Quantum Monte Carlo Simulation in Imaginary Time using the Metropolis Method

Toby Kidd
Student ID: 20237635
School of Physics and Astronomy
University of Nottingham

Abstract

This project simulates the quantum harmonic oscillator to find expectation values of the form $\langle x^a \rangle$, where $x = position$ and $a \in \mathbb{Z}^+$. The program from the project generates the expectation values for finite-temperatures. By evaluating the expectation values with error bars for multiple temperatures, a plot of $\langle x^2 \rangle$ against temperature is generated. This plot can then be extrapolated for $T \to 0$, in order to find the ground state energy, E_0, of the quantum harmonic oscillator. The project further goes on to outline how it can be shown that the uncertainty principle dictates that even in the zero-point energy of the quantum harmonic oscillator, there is still uncertainty in $\langle x \rangle$. This can be seen in figure 5 in the results section of this report. This report goes on to outline the various ways in which the program's efficiency can be improved in the future, and how it has been improved beyond the standard function of producing expectation values.

Table of Contents

1 Introduction 3
2 Theory 3
2.1 Principle of Least Action 3
2.2 Path Integral Formulation of Quantum Mechanics 4
2.3 Imaginary Time 4
2.4 Metropolis Quantum Monte Carlo Method 4
2.5 Quantum Harmonic Oscillator 4
2.5.1 Ground State Energy 5
2.5.2 Analytical Results 6
3 Method 7
3.1 Basic Functionality 7
3.2 Checking Agreement with Analytical Results 8
3.3 Improving Efficiency 8
3.4 Zero-Point Energy Position 8
4 Results 9
4.1 Main Findings 9
4.2 Error Analysis 13
5 Discussion 15
5.1 Outline 15
5.2 Limitations 16
5.3 Extensions 17
6 Conclusion 17
7 References 17

1 Introduction

Quantum Monte Carlo methods are used to study complex quantum behaviours. In this project the path integral Monte Carlo method is used to evaluate finite-temperature quantum effects. This project approaches quantum Monte Carlo by using, Metropolis Monte Carlo to evaluate the path integral of a system. This works according to the path integral formulation of quantum mechanics and the action of the system. The aim of this project is to simulate the quantum harmonic oscillator for finite-temperatures, and return expectation values of the form $\langle x^a \rangle$. These expectation values are then compared to analytical results, to ensure that our model is reliably within one standard deviation of analytically expected results. The motivation behind creating a program such as this one is varied. Specifically, path integral Monte Carlo is typically applicable to bosons. One of the first applications of this was in the study of superfluid helium, which helps us understand many important aspects of modern physics: the shape of neutron stars, anisotropic superconductors and oxide high-temperature superconductors. These high-temperature superconductors have many important applications in the industrial world, for example: magnetic levitation (enabling trains to travel at higher speeds with greater efficiency), MRI, synchrotrons and cyclotrons. Another important reason for studying the quantum harmonic oscillator in particular is that, "a one-dimensional potential with a local minimum that may be approximated by the parabolic potential of a harmonic oscillator." [1] This means that other potentials can be approximated as a harmonic oscillator near its local minima.

Furthermore, it should be noted that for this project the Boltzmann constant is set to 1, $k_B = 1$.

2 Theory

2.1 Principle of Least Action

For classical mechanics, the principle of least action states that the path taken by a system is such that the line integral, S, is minimised. Therefore, if we find a curve for which the integral is the least, we can find the motion of a system.

Therefore, the motion of a system from t_1 to t_2 is such that the line integral, S, is minimised,

$$S = \int_{t_1}^{t_2} L dt \tag{2.1}$$

$$S = \int_{t_1}^{t_2} \frac{1}{2} m\dot{x}^2 - \frac{1}{2} m\omega^2 x^2 dt \tag{2.2}$$

where $L = \frac{m}{2}\dot{x}^2 - U(x)$, $m =$ Mass of particle, $\dot{x} = \frac{dx}{dt} =$ Velocity, and $U =$ Potential Energy. Both the kinetic energy and the potential energy are dependent on time, making the calculation more complicated. To solve this problem, one has to find the path in space for which the number is the minimum. It can be assumed that there exists a path for which the action is minimum, which we name the true path, and all other paths are called false paths. All paths must begin and end at the same point. One way of going about this is to try millions of different paths, and find the path with the lowest action, and call it the true path. However, at a minimum in the first order approximation, the action does not change, as the deviation of the function from its minimum value is only second order. Therefore, to solve this problem, one uses the usual procedures of the calculus of variations. This leads to the statement that the action is maximised or minimised along a curve which satisfies the following [2]:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = 0 \tag{2.3}$$

2.2 Path Integral Formulation of Quantum Mechanics

In quantum mechanics, one typically wants to determine the time evolution of the state of a particle. From the classical interpretation, the principle of least action shows that a particle will follow a path which extremises the action. This presents a complication when looking at quantum behaviour, because the path of a particle cannot be known precisely, due to the principle of superposition. The state of the particle can be interpreted as a superposition of many different paths. "The contribution of a path has a phase proportional to the action S:" [2],

$$\phi[x(t)] = const\ e^{\left(\frac{i}{\hbar}\right)S[x(t)]} \tag{2.4}$$

where the action is the same for that in the classical case. Therefore, by summing all of the contributions of each path from a to b, one can obtain the total probability of going from a to b. In our simulation this phase factor is used to determine the probability of accepting or rejecting a path in space, this ensures that a path in space contributes to the overall line integral according to its phase factor. Meaning that, paths in space closer to the minimum of the potential will contribute more, because the change in action is smaller.

2.3 Metropolis Quantum Monte Carlo Method

From the classical interpretation, the aim for solving the time evolution of a system is to find the line integral for which the action is minimised. However, due to the superposition of states in quantum mechanics, the all possible paths in space must be taken into account. Each path has a phase contribution and therefore if we integrate over all possible paths with regards to their phase contribution, we can find expectation values of the system.

2.4 Imaginary Time

Using a transformation from real time, t, to imaginary time τ, so called the Wick rotation,

$$t = -i\tau \tag{2.5}$$

makes the path integral easier to solve.

This change of variables leads to further equations, which will be needed for substitutions later,

$$\frac{dt}{d\tau} = -i \tag{2.6}$$

$$\frac{dx}{dt} = \dot{x} = \frac{dx}{-id\tau} \tag{2.7}$$

2.5 Quantum Harmonic Oscillator

Starting from (Equation 2.2) and applying the change of variables [3] (Equations 2.5, 2.6, 2.7):

$$S = \int_{t_1}^{t_2} \frac{1}{2} m \left(\frac{dx}{dt}\right)^2 - \frac{1}{2} m\omega^2 x^2 dt$$

$$= \int_{-i\tau_1}^{-i\tau_2} \frac{1}{2} m \left(\frac{dx}{-id\tau}\right)^2 - \frac{1}{2} m\omega^2 x^2 (-i) d\tau$$

$$= -i\int_{\tau_1}^{\tau_2}\frac{1}{2}m\dot{x}^2\left(\frac{1}{-i}\right) - \frac{1}{2}m\omega^2x^2(-i)d\tau$$

$$= \int_{\tau_1}^{\tau_2}\frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\omega^2x^2 d\tau \tag{2.8}$$

The variables need to be discretised, in order for the computer to be able to do calculations with them, as there needs to be a finite number of data points. It is possible to create a path in space by discretising time into a one-dimensional lattice with N number of lattice points, with separation, ε. Leading to the following:

$$t_2 - t_1 = N\varepsilon \tag{2.9}$$

Using the trapezoidal rule from calculus to approximate the definite integral using the discretisation outlined above, and the forward difference method, it is shown that,

$$S_N(\underline{x}) = \varepsilon\sum_{j=0}^{N-1}\left[\frac{m}{2}\left(\frac{x_{j+1}-x_j}{\varepsilon}\right)^2 + \frac{1}{2}m\omega^2x_j^2\right] \tag{2.10}$$

Change in action for update on lattice site x_i and $x_i{}'$ is the updated lattice site:

$$\Delta S = \frac{m\varepsilon}{2}\left[\left(\frac{x_i^{'}-x_{i-1}}{\varepsilon}\right)^2 + \left(\frac{x_{i+1}-x_i{}'}{\varepsilon}\right)^2 - \left(\frac{x_i^{'}-x_{i-1}}{\varepsilon}\right)^2 - \left(\frac{x_{i+1}-x_i{}'}{\varepsilon}\right)^2\right] + \frac{1}{2}\varepsilon\left(\left(x_i^{'}\right)^2 - x_i^2\right) \tag{2.11}$$

2.5.1 Ground State Energy

The potential for the quantum harmonic oscillator is given by,

$$V(x) = \frac{1}{2}m\omega^2x^2 \tag{2.12}$$

The expectation value for x^2 is shown to be:

$$\langle n|\hat{x}^2|n\rangle = \frac{\hbar\omega}{m}\left(n+\frac{1}{2}\right) \tag{2.13}$$

$$\langle 0|\hat{x}^2|0\rangle = \langle\hat{x}^2\rangle_{T\to 0} = \frac{\hbar\omega}{2m} \tag{2.14}$$

For our units of $\hbar = \omega = m = 1$,

$$\langle\hat{x}^2\rangle_{T\to 0} = \frac{1}{2} \tag{2.15}$$

Due to the uncertainty principle $\Delta x \Delta p \geq \frac{1}{2}\hbar$, the position and momentum of the particle are uncertain. Therefore, the kinetic and potential energies are uncertain, leading to the following equations [4]:

$$\frac{1}{2}m\omega^2\langle x^2\rangle = \frac{1}{2}E_0 \tag{2.16}$$

$$\frac{\langle p^2\rangle}{2m} = \frac{1}{2}E_0 \tag{2.17}$$

This implies that,

$$E_0 = m\omega^2\langle x^2\rangle \tag{2.18}$$

Using units $\hbar = \omega = m = 1$, it is shown that for our simulation,

$$E_0 = \langle x^2\rangle \tag{2.19}$$

2.5.2 Analytical Results

In order to test the code to be correct and in agreement with solvable equations, we need at least 68.2% of our simulated data points to be within error bars of the expected analytical results. To this end, producing a plot of $\langle x^2\rangle$ against temperature can be used to show this. Firstly, simulating two lattice points gives analytically,

$$\langle\overline{x^2}\rangle = \frac{2T(8mT^2 + m\omega^2T)}{16m^2\omega^2T^2 + m\omega^2} \tag{2.20}$$

For units $\hbar = \omega = m = 1$:

$$\langle\overline{x^2}\rangle = \frac{16T^3 + 2T}{16T^2 + 1} \tag{2.21}$$

Simulating three lattice points gives analytically,

$$\langle\overline{x^2}\rangle = \frac{729m^2T^5 + 108m^2\omega^2T^3 + 3m^2\omega^4T}{729m^3\omega^2T^2 + 54m^3\omega^4T^2 + m^3\omega^6} \tag{2.22}$$

For units $\hbar = \omega = m = 1$:

$$\langle\overline{x^2}\rangle = \frac{729T^5 + 108T^3 + 3T}{729T^2 + 54T^2 + 1} \tag{2.23}$$

It can further be shown that in the purely quantum case, $\langle x^2\rangle_T = expectation\ value\ of\ x^2\ with\ respect\ to\ temperature$, can be calculated for all temperature values, according to the equations below:

$$\langle x^2 \rangle_T = \frac{\sum_{n=0}^{\infty} \langle n|x^2|n\rangle e^{-\frac{E_n}{T}}}{\sum_{n=0}^{\infty} e^{-\frac{E_n}{T}}} = \frac{\sum_{n=0}^{\infty} \frac{\hbar}{m\omega}\left(n+\frac{1}{2}\right) e^{-\frac{\hbar\omega\left(n+\frac{1}{2}\right)}{T}}}{\sum_{n=0}^{\infty} e^{-\frac{\hbar\omega\left(n+\frac{1}{2}\right)}{T}}}$$

$$= \frac{\hbar}{m\omega}\left[\frac{e^{-\frac{\hbar\omega}{T}}}{1-e^{-\frac{\hbar\omega}{T}}} + \frac{1}{2}\right] \tag{2.24}$$

For units $\hbar = \omega = m = 1$:

$$\langle x^2 \rangle_T = \frac{1}{e^{\frac{1}{T}} - 1} + \frac{1}{2} \tag{2.25}$$

This matches the previous calculation for the ground state energy in the limit that $T \to 0$,

$$\langle x^2 \rangle_{T\to 0} = \frac{1}{2} \tag{2.26}$$

3 Method

3.1 Basic Functionality

The simulation created in this project calculates expectation values $\langle x \rangle$ and $\langle x^2 \rangle$ for a given specific finite temperature. Initial conditions must be given for each run of the simulation. For given initial conditions, $m = k = 1$, where $k = m\omega^2$, and given temperature; the number of lattice points of imaginary time, N, must also be specified. The program starts by generating an initial path in space using these initial conditions, where $x = 0 \; \forall \; N \; lattice \; points$. A lattice point is then selected at random, with an equal probability for each. This point's position is then increased or decreased according to a uniform distribution, for values between -1 and 1. The change in action due to this update is then calculated, and accepted or rejected according to the following acceptance probability conditions:

$$p = \begin{cases} e^{-\Delta S}, & if \; \Delta S > 0 \\ 1, & if \; \Delta S \leq 0 \end{cases} \tag{3.1}$$

If the new path is accepted, the mean position value of this path is then stored; otherwise, the old path's mean position value of the path is appended to the existing array of mean positions. The path is then continuously updated, and paths are accepted or rejected over a specified number of iterations. Therefore, at the end of the simulation, an array of the average positions for paths in space are stored. Next, the program takes all of these means of paths in space and finds the overall mean and standard error of these results. However, successive measurements, which are the same, are not statistically independent, and we need statistically independent data points in order to calculate the standard error of our results. To do this, we split the means of the paths returned into tenths, and discard the first tenth. This is because there is a certain time needed to get away from zero, known as the equilibration time. Now we are left with nine blocks of data filled with the means of paths in space. To ensure that the results are statistically independent, we take the average value from each of these blocks and use these averages to calculate the standard error in the data. If we

denote the mean of each block as M_n, where $n = index\ of\ block$ and $N = total\ number\ of\ blocks = 9$. Then the standard error can be calculated using the equation,

$$standard\ error = \sqrt{\frac{\sum_{n=1}^{N}(M_n - \bar{M})^2}{N(N-1)}} = \frac{\sqrt{\sum_{n=1}^{N}(M_n - \bar{M})^2}}{6\sqrt{2}} \tag{3.2}$$

Through this method the expectation value $\langle x^a \rangle$, where $a \epsilon \mathbb{Z}^+$, can be found; with an associated standard error accompanying it.

3.2 Checking Agreement with Analytical Results

To check that the equations listed in (section 2.6.2) are satisfied. The basic code outlined above is run for multiple different temperatures. We set the expectation value to be calculated as $\langle x^2 \rangle$, as$\langle x \rangle_T = 0\ \forall\ T$, and $\langle x^2 \rangle$ is dependent on temperature. These expectation values are then plotted on a graph of $\langle x^2 \rangle$ against temperature, with error bars. The program also returns the percentage of simulated results within error bars of analytically determined expectation values. This is shown to be greater than 68.2%, in figure 3, meaning that the simulated results are within one standard deviation of expected results.

3.3 Improving Efficiency

The efficiency of the simulation can be improved by storing the means of the paths over each iteration instead of storing all the paths and then calculating the means at the end. This means that less data is needed to be stored by the computer. Furthermore, instead of calculating the action of each path in space using (equation 2.10), the change in action can be calculated using (equation 2.11) and does not need to be stored but rather calculated over each iteration. Finally, by updating a sequential set of points, so that multiple points are being updated simultaneously, more updates for paths in space are accepted, particularly for higher temperatures. This is due to the fact that at large temperatures, epsilon becomes very small $\varepsilon = \frac{1}{NT}$. This leads to a smaller action, meaning that a change is more likely to be accepted. Updating a sequential set of points means that for the same number of points updated, less computation for the change in action is needed. This is because when updating multiple sequential points, although the potential energy term of the action is dependent on all points updated, the kinetic energy term is only dependent on the far left and far right lattice points. This is less calculation that performing (equation 2.11) for each iteration.

3.4 Zero-Point Energy Position

The zero-point energy is the lowest possible energy of a quantum system. As a consequence of the uncertainty principle of quantum mechanics, even at the zero-point energy, a system still retains some vibrational motion, due to the fact that precise values of position and momentum cannot be measured simultaneously. The project studies the zero-point energy, by running the simulation for the lowest temperature for which the assumption, $\varepsilon\ is\ small$, holds for a given N lattice points in imaginary time, which is shown in figure 3 in the results section, and then collecting the average position of each path into bins, and then plotting the probability that the position of the particle will be in one of the bins.

4 Results

4.1 Main Findings

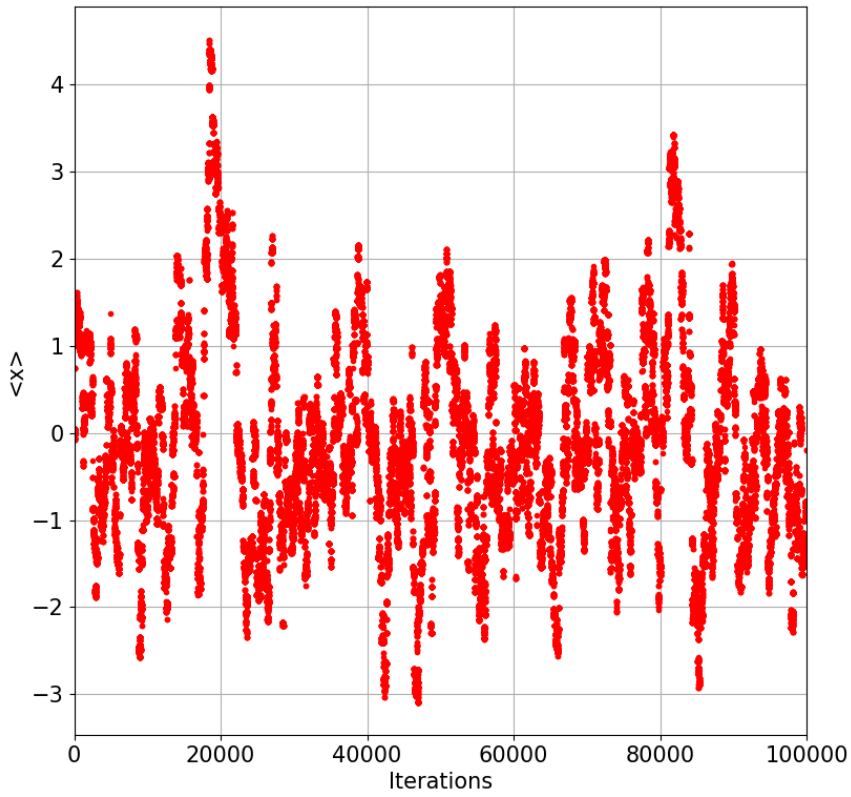

Figure 1: The position of a particle, $\langle x \rangle$, against number of iterations.

Figure 1 was generated by taking the mean $position\ value$ for each path in space generated after each iteration. Figure 1 shows how the Monte Carlo method applied to the quantum harmonic oscillator can be interpreted as the particle taking a random walk up or down the potential with a defined probability in each direction. This probability is determined according to the action, meaning that the particle is most likely to be found near the bottom of the potential where $x = 0$, where the action is a minimum.

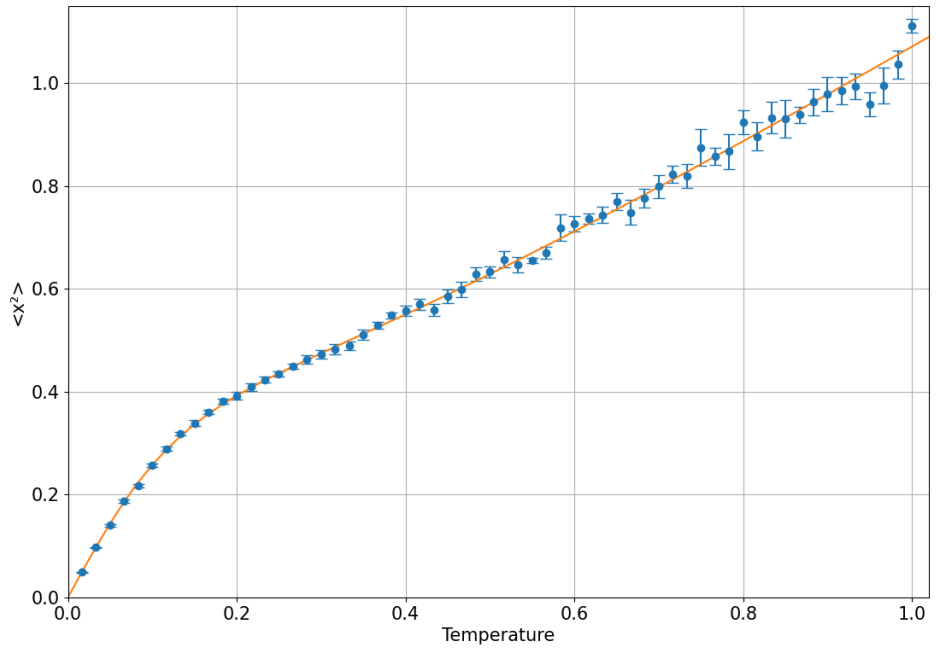

Figure 2: $\langle x^2 \rangle$ against imaginary time for 3 lattice points on imaginary time, with 100,000 iterations for each temperature. Orange expected curve, according to (equation 2.23).

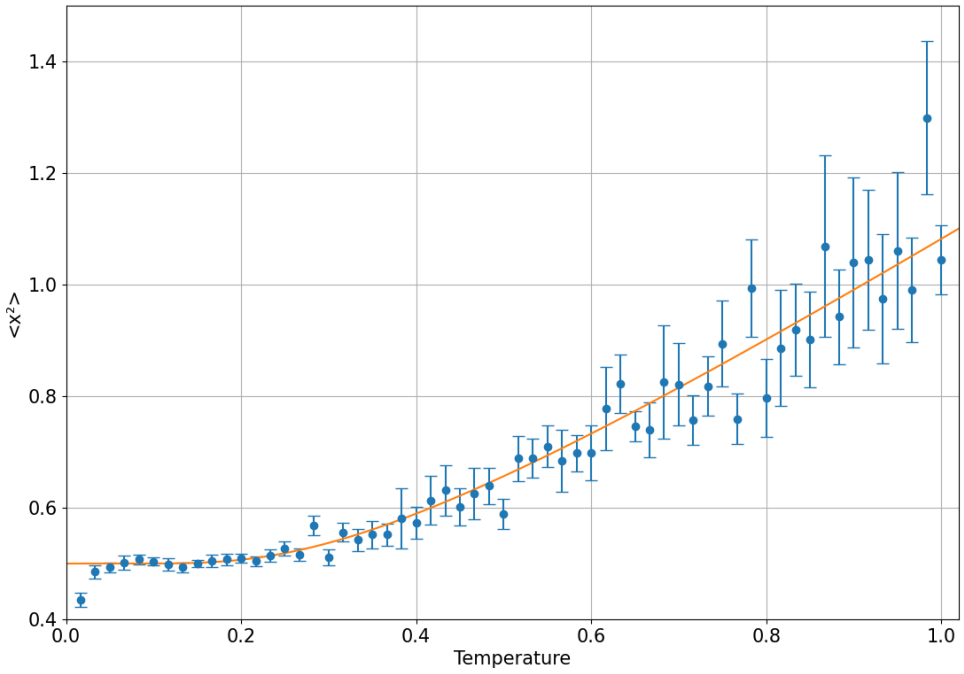

Figure 3: $\langle x^2 \rangle$ against imaginary time for 50 lattice points on imaginary time, with 100,000 iterations for each temperature. Orange expected curve, according to (equation 2.25).

Figures 2 and 3 are generated by using the method outlined in section 3.2. Both plots use different equations for the analytically expected expectation values, as outlined in section 2.6.2. In figure 2, 75.0% of the simulated data points fit within error bars of what is expected analytically. The average error is 0.0145497873233694. In figure 3, 78.333333% of the simulated data points are within error bars. The average error on all data points is 0.049002151360037. Therefore, both are within one standard deviation of the analytically calculated expectation values for $\langle x^2 \rangle$. This means that the program is reliable for calculating $\langle x^2 \rangle$, for finite-temperature. It can further be seen in figure 3 that the simulation breaks down for small temperatures, as the assumption that $\varepsilon = \frac{\beta}{N} = \frac{1}{NT}$ is small no longer holds.

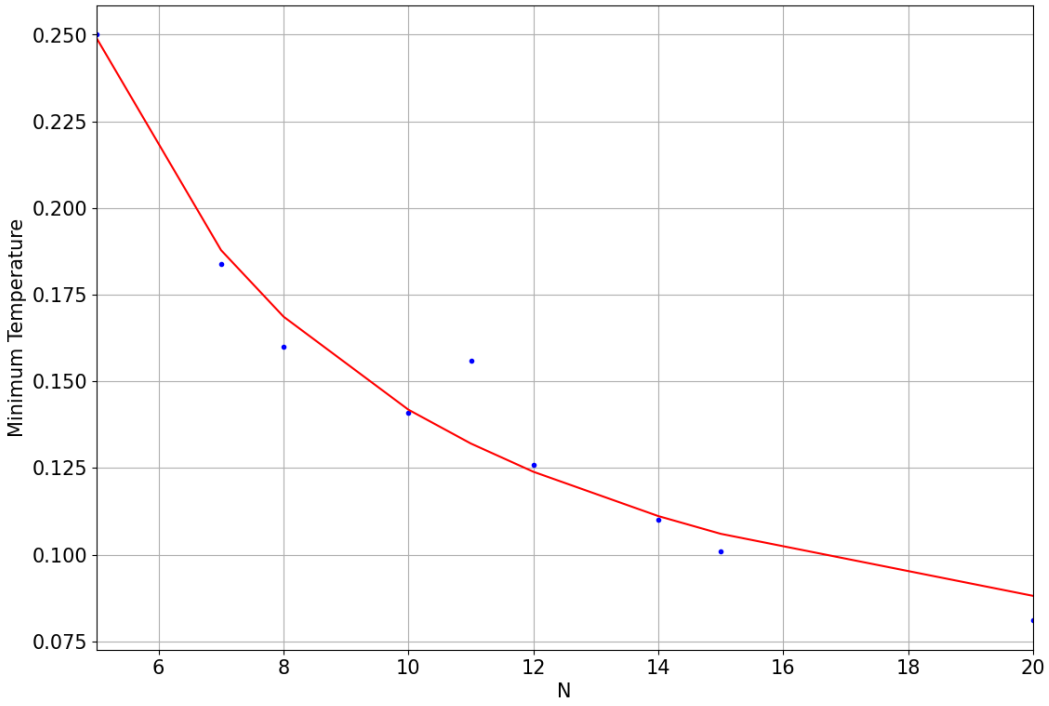

Figure 4: Lowest temperature for which simulation holds $\langle x^2 \rangle = 0.5$. Blue line of best fit with equation $f(x) = \frac{a}{x} + b$. 100,000 iterations

The lowest temperature is shown to follow the best curve fit of: $f(x) = \frac{a}{x} + b \approx \frac{1}{x} + 0.04$. This is important is trying to find the ground-state energy of the quantum harmonic oscillator. As shown in figure 4, the greater the number of lattice points in imaginary time, the smaller the minimum temperature for which the simulation holds. Therefore, this is useful for simulating the ground state expectation values, because the ground state energy can be more accurately predicted for increasing values of N.

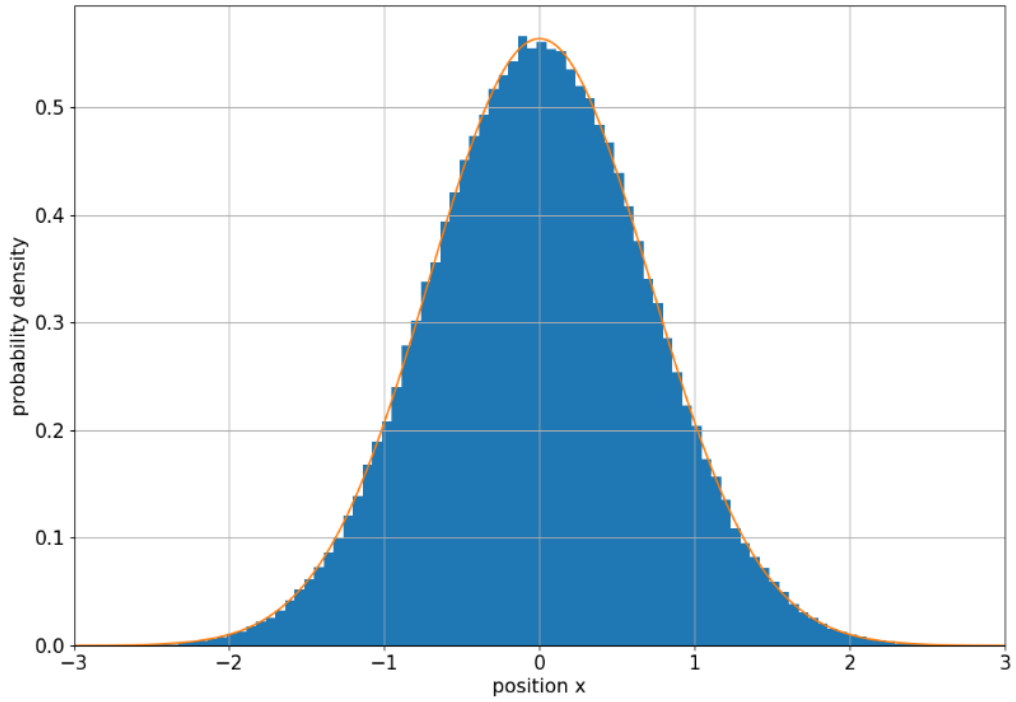

Figure 5: Zero-point energy position probability.

By using the minimum temperature for which the simulation predicts reliable results, we can use this to calculate the ground state energy, $E_0 = \langle x^2 \rangle$. In order to produce a plot for $|\psi_0(x)|^2$ we will need to use the value of the ground state energy. By running the simulation for the energy of the system as being equal to the ground state energy, we can plot the ground state probability density. In detail the simulation produces this plot by first calculating $\langle x^2 \rangle$ for the minimum temperature for which the model holds. We then use this ground state energy value, to simulate the quantum harmonic potential at this energy. This is the same as using $T = 0.5$ in our simulation, due to the constants that we have previously assigned. This produces an array of the means of the paths in space for E_0, for a particular number of iterations. By discarding the first tenth, due to equilibration time, and sorting the remaining paths in space into 100 bins, a histogram for $|\psi_0(x)|^2 \; against \; x$ can be produced. The orange expected curve for the ground state energy is calculated using $|\psi_0(x)|^2 = \frac{1}{a\sqrt{\pi}} e^{-\frac{x^2}{a^2}} \; where \; a = \sqrt{\frac{\hbar}{m\omega}} \; in \; our \; units, a = 1$. Therefore in units $\hbar = m = \omega = 1$,

$$|\psi_0(x)|^2 = \frac{1}{\sqrt{\pi}} e^{-x^2} \tag{4.1}$$

Figure 5 shows excellent agreement with what is expected analytically.

4.2 Error Analysis

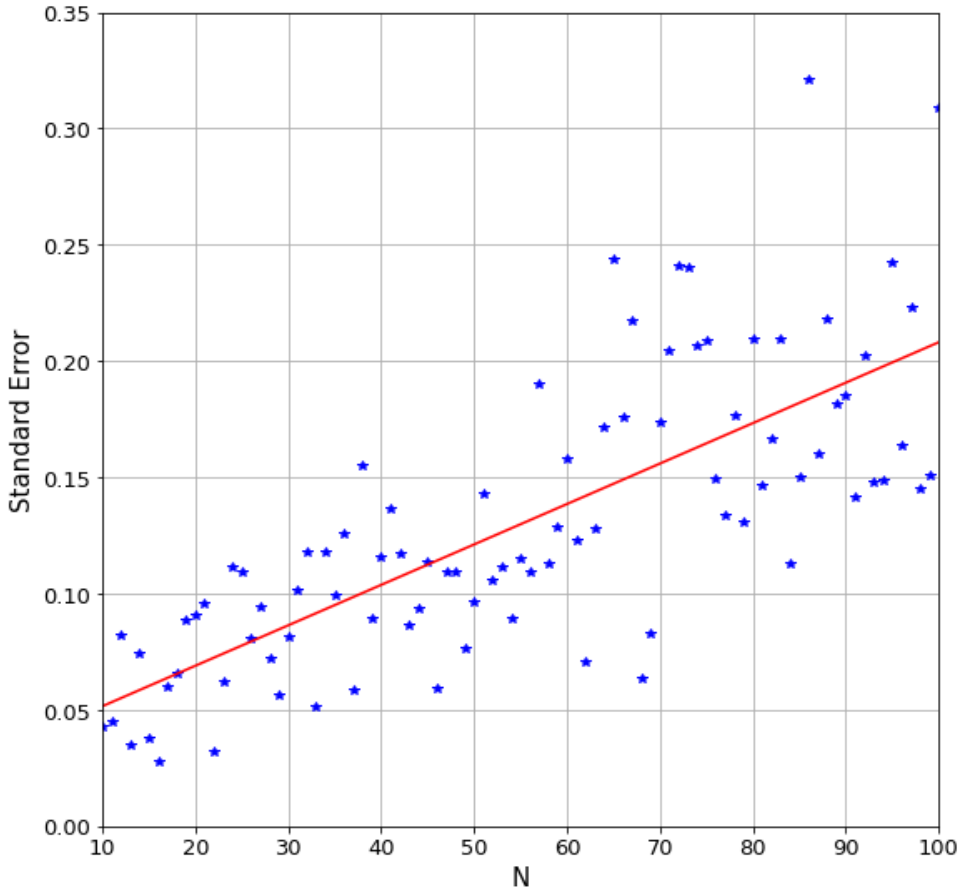

Figure 6: Average error against number of lattice points in discretised imaginary time for 100,000 iterations

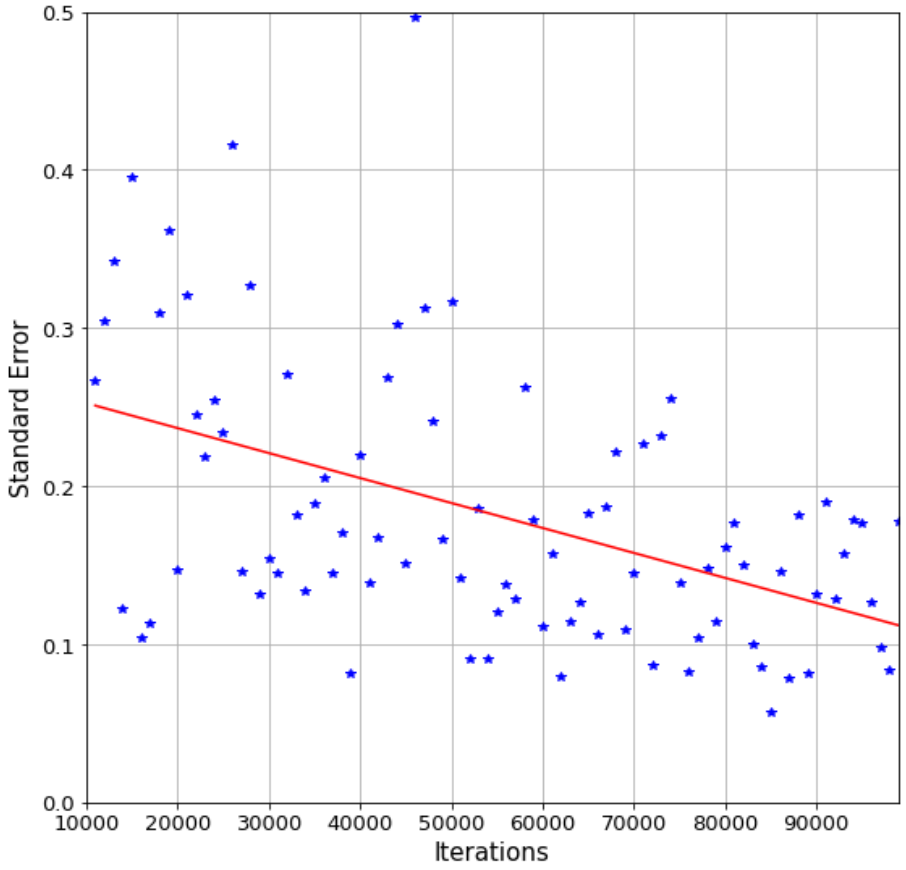

Figure 7: Average error against number of iterations using a curve fit following the form $f(x) = mx + c$

Figure 6, displays how increasing N, increases the standard error on the simulated results for the same number of iterations. Figure 7, displays how by increasing the number of iterations of the system. However, this graph only displays this key fact, it does not determine by what degree each an increase in iterations will decrease the standard error. Combining the ideas present between figures 6 and 7 it is wise to see for how much increase in iterations would be needed to maintain the same standard error on the simulated results. The increase in standard error for each increase in N is approximately 0.00172. The decrease in standard error for each increase in number of iterations is approximately equal to 1.5625×10^{-6}. These means that to offset an rough increase of N by 1, 1102 iterations. Therefore, this could be made into a function for future use, in order to guarantee the same standard error for each value of N, will be able to pre-set the standard error needed.

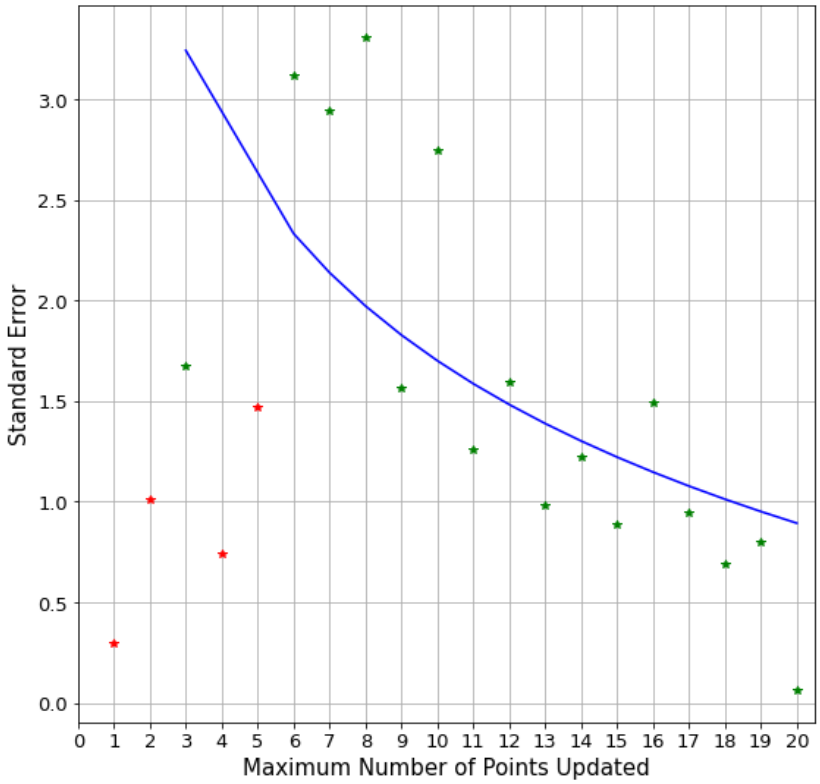

Figure 8: Standard error for $\langle x^2 \rangle$ against maximum number of points updated for 20 lattice points on imaginary

In figure 8, the red points indicate that these simulated results were not within one standard deviation of the expected results. However, for the green points they are. For figure 8, the simulation is run with initial conditions $T = 5$, $N = 20$. The simulation is then run 20 times. Each time the simulation is run a maximum number of points that can be updated over each iteration is selected in integer steps from 1 to 20 inclusive. The first data point generated is for when only one point is allowed to be updated when finding each new path in space for each iteration. This is iterated over 1,000,000 times. The statistically independent standard error is then calculated for these paths in space, and a point is plotted on the graph. Next a maximum of two points are allowed to be updated, with each having an equal chance of happening for each iteration. Following this a maximum of three points can be updated on each iteration, with again each having an equal probability of occurring. This is then repeated for all maximum number of points possible until N. The curve fit used here is of the form $f(x) = \frac{a}{n^{0.1}} + b$, where n is the maximum number of points updated on each iteration. This clearly shows that the standard error decreases with increasing maximum number of points, with a particular reduction for $maximum\ number\ of\ points\ updated = N$. For this reason, $maximum\ number\ of\ points\ updated = N$ in the final code used in the project for simulating expectation values.

5 Discussion

5.1 Outline

From figure 2 and figure 3, it can be seen that expectation values, $\langle x \rangle$ and $\langle x^2 \rangle$, can be found for the quantum harmonic oscillator. This can be calculated for all temperature values above the minimum temperature, as shown in figure 4, which is where the initial assumption that $\varepsilon = \frac{\beta}{N}$ is small no longer holds, for different number of lattice points in imaginary time. The expectation values calculated, are shown to be within one standard deviation of the analytically calculated results. This shows that the program can reliably predict expectation values for the quantum harmonic oscillator. This by

extension, implies that by changing the action, the program should provide reliable results for other potentials. However, other potentials with known analytical results will first need to be run for the program, in order to check that this is the case. For example, one could try the double-well potential $V(x) = 3x^4 - 8x^2$, or use perturbation theory for a small perturbation to the quantum harmonic oscillator.

The program exhibits quantum effects, as shown in figure 5. This is due to the fact that uncertainty in position when temperature tends to 0, is a purely quantum phenomena. In the classical harmonic oscillator, as temperature tends to 0, $\langle x \rangle$ tends to zero.

5.2 Limitations

The program is limited by time, the program has a finite time for which it can be run according the whoever runs the program, and how much data can be stored. To get around this the efficiency of the program can be to decrease the uncertainty for calculations in a specific time frame. There are two ways of looking at this; reducing the uncertainty for the same time, or reducing the time for the same uncertainty. For the former, the number of sequential lattice points that are updated on each iteration can be increased. For the latter, the code can be vectorised, getting rid of any unnecessary for loops and instead replacing them with large arrays, which are allocated to memory at the start of the program, instead of constantly being appended to. Removing any conditional statements would again improve the efficiency of the code, such as checking the change in action. Programming in a different programming language, for example C and Assembler, would further improve the efficiency of the code, as these are lower level languages, and the code can run faster. Storing less data would speed up the code, as less time is needed to allocate memory, so all allocations of memory would all be done at the start of each run.

This leads into the discussion of the shortcomings of this project. The limitations of this project come in two veins; user running the code, and inherent shortcomings of the model used in the program. The user is limited to the amount of time needed for the program to run; if there was an infinite amount of time for the computer to run, there could be an infinite number of iterations over which the code could run, and using a large number of lattice points, for all temperatures above the lowest temperature for which the model holds, the standard errors of each simulated expectation values would approach zero. However, this is not feasible in reality, as problems need to be solved within a reasonable time frame. Furthermore, there are limitations from the computer hardware used. This limits the amount of data that can be stored, and therefore the length of arrays stored, in particular the number of lattice points and number of iterations. Instead of doing this, in the future it would make more sense to sum means of the expectation values of the paths in space, after each iteration, not storing the first tenth of iterations results, and then summing each subsequent tenth after each iteration.

The coding language also limits the speed at which the program runs. This can be improved by importing modules written in C into the python code and running them (e.g. NumPy). However, it would make more sense to write the program in C itself, or even better Assembler, as instructions in Assembler are understood natively by the computer.

Inherent shortcomings in the model used by the program are that discretisation of variables means that the expectation values produced by the program can never be precise. Another shortcoming of the code, is that the number of points updated after each iteration is not a function of temperature, this would make sense to implement as higher temperature values benefit more from updating multiple points at the same time.

5.3 Extensions

For these reasons, the extensions to the project would include changing the distribution of number of points updated in each iteration, from an equal chance of each number, to something that changes dependent on the temperature, in order to improve overall efficiency. Another extension could be to simulate different potentials by modifying the action, and comparing to analytical results. At the moment the simulation works in one spatial dimension, extending this to two spatial dimensions would be the next step, and then three spatial dimensions. A further, more complicated extension to this project would be to carry out a continuous time simulation. This is particularly useful, because at the moment, the lowest temperature for which the simulation holds is highly dependent on the number of lattice points in imaginary time. A continuous time algorithm is the limit of the process of introducing increasingly small imaginary time intervals. Therefore, this method eliminates one of the most profound causes of uncertainty and systematic error within the simulation. Going to the continuum limit in time, is known as the infinite Trotter index. Without this, one would need to run the simulation for an increasing number of lattice points and extrapolate the results to infinity [5]. Extrapolating the results beforehand would be an interesting extension to this project.

6 Conclusion

This project simulates the quantum harmonic oscillator in imaginary time using the Metropolis algorithm. For specific finite temperature values, the simulation generates expectation values of the form, $\langle x^a \rangle, where\ a \in \mathbb{Z}^+$. The program also simulates the ground state energy E_0. The simulated results agree with the analytical results. The results for $\langle x^2 \rangle$ are in excellent agreement with analytical results.

7 References

[1] J., L.A.F. (2006) “6.1,” in Applied Quantum Mechanics. Second. Cambridge: Cambridge University Press, p. 281.

[2] Feynman, R.P., Hibbs, A.R. and Styer, D.F. (2005) Quantum Mechanics and Path Integrals. Emended. Mineola: Dover Publications.

[3] Manoukian, E.B. (2006) Quantum theory. New York: Springer.

[4] Phillips, A.C. (2013) “6.3,” in Introduction to quantum mechanics. Hoboken: Wiley, pp. 115–116.

[5] Landau, D.P. and Binder, K. (2015) “8.3.7,” in A guide to Monte Carlo Simulations in statistical physics. Cambridge, United Kingdom: Cambridge University Press, pp. 310–311.