# PHY407 Lab-03 Report

Patrick Sandoval[1,2] and Chin Chong Leong[1,3]

[1] *Q1 - Numerical Differentiation Errors*
[2] *Q2 - Period of Relativistic Particle on a Spring*
[3] *Q3 - Calculating Quantum Mechanical Observable*

## 1. Q1 - NUMERICAL DIFFERENTIATION ERRORS

### 1.1. *Q1a - Forward Difference Differentiation Scheme*

For the pseudo-code for this and all is consecutive sub-questions please refer to the python script `Lab03_Q1.py`. All the function that were used to develop differentiation schemes are in the python script labeled `MyFunctions.py`.

In this subsection we will be looking at the numerical error for the forward difference differentiation scheme applied to the error function. We will take the first derivative of the error function, evaluate it at $x = 0.5$ and compare our forward difference routine to the analytical result

$$f(x) = e^{-x^2} \tag{1}$$

$$f'(x) = -2xe^{-x^2} \tag{2}$$

$$f'(0.5) = -0.77880 \tag{3}$$

We recall that the forward differentiation scheme requires some small forward step $h$ along the x-domain in order to estimate the function's derivative, of which the implementation in python will be according to the formula (ignoring higher-order-term):

$$f'(x) = \frac{f(x+h) - f(x)}{h} \tag{4}$$

Therefore, we will test out various values for $h$ and compute the absolute value of the numerical error of each derivative calculation.

| h | f'(x) | error |
|---|---|---|
| 1e-16 | -1.11022 | 0.3314 |
| 1e-15 | -0.77716 | 0.001645 |
| 1e-14 | -0.77716 | 0.001645 |
| 1e-13 | -0.77938 | 0.0005758 |
| 1e-12 | -0.77882 | 2.067e-05 |
| 1e-11 | -0.7788 | 1.536e-06 |
| 1e-10 | -0.7788 | 6.847e-07 |
| 1e-09 | -0.7788 | 1.855e-08 |
| 1e-08 | -0.7788 | 7.453e-09 |
| 1e-07 | -0.7788 | 3.854e-08 |
| 1e-06 | -0.7788 | 3.894e-07 |
| 1e-05 | -0.7788 | 3.894e-06 |
| 0.0001 | -0.77884 | 3.893e-05 |
| 0.001 | -0.77919 | 0.0003888 |
| 0.01 | -0.78263 | 0.003829 |
| 0.1 | -0.81124 | 0.03244 |
| 1 | -0.6734 | 0.1054 |

**Table 1**: Results of forward differentiation scheme for $10^{-16} \leq h \leq 1$ where middle column is the numerical value of the derivative of eq.1 evaluated at $x = 0.5$ and right-most column is the absolute value of the error for such computation.

### 1.2. *Q1b - Relation Between Error and Step Size*

From the results shown in table 1 we see that choosing the smallest step size does not lead to a more accurate computation for the differentiation of a function. Therefore, we will plot the absolute numerical error and observer where this error reached a minimum as a function of the step size.

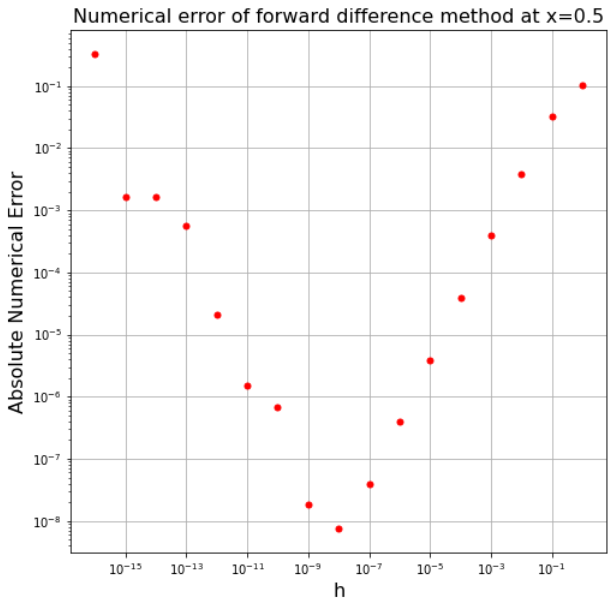


**Figure 1**: Numerical error of forward difference differentiation routine as a function of step size $h$. The optimal step size for the forward difference scheme is around $h = 10^{-8}$

From figure 1 we see that numerical error has a minimum at $h = 10^{-8}$ which is what we would expect for a forward differentiation scheme due to how the error behaves as a function of the step size.

$$\epsilon = \frac{2C|f(x)|}{h} + \frac{1}{2}h|f''(x)| \tag{5}$$

So from eq.5 we see that if the step size to too small i.e $10^{-16}$ then the first term in the RHS will increase drastically causing the error to increase. And if the step size is too big i.e in the order of unity then the second term in the RHS of eq.5 will cause the numerical error to increase drastically. Having these factors into consideration we can say that the happy middle ($10^{-8}$) in between $10^{-16}$ and 1 will be the optimal step size for given routine. Additionally, we can also comment on the fact that rounding error dominates for step sizes smaller than $10^{-8}$ while truncation errors dominate for step sizes greater than $10^{-8}$.

### 1.3. *Central Difference Differentiation Routine*

In this subsection we will repeat a similar analysis from subsection 1.2 where we plot the numerical error of the routines as a function of the step size $h$. Here we will overlay the plot for the central difference routine on top of the forward difference routine, in which the central difference differentiation is implemented in python according to the following formula (ignoring higher-order-term):

$$f'(x) = \frac{f(x + \frac{h}{2}) - f(x - \frac{h}{2})}{h} \tag{6}$$

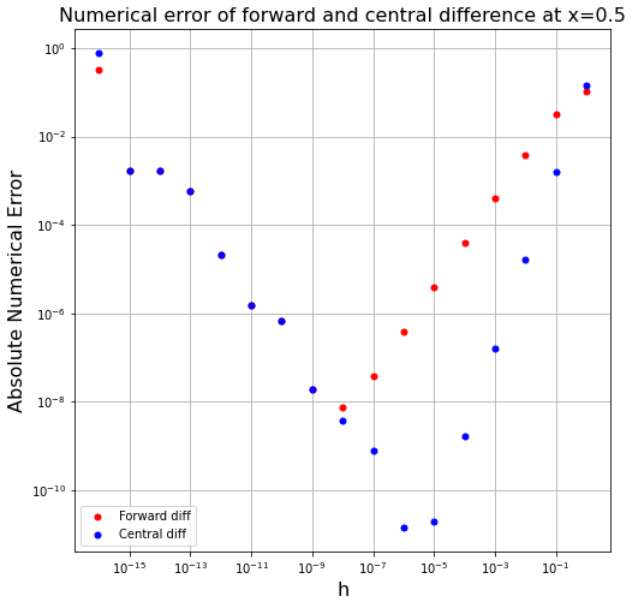


**Figure 2**: Numerical error of central difference differentiation routine as a function of step size $h$. The optimal step size for the central difference scheme is around $h = 10^{-6}$

From figure 2 we see that the central error has a lower minimum numerical error at step size $h = 10^{-6}$. Additionally we also see some y-offset from the forward difference points, for the errors with $h > 10^{-6}$. This indicates that for the same step size the central difference routine has better accuracy then the forward difference routine. Although we may want to conclude that the central difference scheme is always better we need to consider if our data was evenly sampled. If this was the case then central difference would not choose the data points from $x - h/2$ to $x + h/2$ because there are no data points in those regions, instead the central difference would have to choose $x - h$ to $x + h$ so now the step size is $2h$. If we replace $h$ with $2h$ in the truncation error equation we get the following.

$$\epsilon = \frac{1}{2}h^2|f'''(x)| \tag{7}$$

From eq.7 we see now that the truncation error is comparable to the truncation error of the forward difference scheme. Therefore we can calculate the minimum step size for the forward difference to become a better approximation than the central difference.

$$h < \frac{f''(x)}{f'''(x)} \tag{8}$$

So if the step size is any larger than the RHS of eq.8 then the forward difference becomes a better approximation then the central difference for evenly sampled data points.

## 2. Q2 - PERIOD OF RELATIVISTIC PARTICLE ON A SPRING

### 2.1. *Q2a - Classical Limit Period*

For the code and pseudo-code for each sub-question please refer to the python script `Lab03_Q2.py`. Additionally, `gaussxw` and `gaussxwab` function were written down in the python script `MyFunctions.py`.

For some relativistic particle on a spring whose energy is conserved it starts from rest and at some initial position $x_0$ we can express its velocity at some $x$ as follows.

$$v(x) = c\sqrt{\frac{k(x_0^2 - x^2)[2mc^2 + k(x_0^2 - x^2)/2]}{2[mc^2 + k(x_0^2 - x^2)/2]^2}} \tag{9}$$

Where $k$ is the spring's constant, $m$ is the mass of the particle and $c$ is the speed of light in vacuum. In the classical limit where we are dealing with small amplitudes i.e $k(x_0^2 - x^2)/2 \ll mc^2$ we find that the velocity is approximately given by

$$v \approx \sqrt{k(x_0^2 - x^2)} \tag{10}$$

Which is what he expect for a classical pendulum when its energy is conserved. In the relativistic limit where we observe large amplitudes i.e $k(x_0^2 - x^2)/2 \gg mc^2$ we see that $v$ approaches $c$ but remains below it. We can calculate the period of oscillation for this system by noting that the period is the 4times the time it takes the particle to travel from $x_0$ to 0.

$$T = 4\int_0^{x_0} \frac{dx'}{v(x')} \tag{11}$$

In the case for a small enough amplitude of $x_0 = 1\text{cm}$ we can expect the particle to behave classically and if we set $k = 12\,\text{N/m}$ and $m = 1\,\text{kg}$ we get that the period is

$$T = 2\pi\sqrt{\frac{m}{k}} = 1.8137\,\text{s} \tag{12}$$

We can verify the equality in eq.12 by integrating eq.11 through an implementation of Gaussian quadrature for $N = 8$ and $N = 16$ samples. The results of such implementation are the following

$$T_8 = 1.7301\,\text{s} \qquad T_{16} = 1.7707\,\text{s} \tag{13}$$

Where their respective fractional errors are

$$\Delta_8 = 0.048 \qquad \Delta_{16} = 0.024 \tag{14}$$

### 2.2. *Q2b - Gaussian Quadrature on Period Estimation*

As previously discussed in subsection 2.1 we used Gaussian quadrature to integrate eq.11. Here we will look at how the integrand in eq.11 behaves as a function of the number of samples. Additionally we will also look at the weighted integrands to see how they weights change the original plot.

From the right panel of figure 3 we notice the how the number of sample points plays a crucial role on how the weights scale $4/g_i$. We see that when there is less sample points the weights scale the function more drastically as opposed when there is more sample points the weights allow for a smoother curve which allows for higher precision since the weights can adjust themselves in a more precise manner. So that implies that more sample points would allow the weights to fine tune themselves better than weights with lower sample points.

### 2.3. *Q2c - Critical Displacement*

The equation of motion for a 1-dimensional classical particle in a spring (harmonic oscillator) is given by eq.15 which is easily solved by the superposition of two sinusoids.

$$m\frac{d^2x}{dt^2} = -kx \tag{15}$$

$$\implies \quad x(t) = A\sin\left(\sqrt{k/m}t\right) + B\cos\left(\sqrt{k/m}t\right) \tag{16}$$

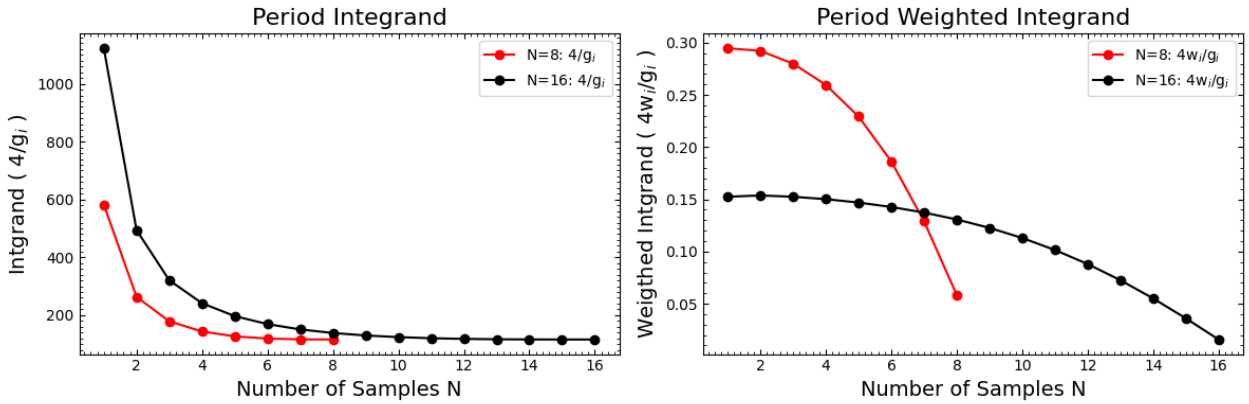


**Figure 3**: (Left panel) Integrand from eq.11, from Gaussian quadrature routines with N=8 and 16 samples. (Right panel) Weighted integrand from eq.11 where the sum all points result in the estimated period. Red and black curves represent the results for N=8 and N=16 points respectively.

If we can find the value of the constants $A$ and $B$ by ensuring our solutions satisfies the inital conditions $x(0) = x_0$ and $\dot{x}(0) = 0$

$$\begin{aligned} x(0) &= A = x_0 \\ \dot{x}(0) &= \sqrt{k/m}B = 0 \\ \therefore \quad x(t) &= x_0 \sin\Big(\sqrt{k/m}t\Big) \end{aligned}$$

Now if we want to set the condition for the velocity at $x = 0$ to be equal to the speed of light we need to calculate the time it takes for the particle to reach $x = 0$.

$$\begin{aligned} & x(t) = x_0 \cos\Big(\sqrt{k/m}t\Big) = 0 \\ \implies \quad & t' = (2n+1)\frac{\pi}{2}\sqrt{\frac{m}{k}} \\ \text{let n} = 0 \quad & v(t') = -\sqrt{\frac{k}{m}}x_0 = c \\ \implies \quad & |x_0| = c\sqrt{\frac{m}{k}} \end{aligned}$$

We only care about the magnitude of the initial displacement because the sign only provides us information about the direction of the velocity which does not play a significant role for us.

### 2.4. *Q2d - Increasing Sample Size for Small Amplitudes*

Here we will quickly estimate the period for the small amplitude oscillator and calculate its percentage error for a Gaussian quadrature routine with $N = 200$ samples.

$$T_{200} = 1.8102 \qquad \Delta_{200} = 0.0019 \tag{17}$$

Where the percentage error is calculated through the following expression.

$$\Delta = \frac{x_{obs} - x_{true}}{x_{obs}} \tag{18}$$

### 2.5. *Q2e - Period and Amplitude Relation*

In this section we will see how the period of the particle transitions from its classical limit to its relativistic limit. We note that in the relativistic limit the period of oscillation approaches the following quantity

$$T = \frac{4x_0}{c} \tag{19}$$

In note that in order to see how the period evolves with time we need to redefine the boundaries of the integral in eq.11 therefore we will use *gaussxw* for $N = 500$ once and the adjust $x_i$ and $w_i$ with the dynamic bounds in order to make our code more computationally efficient.

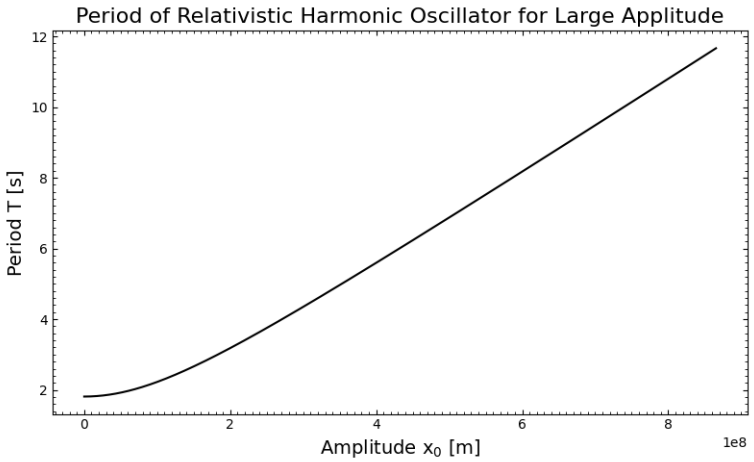


**Figure 4**: Relativistic particle's period relationship with various amplitudes ranging from $x_0 = 1\,\mathrm{m}$ to $x_0 = 10x_c$, where $x_c$ is the critical displacement calculated in subsection 2.3

From figure 4 we see that for small amplitudes the period is approximately constant and independent of the initial displacement as shown in eq.12. However, as the amplitude starts increasing the period increases linearly as well as suggested by the relationship in eq.19 i.e $T \propto x_0$ and with an approximate slope of $4/c$.

## 3. Q3 - CALCULATING QUANTUM MECHANICAL OBSERVABLES

For the code and pseudo-code of this question please refer to the python script `Lab03_Q3.py`. Additionally, the `gaussxwab` function were written down in our python script `MyFunctions.py` and used in this part.

### 3.1. *Q3a - Harmonic Oscillator Wavefunctions for n=0,1,2,3*

For this part of the exercise, we wrote our own defined function in python for the Hermite Polynomials `H(n,x)` in order to find the $n$-th Hermite Polynomial $H_n(x)$; where the Hermite Polynomials take the form of:

$$H_{n+1}(x) = 2xH_n - 2nH_{n-1}(x) \tag{20}$$

and have the base cases:

$$H_0 = 1 \tag{21}$$

$$H_1 = 2x \tag{22}$$

Our defined function also included an additional optional parameter that in addition also returns the $(n-1$-th Hermite Polynomial (so it returns both $H_n(x)$ and $H_{n-1}(x))$, which will be useful for calculating the first derivative of Hermite Polynomials later in section 3.3.

With the Hermite Polynomials calculated, the $n$-th wavefunction of the quantum harmonic oscillator can be calculated using the following equation:

$$\Psi_n(x) = \frac{1}{\sqrt{2^n n!\sqrt{n}}} e^{-(\frac{x^2}{2})} H_n(x) \tag{23}$$

where $H_n(x)$ again is the $n$-th Hermite Polynomial. Using this equation the harmonic oscillator wavefunctions for $n \in 0, 1, 2, 3$ is plotted in Figure 5.

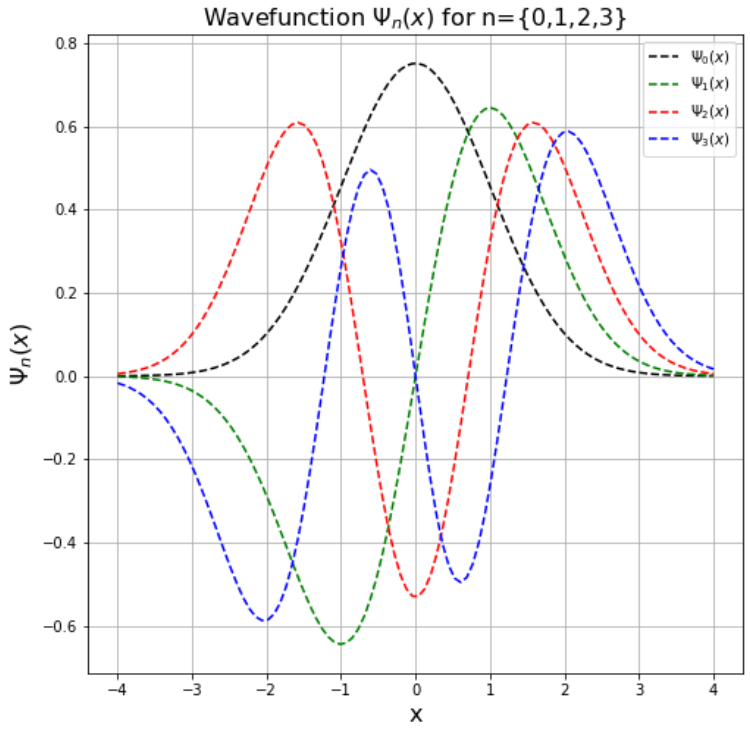


**Figure 5**: Harmonic oscillator wavefunctions at various energy level in the domain $x \in [-4, 4]$: Black line represents $\Psi_0(x)$, green line represents $\Psi_1(x)$, red line represents $\Psi_2(x)$, and blue line represents $\Psi_3(x)$.

### 3.2. *Q3b - Harmonic Oscillator Wavefunctions for n=30*

Using the same functions defined in the previous section, we plotted the wavefunction at $n = 30$ in Figure 6.

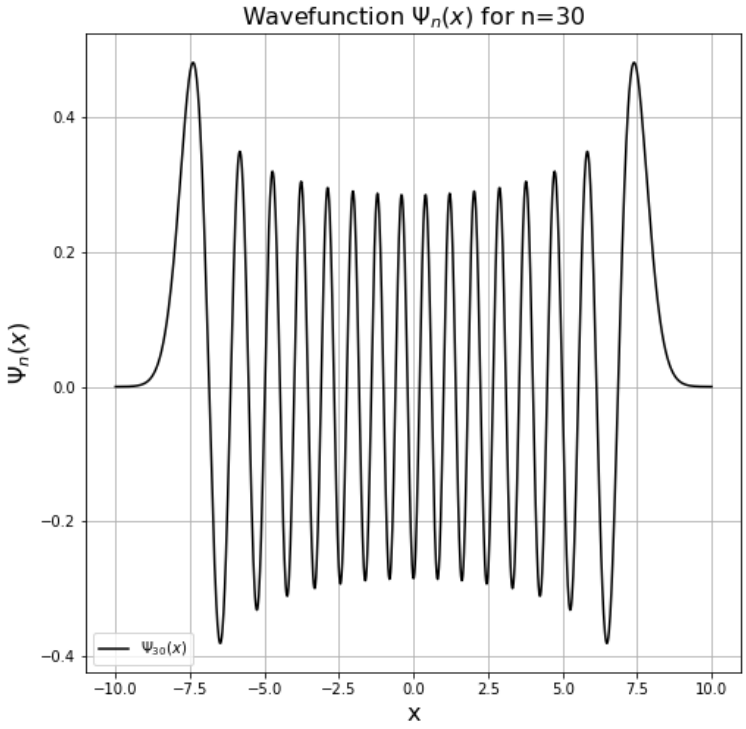


**Figure 6**: Harmonic oscillator wavefunction at $n = 30$ in the domain $x \in [-10, 10]$

### 3.3. *Q3c - $\langle x^2 \rangle$, $\langle p^2 \rangle$, Uncertainties and Energy Calculations*

According to Eq.12-13 in the lab handout, $\langle x^2 \rangle$ and $\langle p^2 \rangle$ are calculated using:

$$\langle x^2 \rangle = \int_{-\infty}^{+\infty} x^2 |\Psi_n(x)|^2 dx \tag{24}$$

$$\langle p^2 \rangle = \int_{-\infty}^{+\infty} |\frac{d\Psi_n(x)}{dx}|^2 dx \tag{25}$$

where

$$\frac{d\Psi_n(x)}{dx} = \frac{1}{\sqrt{2^n n! \sqrt{n}}} e^{-(\frac{x^2}{2})} (-xH_n(x) + 2nH_{n-1}(x)) \tag{26}$$

is the first derivative of the $n$-th wavefunction $\Psi_n(x)$, and the $H_{n-1}(x)$ is the $(n-1)$-th wavefunction that can be obtained by toggling the optional parameter as described in section 3.1.

Notice both $\langle x^2 \rangle$ and $\langle p^2 \rangle$ are double-improper integrals (that integrates from $-\infty$ to $+\infty$), we have therefore used the evaluation technique from Eq. 5.74-5.75 in the textbook, where by change of variables they obtained:

$$\int_{-\infty}^{+\infty} f(x)dx = \int_{-\frac{\pi}{2}}^{+\frac{\pi}{2}} \frac{f(\tan z)}{\cos^2(z)} dz \tag{27}$$

where $f(x)$ here is the integrand of $\langle x^2 \rangle$ and $\langle p^2 \rangle$. To conclude, we obtained:

$$\langle x^2 \rangle = \int_{-\frac{\pi}{2}}^{+\frac{\pi}{2}} \frac{\tan^2 z}{\cos^2 z} |\Psi_n(\tan z)|^2 dz \tag{28}$$

$$\langle p^2\rangle = \int_{-\frac{\pi}{2}}^{+\frac{\pi}{2}} \frac{1}{\cos^2 z} |\frac{d\Psi_n(\tan z)}{dz}|^2 dz \tag{29}$$

We then used the Gaussian quadrature function `gaussxwab` with $N = 100$ points and bounds set to $a = -\frac{\pi}{2}$, $b = \frac{\pi}{2}$ to calculate the domain points and weights; and use them to evaluate the integrals in Eq.28 and Eq.29 similar to the example in pg.180 in the textbook (see code for more detail). Subsequently we calculated the uncertainties of each quantity by taking their square-root (i.e. $\sqrt{\langle x^2\rangle}$ and $\sqrt{\langle p^2\rangle}$), as well as the total energy in the system given by:

$$E = \frac{1}{2}(\langle x^2\rangle + \langle p^2\rangle) \tag{30}$$

The resulting values for each of these quantities for $n \in \{0, 1, ..., 15\}$ are recorded in the following table:

| $n$ value | $\langle x^2\rangle_n$ | $\sqrt{\langle x^2\rangle_n}$ | $\langle p^2\rangle_n$ | $\sqrt{\langle x^2\rangle_n}$ | $E_n$ |
|---|---|---|---|---|---|
| 0 | 0.500 | 0.707 | 0.500 | 0.707 | 0.500 |
| 1 | 1.500 | 1.225 | 1.500 | 1.225 | 1.500 |
| 2 | 2.500 | 1.581 | 2.500 | 1.581 | 2.500 |
| 3 | 3.500 | 1.871 | 3.500 | 1.871 | 3.500 |
| 4 | 4.500 | 2.121 | 4.500 | 2.121 | 4.500 |
| 5 | 5.500 | 2.345 | 5.500 | 2.345 | 5.500 |
| 6 | 6.500 | 2.550 | 6.500 | 2.550 | 6.500 |
| 7 | 7.500 | 2.739 | 7.500 | 2.739 | 7.500 |
| 8 | 8.500 | 2.915 | 8.500 | 2.915 | 8.500 |
| 9 | 9.500 | 3.082 | 9.500 | 3.082 | 9.500 |
| 10 | 10.500 | 3.240 | 10.500 | 3.240 | 10.500 |
| 11 | 11.500 | 3.391 | 11.500 | 3.391 | 11.500 |
| 12 | 12.500 | 3.536 | 12.500 | 3.536 | 12.500 |
| 13 | 13.500 | 3.674 | 13.499 | 3.674 | 13.499 |
| 14 | 14.496 | 3.807 | 14.499 | 3.808 | 14.497 |
| 15 | 15.497 | 3.937 | 15.507 | 3.938 | 15.502 |

**Table 2**: The values of $\langle x^2\rangle_n$, $\sqrt{\langle x^2\rangle_n}$, $\langle p^2\rangle_n$, $\sqrt{\langle x^2\rangle_n}$, and $E_n$ for each $n \in \{0, 1, ..., 15\}$ printed out from python. Notice for $n = 13, 14, 15$, there are significant numerical roundoff error, so for example at $n = 15$, $\langle x^2\rangle_n = 15.497$ should be interpreted as $\langle x^2\rangle_n = 15.5$

From the table we can see that the uncertainties in position and uncertainties in momentum are the same at each $n$ level; and similarly $\langle x^2\rangle_n$, $\langle p^2\rangle_n$, and energy $E$ are the same at each $n$ level. Furthermore, we can observed that at each level $n$, the energy of the oscillator always obey $E_n = n + \frac{1}{2}$.