PH2255 Course:
Quantum Harmonic Oscillators

Thomas Bass

27 March 2021

Abstract

The mechanics and phenomena in the quantum realm are often difficult to relate to classical mechanics - wave-particle behaviour and quantum effects are not observed day-to-day. However, Bohr's Correspondence Principle provides us with a way to relate a quantum mechanical system at sufficiently high quantum numbers to its classical counterpart. By using Hooke's classical spring laws for a harmonic oscillator potential to generate wave functions for a quantum system, we can use the inner product laws and observable operators to understand the uncertainties of the wave functions, and compare those to Heisenburg's Uncertainy Principle. We can compare also the probability distributions of a quantum harmonic oscillator (QHO) to a classical simple harmonic oscillator (SHO), by using the correspondence principle with matched energies between the two systems. A quantum harmonic oscillator is one of very few quantum mechanical systems for which we can derive exact analytical solutions, and Python provides us with numerical integration and differentiation methods to verify the orthogonality of these.

1 Hooke's Law and the Quantum Harmonic Oscillator

To begin an analysis of quantum harmonic oscillators (QHO), we begin with Hooke's law. While this law is in the classical regime, later it can be proved[1] that same potentials described by Hooke's spring model can be applied to the forces experienced by an atom in equilibrium. By differentiating Hooke's Law, the work done by displacing the mass is obtained:

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

Then, by substituting this potential into the 1D Schrödinger Equation, the equation for our QHO is obtained:

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

To make the variables easier to work with, the reduced Planck constant is used to substitute the displacement x and energy E for dimensionless variables:

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

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

[1] In Section 5

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

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

2 Hermite Polynomials

To solve the differential equation 5, we use a trial solution $\psi_n(y) = H_n(y) \cdot \exp\left(-y^2/2\right)$, where H_n is a function to be determined. From a brief analysis, it can be seen that the exponential term will survive differentiation, and that it will cancel out with the negative y^2 term. By taking the second derivative, we obtain:

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

Substituting this into Equation 5, we obtain:

$$\psi''(y) + (\varepsilon - y^2)\psi(y) = [H'' - 2yH' + (\varepsilon - 1)H] \cdot e^{-y^2/2} = 0 \tag{7}$$

As the exponential term is always great than zero, the square-bracketed terms must be solved for zero. These are the Hermite polynomials[2], and are defined by:

$$H(y) = \sum_{p=0}^{\infty} a_p y^p = a_0 + a_1 y + a_2 y^2 + ... \tag{8}$$

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

$$\psi(x) = \frac{H_n(x/a)}{\sqrt{2^n n! \sqrt{\pi a^2}}} e^{-\frac{x^2}{2a^2}} \tag{9}$$

This then provides us with a general solution to our QHO, which is asily computed using Python.

3 Ground State 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 a simple wave function:

$$\psi_0(x) = \frac{1}{(\pi a^2)^{1/4}} e^{-x^2/2a^2} \tag{10}$$

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, it is clear that this result follows the expected normal distribution. From Figure 1, we can see that the ground state wave function $\psi_0(x)$ is an even function.

3.1 Orthogonality

To apply the position and momentum operators to the wave function, it must be verified to be normalised and orthogonal. To verify orthogonality of normalised eigenfunctions, we use the inner product, as defined in [Dahmen and Stroh(2003), §3.1.2∼p.25]:

$$\langle\psi_m|\psi_n\rangle = \int_{-\infty}^{\infty} \psi_n^*(x)\psi_m(x)dx = \delta_{mn} \tag{11}$$

[2]Discussed in detail in [Griffiths(2013), §1T1 ∼ p.57]

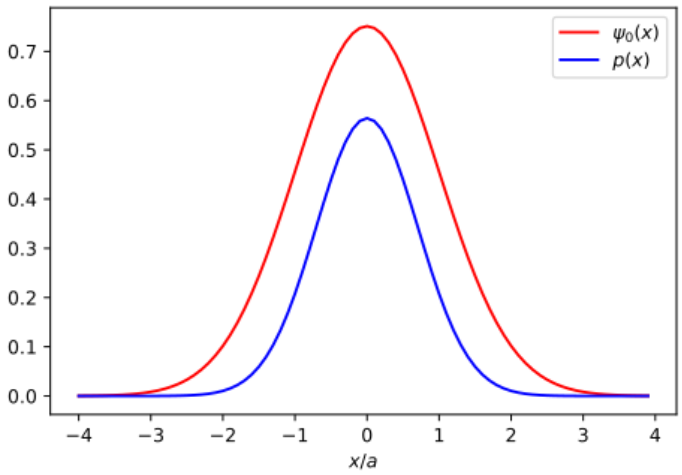


Figure 1: Graph showing the energy of the ground state wave function, and its probability density.

Where δ_{ij} is the Kronecker delta function:

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases} \tag{12}$$

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:

$$\langle \psi_0 | \psi_n \rangle = \int_{-\infty}^{\infty} \psi_0^*(x) \psi_n(x) dx \tag{13}$$

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, the position operator $\hat{x} = x$ are used to calculate the expectation value of the position observable.

$$\langle x \rangle = \langle \psi_0 | \hat{x} | \psi_0 \rangle = \int_{-\infty}^{\infty} \psi_0^*(x)\ x\ \psi_0(x) dx \tag{14}$$

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, $\langle x^2\rangle$, a similar code snippet is used, 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 the values in Table 1 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 $\langle p\rangle$.

$$\langle p\rangle = \langle\psi_0|\hat{p}|\psi_0\rangle = -i\hbar\int_{-\infty}^{\infty}\psi_0^*(x)\,\frac{\partial}{\partial x}\psi_0(x)dx \tag{15}$$

To compute this derivative, another method provided by SciPy is used, `misc.derivative`, assigning a temporary callable variable for $\psi_0(x)$.

```
def psi0(x): return psi(x, 0)

expectation_p = hbar*1j*integrate.quad(np.vectorize(lambda x : psi0(x) * derivative(
    psi0, x, dx=0.0001, n=1)), -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 $\langle p^2\rangle$, the above code snippet is modified to produce the second order derivative, using `n=2`:

$$\langle p^2\rangle = \langle\psi_0|\hat{p}^2|\psi_0\rangle = -\hbar^2\int_{-\infty}^{\infty}\psi_0^*(x)\,\frac{\partial^2}{\partial x^2}\psi_0(x)dx \tag{16}$$

```
expectation_p_sq = -hbar**2*integrate.quad(np.vectorize(lambda x : psi0(x) *
    derivative(psi0, x, dx=0.0001, n=2)), -float("inf"), float("inf"))[0]
```

From which we obtain a value of `0.49999999789637256`, 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 $\hbar$.

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

a	$\langle x \rangle^2$
1	0.49999999789637256
2	0.10937499999999994
3	0.03497942386831273

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

3.4 Calculating Uncertainties

Now that we have calculated the expectation values for position and momentum, the uncertainties Δx and Δp for position and momentum can be calculated. Reference [Mandl(1992), §3.3∼p.123] provides an equation relating the expectation values of a system to the standard deviation of these observables:

$$\Delta A = \sqrt{\langle (A - \langle A \rangle)^2 \rangle} \tag{17}$$

With some simple commutator algebra, the following equations describing the uncertanties for position and momentum are found:

$$(\Delta x)^2 = \langle x^2 \rangle - \langle x \rangle^2 \tag{18}$$

$$(\Delta p)^2 = \langle p^2 \rangle - \langle p \rangle^2 \tag{19}$$

Therefore, by using the expectation values we calculated in Sections 3.2 and 3.3, we can derive the uncertainties Δx and Δp for position and momentum:

$$\Delta x = \sqrt{\langle x^2 \rangle - \langle x \rangle^2} = \sqrt{\frac{a^2}{2} - 0^2} = \frac{a}{\sqrt{2}} \tag{20}$$

$$\Delta p = \sqrt{\langle p^2 \rangle - \langle p \rangle^2} = \sqrt{\frac{\hbar^2}{2a^2} - 0^2} = \frac{\hbar}{\sqrt{2}a} \tag{21}$$

From these, we can easily arrive at the expected expression for the Heisenburg Uncertainy Principle [Griffiths(2013), Eq.63∼p.113]:

$$\Delta x \Delta p = \frac{a}{\sqrt{2}} \cdot \frac{\hbar}{\sqrt{2}a} = \frac{\hbar}{2} \tag{22}$$

4 n>0 Analysis

4.1 Position and Momentum Expectation Values and Uncertainties

To verify these results for wave functions $\psi_n(x)$ above the ground state, we modify the existing code and iterate over wave functions for $n = 0, 1, ..., 10^3$. The results of these calculations are tabulated below.

[3] See Appendix B

n	Δx_n	Δp_n	$\approx \Delta x_n \Delta p_n$
0	0.7071067811865483	0.7071067796990583	1/2
1	1.2247448713915887	1.2247448710189657	3/2
2	1.5811388300841895	1.5811388282180328	5/2
3	1.8708286933869707	1.8708286880893845	7/2
4	2.1213203435596424	2.1213203376391427	9/2
5	2.345207879911715	2.345207872688048	11/2
6	2.5495097567963927	2.5495097437877368	13/2
7	2.738612787525834	2.738612774744414	15/2
8	2.915475947422669	2.915475933768039	17/2
9	3.0822070014845906	3.0822069813612556	19/2
10	3.2403703492043427	3.240370327321197	21/2

Table 3: Values of Δx, Δp, and $\Delta x \Delta p$ calculated using SciPy's numerical integration and differentiation functions.

Table 3 shows that the $\Delta x \Delta p$ changes with (2n+1). From these results, recognising the result from Equation 22 and that we have assigned $\hbar = 1$ to operate in natural units, we can easily derive the expected expressions[4] for Δx, Δp, and $\Delta x \Delta p$ in terms of the order n of the wave function $\psi_n(x)$:

$$\Delta x = \sqrt{\langle x^2 \rangle} = \sqrt{\frac{a^2}{2}(2n+1)} = \sqrt{\frac{\hbar}{2m\omega}(2n+1)} \tag{23}$$

$$\Delta p = \sqrt{\langle p^2 \rangle} = \sqrt{\frac{\hbar^2}{2a^2}(2n+1)} = \sqrt{\frac{\hbar m w}{2}(2n+1)} \tag{24}$$

$$\Delta x \Delta p = \frac{(2n+1)\hbar}{2} \tag{25}$$

4.2 Kinetic and Potential Energy Expectation Values

Similar algebra can be used to determine the expectation values for this system for Kinetic Energy $K.E.$ and Potential Energy V ($P.E.$). The energy for a free particle is given as $E = p^2/2m$, which gives us the energy operator E:

$$\langle K.E. \rangle = \langle \psi_n | \hat{E} | \psi_n \rangle = \langle \psi_n | \frac{\hat{p}^2}{2m} | \psi_n \rangle = \int_{-\infty}^{\infty} \psi^*(x) \frac{(-\hbar)^2}{2m} \frac{\partial^2}{\partial x^2} \psi(x) \, dx \tag{26}$$

We obtain the expression for potential energy all the way from Equation 1:

$$\langle P.E. \rangle = \langle \psi_n | \hat{V} | \psi_n \rangle = \langle \psi_n | \frac{m\omega^2 \hat{x}^2}{2} | \psi_n \rangle = \int_{-\infty}^{\infty} \psi^*(x) \frac{m\omega^2}{2} x^2 \psi(x) \, dx \tag{27}$$

Instead of computing these integrals, the known values for $\langle p \rangle$ and $\langle x \rangle$ (that we found in Equations 24 and 23) are factored out of Equations 28 and 27 respectively.

$$\langle K.E. \rangle = \left\langle \frac{p^2}{2m} \right\rangle = \frac{1}{2m} \langle p^2 \rangle = \frac{1}{2m} \cdot \frac{\hbar m \omega}{2}(2n+1) = \frac{\hbar \omega}{4}(2n+1) \tag{28}$$

[4] [Cappellaro(2012), §9.2 Eq7∼p.83]

$$\langle P.E.\rangle = \left\langle \frac{m\omega^2 x^2}{2} \right\rangle = \frac{m\omega^2}{2} \left\langle x^2 \right\rangle = \frac{m\omega^2}{2} \cdot \frac{\hbar}{2m\omega}(2n+1) = \frac{\hbar\omega}{4}(2n+1) \tag{29}$$

This verifies that the expectation values for the two energies $\langle K.E.\rangle$ and $\langle P.E.\rangle$ are equal, as expected by the equipartition theorem.

5 Bohr's Correspondence Principle

Despite the dissimilarities between classical and quantum phenomena, we can use Bohr's Correspondence Principle[5] to state that a system described by quantum mechanics must reproduce classical physics at sufficiently large quantum numbers (n).

$$p(x)_{\text{quan.}} = |\psi_n(x)|^2 \tag{30}$$

To compare our quantum system with the probability density of a classical harmonic oscillator (SHO), the energies of the two systems must be equated. For our classical system, oscillating with amplitude A:

$$p(x)_{\text{class.}} = \frac{1}{\pi\sqrt{A^2 - x^2}} = \frac{1}{\pi\sqrt{(\frac{2E_n}{m\omega^2})^2 - x^2}} \tag{31}$$

The energy of the SHO must equal that of the QHO, therefore we can derive an expression for energy for both, using the expectation values derived in Equations 28 and 29:

$$E = K.E. + P.E. = 2 \cdot \frac{\hbar\omega}{4}(2n+1) = \hbar\omega(n + \frac{1}{2}) \tag{32}$$

Substituting into the original equation for a SHO, we now have expressions for the probability density of both a QHO and SHO in the same units. At low quantum values of n, as shown in Figure 2 we do not see correspondence, as expected. However, as sufficiently large quantum numbers are approached, Bohr's principle emerges, and the quantum distribution closely follows the classical distribution, as shown in Figure 3.

[5] [Radozycki(2016)]

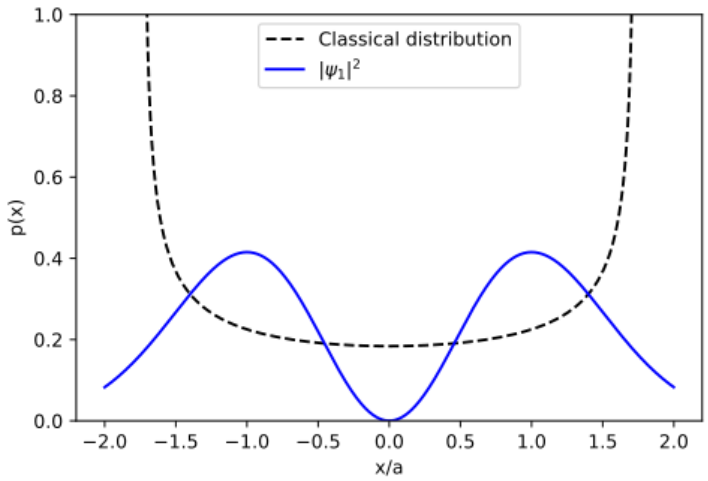
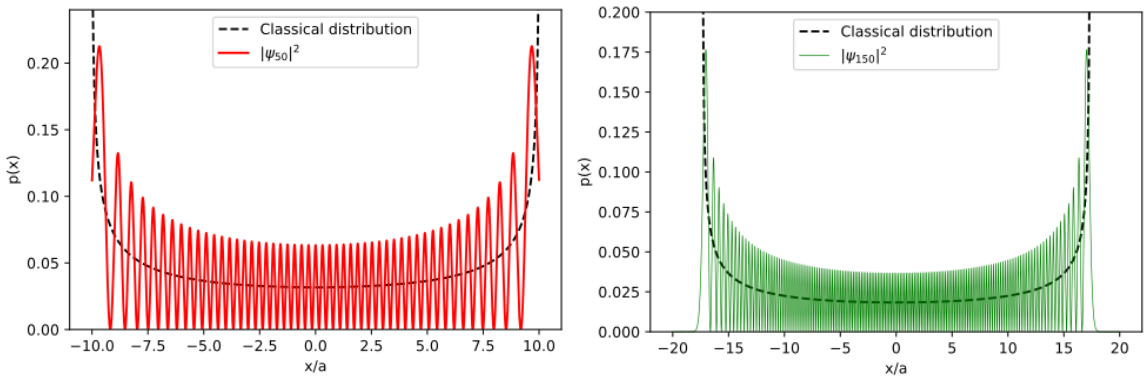


Figure 2: Graph comparing the probability distributions of a SHO and QHO at quantum number $n = 1$.



(a) n=50 (b) n=150

Figure 3: Graphs comparing the probability distributions of a SHO and QHO at quantum numbers $n = 50$ and $n = 150$.

From these graphs, we can conclude that the classical harmonic oscillator approximates a quantum harmonic oscillator at sufficiently large quantum numbers, as described by Bohr's correspondence principle.

A Bibliography

References

[Cappellaro(2012)] Paola Cappellaro. Chapter 9: Harmonic oscillator. In *Quantum Theory of Radiation Interactions—MIT Course No. 22.51*. 2012. URL `https://ocw.mit.edu/courses/nuclear-engineering/22-51-quantum-theory-of-radiation-interactions-fall-2012/lecture-notes/MIT22_51F12_Ch9.pdf`. License: CC BY-NC-SA.

[Dahmen and Stroh(2003)] Hans Dieter Dahmen and Tilo Stroh. *Interactive Quantum Mechanics Under Java*. Springer, New York, NY, 2003. ISBN 9781280188206. URL `http://ebookcentral.proquest.com/lib/rhul/detail.action?docID=4975862`. ID: 4975862.

[Griffiths(2013)] David J. Griffiths. *Introduction to Quantum Mechanics: Pearson New International Edition*. Pearson Education Limited, NOIDA, 2013. ISBN 9781292037141. URL `http://ebookcentral.proquest.com/lib/rhul/detail.action?docID=5483644`. ID: 5483644.

[Mandl(1992)] Franz Mandl. *Quantum Mechanics*. John Wiley & Sons, Incorporated, Chicester, 1992. ISBN 9781118728895. URL `http://ebookcentral.proquest.com/lib/rhul/detail.action?docID=1212545`. ID: 1212545.

[Radozycki(2016)] Tomasz Radozycki. Classical probability density distributions with uncertainty relations for ground states of simple non-relativistic quantum-mechanical systems. 2016. doi: 10.1080/00268976.2016.1219409. URL `https://arxiv.org/pdf/1605.08202.pdf`.

B Python Code for Wave Functions

```
import numpy as np
import numpy.polynomial.hermite as Hermite
from scipy import integrate
from scipy.misc import derivative

# Nth order wave function
n = 1
# Use natural units
hbar = 1
m = 1
w = 1
# Define "a" substitution
a = np.sqrt(hbar / (m*w))

# Define Hermite polynomial generator
def hermite(x, n):
    herm_coeffs = np.zeros(n+1)
    herm_coeffs[n] = 1
    return Hermite.hermval(x/a, herm_coeffs)

# Define wave function (split into prefactor for readability)
def psi(x, n):
    prefactor = 1./np.sqrt(2.**n * np.math.factorial(n) * np.sqrt(np.pi*a**2))
    return prefactor * np.exp(- x**2 / (2*a**2)) * hermite(x,n)

## Code to generate expectation values (Section 4.1)
# Iterate over quantum states 0 to 10
```

```
for order in range(0, 11):
    # Define callable for each quantum state
    def psi_n(x): return psi(x, order)
    # Position expectation values
    ex_x = integrate.quad(np.vectorize(lambda x: psi(x, order)*x*psi(x, order)), -
        float("inf"), float("inf"))[0]
    ex_x2 = integrate.quad(np.vectorize(lambda x: psi(x, order)*x**2*psi(x, order)),
        -float("inf"), float("inf"))[0]
    # Momentum expectation values
    ex_p = -hbar*1j*integrate.quad(np.vectorize(lambda x : psi_n(x) * derivative(
        psi_n, x, dx=0.0001, n=1)), -float("inf"), float("inf"))[0]
    ex_p2 = -hbar**2*integrate.quad(np.vectorize(lambda x : psi_n(x) * derivative(
        psi_n, x, dx=0.0001, n=2)), -float("inf"), float("inf"))[0]
    print(f"------\torder: {order}\t------")
    # Uncertanties
    Dx = np.math.sqrt(ex_x2 - ex_x**2)
    Dp = np.math.sqrt(ex_p2 - abs(ex_p)**2)
    DxDp = Dx*Dp
    print(f"Dx\t{Dx}\nDp\t{Dp}")
    print(f"DxDp\t{DxDp}")
```

py.py