2m}\nabla^2\psi(x) + \frac{1}{2}m\omega_c^2x^2\psi(x) = E\psi(x) \tag{2}$$

To begin to solve this differential equation, we first substitute the displacement x and energy E for dimensionless variables, using the reduced Planck constant:

$$y = \frac{x}{a} = \sqrt{\frac{m\omega_c}{\hbar}} \cdot x \tag{3}$$

$$\varepsilon = \frac{E}{\hbar\omega_c/2} \tag{4}$$

By substituting Equations 3 and 4 into Equation 2, we obtain a homogeneous equation, for which we can solve

$$\nabla^2\psi(y) + (\varepsilon - y^2)\psi(y) = 0 \tag{5}$$

2 Solving Asymptotically and Generally

To solve this equation, we first use a trial solution $\psi(y) = y^n \cdot \exp\left(-y^2/2\right)$ in the asymptotic regime of y. For any finite value of E, and thus ε, as $y \to \pm\infty$, we can see from the trial solution that $\psi(y) \to 0$. In this case, we can differentiate the trial solution for $\psi'(y)$ and $\psi''(y)$, which we substitute into Equation 5:

$$\psi''(y) = [n(n-1)y^{n-2} - (2n+1)y^n + y^{n+2}]e^{-y^2/2} \approx y^{n+2}e^{-y^2/2} = y^2\psi(y) \tag{6}$$

Here, we have assumed all lower-order powers of y to be negligible, as in our asymptotic limit $y^{n+2} \gg y^{n-2}, y^n$. Substituting Equation 6 back into Equation 5, and treating ε as negligible in our limit, we can verify that this trial solution is correct:

$$\psi''(y) - y^2\psi(y) = y^2\psi(y) - y^2\psi(y) = 0 \tag{7}$$

To solve for a general case, we replace our trial solution with $\psi_n(y) = H_n(y) \cdot \exp\left(-y^2/2\right)$, where H_n is a function to be determined. We use the subscript n, as we are now solving for all possible quantum states, not just the asymptotic case. By once again taking the second derivative, we obtain:

$$\psi''(y) = [H'' - 2yH' + H(y^2 - 1)] \cdot e^{-y^2/2} \tag{8}$$

We substitute this into Equation 5, and obtain:

$$\psi''(y) + (\varepsilon - y^2)\psi(y) = [H'' - 2yH' + (\varepsilon - 1)H] \cdot e^{-y^2/2} = 0 \tag{9}$$

We know that the exponential term must always be positive (and is in fact a Gaussian distribution), so we must solve the square-bracketed terms for zero. These are in fact the Hermite polynomials, as defined by:

$$H(y) = \sum_{p=0}^{\infty} a_p y^p = a_0 + a_1 y + a_2 y^2 + ... \tag{10}$$

As these polynomials are now in a computable form, we can rewrite our general solution, converting back to the variable x:

$$\psi(x) = \frac{H_n(x/a)}{\sqrt{2^n n! \sqrt{\pi a^2}}} e^{-\frac{x^2}{2a^2}} \tag{11}$$

This then provides us with a general solution to our quantum harmonic oscillator. Despite the solution still having an "unsolved" function H_n, we can easily compute these Hermite polynomials using Python.

3 Numerical Analysis

In the ground state, $\psi_0(x)$, we expect to see the normal probability distribution for our wave function. The 0^{th} order Hermite polynomial is given as $H_0(y) = 1$, which provides us with a simple wave function:

$$\psi_0(x) = \frac{1}{(\pi a^2)^{1/4}} e^{-x^2/2a^2} \tag{12}$$

For the probability density of the ground state, we take the absolute square of the wave function $p_0(x) = |\psi_0(x)|^2$. By plotting these in python, we can see that this result follows the expected normal distribution. From these graphs, we can see that the ground state wave function $\psi_0(x)$ is an even function.

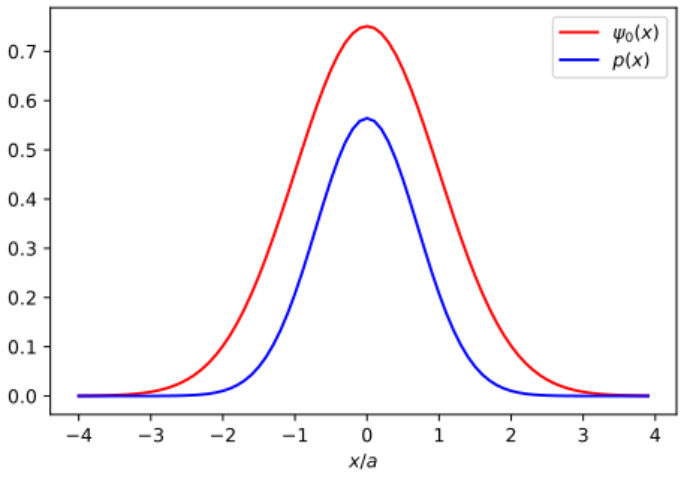

Figure 1: Graph showing the energy of the ground state wave function, and its probability density.

3.1 Orthogonality

To apply the position and momentum operators to the wave function, we must first verify that it is normalised and orthogonal. To verify orthogonality of normalised eigenfunctions, we use the inner product:

$$\langle \psi_m | \psi_n \rangle = \delta_{mn} \tag{13}$$

Where δ_{ij} is the Kronecker delta function:

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases} \tag{14}$$

To verify this, we use the integrate.quad method provided by Python's SciPy library. By constructing a function psi(x, n) in python, analogous to $\psi_n(x)$, we can calculate the inner product as the integral:

$$\langle \psi_0 | \psi_n \rangle = \int_{-\infty}^{\infty} \psi_0^*(x) \psi_n(x) dx \tag{15}$$

For $\psi_n(x)$ wave functions of $n = 1, 3, 4, 7$.

```
for i in [1, 3, 4, 7]:
    inner_product = integrate.quad(np.vectorize(lambda x: psi(x, 0)*psi(x, i)), -float("
        inf"), float("inf"))[0]
```

As the wave functions are not complex, we use $\psi^* = \psi$. This numerical integration method verifies the orthogonality of the ground state wave function. The result for $\langle \psi_m | \psi_4 \rangle$ is calculated as −6.106226635438361e−16, but this is disregarded as a floating-point inaccuracy, as all other values of n return precisely zero.

3.2 Position operator

To calculate the uncertainty of position for the wave function, we employ the position operator $\hat{x} = x$ to calculate the expectation value of the position observable.

$$\langle x \rangle = \langle \psi_0 | \hat{x} | \psi_0 \rangle = \int_{-\infty}^{\infty} \psi_0^*(x)\ x\ \psi_0(x) dx \tag{16}$$

We once again use the integrate.quad method in python:

```
expectation_x = integrate.quad(np.vectorize(lambda x: psi(x, 0)*x*psi(x, 0)), -float("inf"
    ), float("inf"))[0]
```

This returns the value $\langle \psi_0 | \hat{x} | \psi_0 \rangle = 0$, as we expected to obtain from the even parity of the probability density $|\psi_0(x)|^*$. To find the position-squared expectation value, $\langle x^2 \rangle$, we use a similar same code snippet, replacing x with x**2, from which we obtain the value 0.5000000000000012. Again, we round this to 0.5 to account for floating point inaccuracy . By varying our value of a, we obtain the following values for $\langle x^2 \rangle$.

a	$\langle x \rangle^2$
1	0.5000000000000012
2	0.25000000000000056
3	0.16666666666666707

Table 1: Values of $\langle x \rangle^2$ calculated using SciPy's numerical integration function.

From these values, we can readily see that the expected value $\langle \psi_0 | \hat{x}^2 | \psi_0 \rangle = a^2/2$ is obtained.

3.3 Momentum Operator

To calculate the momentum uncertainties, we now employ the momentum operator $\hat{p} = -i\hbar\nabla$ to calculate the momentum expectation value $\langle p \rangle$.

$$\langle x \rangle = \langle \psi_0 | \hat{p} | \psi_0 \rangle = -i\hbar \int_{-\infty}^{\infty} \psi_0^*(x)\ \frac{\partial}{\partial x} \psi_0(x) dx \tag{17}$$

This differentiation is trivial, as the wave function is an exponential:

$$\frac{\partial}{\partial x} \psi_0(x) = -\frac{x}{a^2} \psi_0(x) \tag{18}$$

We can then complete the integral in Python, using 1j for i:

```
expectation_p = -1j*hbar**2*integrate.quad(np.vectorize(lambda x: psi(x, 0)* (-x/a**2) *
    psi(x, 0)), -float("inf"), float("inf"))[0]
```

This gives us the expected result of $\langle \psi_0 | \hat{p} | \psi_0 \rangle = 0$. To find the momentum-squared expectation value $\langle p^2 \rangle$, we differentiate the wave function again:

$$\frac{\partial^2}{\partial x^2} \psi_0(x) = -\frac{x^2 - a^2}{a^4} \psi_0(x) \tag{19}$$

We can once again use the inner product:

$$\langle x \rangle = \langle \psi_0 | \hat{p}^2 | \psi_0 \rangle = -\hbar^2 \int_{-\infty}^{\infty} \psi_0^*(x) \, \frac{\partial^2}{\partial x^2} \psi_0(x) dx \tag{20}$$

By modifying the previous code snippet, we can numerically calculate this integral.

```
expectation_p_sq = -hbar**2*integrate.quad(np.vectorize(lambda x: psi(x, 0)* ( (x**2 - a
    **2) / a**4) * psi(x, 0)), -float("inf"), float("inf"))[0]
```

From which we obtain a value of 0.4999999999999988, rounded to 0.5 for floating point accuracy. By once again varying our value of a, we can obtain a formula for $\langle p^2 \rangle$ in terms of a and $\hbar$.

a	$\langle x \rangle^2$
1	0.4999999999999988
2	0.10937499999999994
3	0.03497942386831273

Table 2: Values of $\langle p \rangle^2$ calculated using SciPy's numerical integration function.

From Table 2, we can show that $\langle \psi_0 | \hat{p}^2 | \psi_0 \rangle$ matches the expected equation $\hbar^2 / 2a^2$.