THE UNIVERSITY *of* EDINBURGH
School of Physics
and Astronomy

Accelerated Tempering Dynamics in HMC Simulations of Lattice Field Theory

MPhys Project Report

Jack Frankland

April 11, 2018

Abstract

This report explains how the Hybrid Monte Carlo algorithm can be used numerically calculate the properties of the quantum harmonic and anharmonic oscillators and discusses the results of a simulation written for this purpose. In the case of the anharmonic oscillator there are isolated modes in the probability distribution and hence the states in the simulation can have difficulty tunnelling between the minima in the double well potential, leading to incorrect estimates. We investigate the effect of introducing a tempering parameter into the Hybrid Monte Carlo simulation, in order to reduce the effect of the tunnelling problem. Tempering is shown to be ineffective for the system in question and we examine why this is case.

Supervisors: Dr Roger Horsley, Dr Brian Pendleton

Personal statement

Since the main component of my work would be writing a simulation, in preparation for my MPhys project I spent several weeks during the summer vacation thoroughly learning the C++ programming language. I choose to use C++ rather than another programming language due to its speed, which proved to be valuable for any intensive computations I needed to perform during my project.

Having had no previous experience with Monte Carlo simulations, I spent the first two weeks of semester one reading introductory articles and exploring the theory behind Markov chain Monte Carlo algorithms. During this time I also began reading papers explaining the connection between the path integral formulation of quantum mechanics and statistical mechanics, and how in this context Monte Carlo methods can be used to compute properties of the quantum system.

As an initial exercise and preliminary test, the first simulation I wrote used the Metropolis algorithm rather than Hybrid Monte Carlo (HMC) algorithm to calculate the path integrals for the harmonic oscillator. Writing the simulation took around two weeks and during this time I also read articles and papers on the HMC algorithm that I would be using in my actual simulation. At this point I was able to begin writing the HMC simulation for the quantum harmonic oscillator. Implementing the HMC algorithm was more complicated than the Metropolis algorithm, and at the advice of my supervisors I included several tests in my simulation to check its validity, overall this took two weeks.

Although at this point I had a working simulation, the properties of the system it could calculate were limited to basic quantities such as position expectation in the ground state. I therefore spent the next three weeks investigating how more complicated quantities such as the ground state density function and energy eigenvalues could be computed from the Monte Carlo simulation. As well as reading the theory behind how these quantities are obtained, I implemented methods within my simulation to actually calculate them, I also learned how to use the plotting tool Gnuplot to plot and fit data which was very useful for analysing results. Checking these results against exact theoretical ones in the case of the discrete quantum harmonic oscillator, enabled me to verify the validity of my calculations in the simulation.

Since I now had a working HMC simulation for the quantum harmonic oscillator, I spent the final two weeks of the semester editing my code to include the option of simulation for the anharmonic oscillator as well; I could reuse most of the methods from the harmonic case.

Over the Christmas vacation period I spent time investigating how in the case of the quantum harmonic oscillator the discrete theory can still be solved exactly, and re-derived the results that I had been checking my simulation against. During this time I also read into the effects on numerical results of correlations between measurements in a Monte Carlo simulation, and how to take this into account with the error analysis.

During the first week of semester two I investigated the effect of varying well depths and distances in my anharmonic oscillator simulation and found example cases when tunnelling

became a problem for the system. I then spent the following two weeks reading papers explaining how tempered dynamics can be incorporated into the HMC algorithm to solve tunnelling like problems, and implemented methods in my simulation to do this. At this point I found the tempering to be ineffective in the system due to the fact it lowered the acceptance rate without increasing the tunnelling of the system.

I spent the following two weeks taking a break from computational work and looking into some more of the theory behind HMC. I devised proofs for volume preservation and then detailed balance for normal HMC, and then modified them for the tempered case.

Returning to my simulation, I spent a further two weeks making edits to my code so I could observe the effect tempering was having on the proposed update states in the simulation. By observing the evolution of the Hamiltonian in these cases I was able to identify the reason why tempered dynamics as applied to our system did not work.

I spent the remainder of the time generating data and plots for both the tempered and non-tempered simulations and then working these into the report and presentation.

I met with my superiors throughout semesters one and two in weekly hour sessions to discuss issues and questions, as well as explain and demonstrate results.

Acknowledgments

I wish to extend my deep appreciation and gratitude to Dr. Roger Horsely and Dr. Brian Pendleton, my MPhys project supervisors, for all their guidance in answering questions as well as providing feedback, advice, support and encouragement throughout the entire project, both in our weekly meetings and through electronic communications.

Contents

1 Introduction

The Hybrid Monte Carlo (HMC) algorithm (also referred to as Hamiltonian Monte Carlo) was originally developed by Duane, Kennedy, Pendleton and Roweth in [1] for the purposes of numerical simulation of lattice field theory. HMC is an example of a Markov Chain Monte Carlo (MCMC) method, where states are proposed via a Markov chain in order to produce samples distributed according to some probability distribution, these samples can then be used to estimate the expectation value of some function of the samples under the probability distribution. Although MCMC was original introduced via the *Metropolis algorithm* in [2], HMC makes improvements on this method by using Hamiltonian dynamics to propose new states in the Markov chain. The idea is that we consider the variables of interest as *position* and introduce auxiliary *momentum* variables, which we can use to define a Hamiltonian and evolve the system in computer time via a leapfrog integration method to propose new states. We alternate between drawing the momentum variables from a multivariate Gaussian distribution and performing a Metropolis update, where the new state is proposed via the Hamiltonian dynamics. The advantage of the HMC method is that the proposed state can have a high probability of acceptance and be distant from the current state, it therefore avoids the problems of slow state exploration that results from using random-walk proposals such as in the original Metropolis algorithm [3]. Although popular amongst the lattice field theory community the HMC method has also been used in statistics, e.g. [4], [5] and [6] for applications such as neural network models. However, our application of HMC will be to quantum mechanical systems and in particular the case of quantum harmonic and anharmonic oscillators.

Standard HMC can have difficulty sampling from the different areas of a probability distribution if those areas are separated by a region of low probability; we will refer to these areas as *isolated modes* of the distribution. An example of this in quantum mechanics is that of a 1-dimensional double well potential, where the probability of finding a particle in either well is high, whilst finding it in the region between the wells is low. If we are interested in generating samples for say the position of the particle using the HMC algorithm, then for a deep enough well there will be an asymmetry in the number of samples recorded in each well. This is due to the energy cost associated with moving between the wells, and so the Hamiltonian dynamics in the HMC simulation will tend to propose an update state in the well the current state is in, leading to the asymmetry in the recorded samples. The aim of this project is to write an HMC simulation of the anharmonic oscillator and introduce a tempering parameter as proposed in [3] and [7] into the dynamics to investigate whether this increases the frequency at which samples move between the two wells. In terms of our simulation the benefit of successful tempering would be more accurate estimations of observed quantities such as position expectation and ground state density functions. However, in terms of a broader goal, successful tempering results in HMC simulations of simple quantum mechanical systems, may suggest it has potential applications in lattice field theory where isolated modes are a more serious problem. In certain field theories, including QCD, the states we wish to construct via our Markov chain are in "distinct topological sectors which are labelled by a topological charge"[8]. For small lattice spacings moving between these sectors can take a very long time, since they are separated by a region of high Euclidean action

(i.e. low probability in the distribution) and so the transitions are statistically suppressed [8], [9] and [10]. It is therefore possible that tempered dynamics will have applications to theories such as lattice QCD where isolated modes arise as regions of distinct topological charge. In general, the *autocorrelation time* which measures how correlated subsequent samples are in a Monte Carlo simulation will be high if the system is trapped in an isolated mode. The purpose of tempering is to try and diminish the autocorrelation time without the need for extra computation.

In this report we follow the work of Creutz and Freedman in [11] who constructed a MCMC simulation of the harmonic and anharmonic oscillators, however we will use HMC where they used the Metropolis algorithm. Reproducing the results in [11] using the Metropolis algorithm is a popular choice for undergraduate projects e.g. [12], [13] and [14] which provide a useful comparison for our simulation which uses a different algorithm. We begin the report by following the steps taken in [11] where we introduce and define the path integral of quantum mechanics and show that for a discrete lattice in Euclidean time, the path integral can be considered as a canonical partition function. This connection between quantum mechanics and statistical physics is what will enable us to apply Monte Carlo methods and we will see that the quantum expectation values in the ground state correspond to classical expectation values of the statistical system with a canonical distribution. We will then briefly review the topic of Monte Carlo methods and explain how they can be used to approximate expectation values for functions of random variables under some probability distribution via a Markov chain, as well as defining detailed balance and ergodicity which are conditions required for the Monte Carlo simulation to be valid. Before introducing the HMC algorithm we very briefly review Hamiltonian dynamics, explain how Hamilton's equations can be integrated numerically via the leapfrog method and show that as well as being reversible, the leapfrog method preserves volume on the phase space of position and momentum coordinates, which we we will see is necessary for detailed balance to hold. Having provided the relevant background we are then able to define the steps of the HMC algorithm and explain how it can be used to sample from a canonical distribution by defining axillary momenta coordinates and a Hamiltonian. In this section we will also discuss the issue of ergodicity and show that HMC obeys detailed balance. Due to the stochastic nature of MCMC algorithms, samples in the Markov chain can be correlated and we briefly discuss the effect of this on data analysis. The results section begins by applying the HMC algorithm to the canonical distribution for the quantum system. We examine the results of our simulation for the harmonic oscillator where we are able to compare with numerical and exact results for the discrete theory in [11],[12], [13] and [14]. The simple system of the harmonic oscillator provides a good testing ground for our simulation, where we can easily compare our results to theory. Moving on to the anharmonic oscillator we are able to apply our simulation to a system that does not have an analytic solution. For the case of the anharmonic oscillator we are able to compare our results to those acquired through alternative methods [15] and simulations [13] however we will observe in this section that the HMC simulation begins to fail in the case of the isolated modes in the double well potential. In particular the double well system can have difficulty tunnelling between the two modes. At this point we will examine the effects of introducing a tempering parameter into the simulation, with the hope that it will increase the tunnel rate. We will show that volume preservation still holds for tempered dynamics,

which is necessary if it is still to obey detailed balance. Ultimately we find that tempering does not solve the problem of the isolated modes in our system, however we end the report by investigating why this is the case by tracking the evolution of the Hamiltonian during a tempered leapfrog proposal trajectory. Based on this result we are then able to propose a modified tempering method for future investigations into tempered HMC.

2 Background

2.1 Quantum Mechanics

2.1.1 The Path Integral

In quantum mechanics we are often interested in calculating the path integral:

$$\langle x_b, t_b | x_a, t_a \rangle = \int_{x_a}^{x_b} \mathcal{D}x\,(t) \exp\left(\frac{i}{\hbar} S_M\left[x\,(t)\right]\right), \tag{1}$$

where we will use the notation:

$$Z_{ba} = \langle x_b, t_b | x_a, t_a \rangle \tag{2}$$

$$= \langle x_b |\, e^{-i(t_b - t_a)\hat{H}/\hbar}\, | x_a \rangle\,, \tag{3}$$

and $\hat{H}$ is the usual quantum mechanical Hamiltonian operator:

$$\hat{H}\left(\hat{p}, \hat{x}\right) = \frac{\hat{p}^2}{2m} + \hat{V}\left(\hat{x}\right). \tag{4}$$

Z_{ba} is the transition amplitude for a particle of mass m in position eigenstate (in the Heisenberg picture) $|x_a, t_a\rangle$ to move to position eigenstate $|x_b, t_b\rangle$; this gives us the probability amplitude of a particle at x_a at time t_a to move to position x_b at time t_b. The term on the right of equation 1 is known as the *Feynman path integral*. The measure $\int \mathcal{D}x\,(t)$ is an integral over all paths between x_a and x_b and $S_M\left[x\,(t)\right]$ is the Minkowski action of a particle of mass m on the path $x\,(t)$ which is defined by:

$$S_M\left[x\,(t)\right] = \int_{t_a}^{t_b} dt \left[\frac{1}{2} m \left(\frac{dx}{dt}\right)^2 - V(x)\right], \tag{5}$$

where $x\,(t_a) = x_a$ and $x\,(t_b) = x_b$ are the boundary conditions, and $V\,(x)$ (now a function) is the potential the particle is in.

Due to the oscillating integrand in equation 1, it is not clear the integral will converge, and the integral measure needs to be defined before we proceed. In order to do this we follow the steps taken in [11] to get equation 1 into a form we can work with.

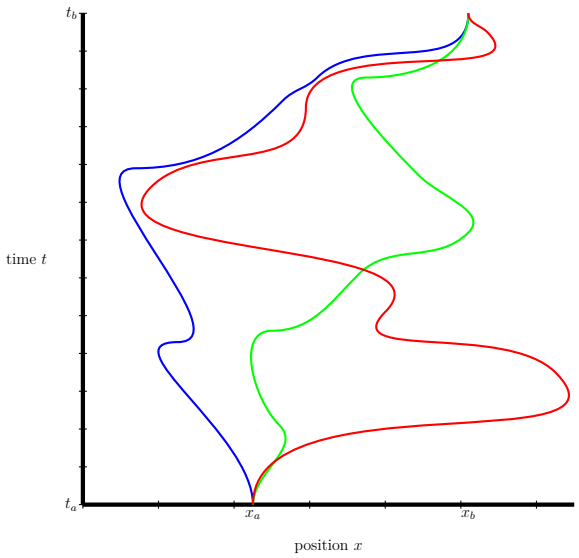


Figure 1. Three of the infinitely many paths from (x_a, t_a) to (x_b, t_b) that contribute to the path integral.

2.1.2 Connecting to Statistical Physics

In the transition amplitude we integrate over an infinite number of paths, figure 1 shows three such paths that contribute to the path integral. The first step in calculating the path integral is to discretise time, this is shown in figure 2 where we have discretised the green path in figure 1 onto a time lattice and we assume the particle travels along a straight line between the time sites. For each time site t_i on the lattice we have a continuous position variable $x_i = x(t_i)$ where $i = 0, 1, \ldots, N$ which gives the position of the particle on the path at that point on the lattice. We introduce the notation:

$$\boldsymbol{x} = (x_0, x_1, \ldots, x_N) \tag{6}$$

to denote a particular path on the lattice where each position coordinate has been specified and we refer to equation 6 as a *configuration* on the lattice. In our notation for the labelling of the position eigenstates in equation 1 we also have $x_b = x_N = x(t_N)$ and $x_a = x_0 = x(t_0)$ in order to match with figure 2. ϵ is the spacing between lattice sites and so $\epsilon = \frac{t_b - t_a}{N} = t_{i+1} - t_i$ and for $k = 0, 1, \ldots, N$ we have $t_k = t_a + k\epsilon$. In order to discretise the action in equation 5 we approximate the derivative by a forward difference and the integral as a Riemann sum since we assume the particle is travelling on a straight line between lattice sites:

$$S_M(\boldsymbol{x}) = \sum_{i=0}^{N-1} \epsilon \left[\frac{1}{2} m \left(\frac{x_{i+1} - x_i}{\epsilon} \right)^2 - V(x_i) \right]. \tag{7}$$

Since for $i = 1, 2, \ldots, N-1, -\infty < x_i < \infty$ we may define the integral measure in equation 1 as:

$$\int_{x_a}^{x_b} \mathcal{D}x \sim \prod_{n=1}^{N-1} \int_{-\infty}^{\infty} dx_n, \tag{8}$$

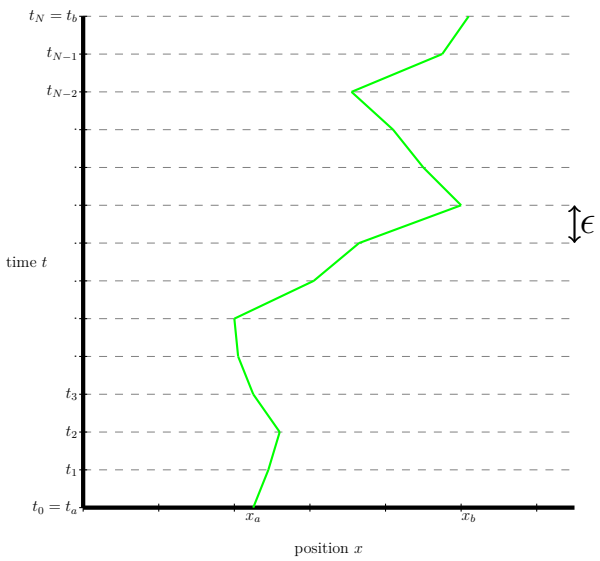


Figure 2. Discretising time and a path from (x_a, t_a) to (x_b, t_b) onto a lattice of time spacing ϵ.

which up to normalisation integrates over all possible routes through the lattice; so that for our discrete time lattice, the path integral is given by:

$$Z_{ba} \sim \int_{-\infty}^{+\infty} \prod_{i=1}^{N-1} dx_i \exp\left(\frac{i}{\hbar} S_M(\boldsymbol{x})\right). \tag{9}$$

In the limit that $N \to \infty$ (or equivalently $\epsilon \to 0$) we recover (up to normalisation) equation 1 from equation 9 exactly. The normalisation in this expression is irrelevant since we will see that in the expectation values we wish to calculate any normalisation would cancel anyway.

In order to work with the discrete path integral we have one final step. We make a *Wick rotation* into imaginary time; this is done via the substitution:

$$\tau = it. \tag{10}$$

Applying this to the discretised theory developed above by defining $a = i\epsilon$, we now have a lattice in imaginary time, of lattice spacing a, substituting a into equation 7:

$$S_M(\boldsymbol{x}) = i \sum_{i=0}^{N-1} a \left[\frac{1}{2} m \left(\frac{x_{i+1} - x_i}{a}\right)^2 + V(x_i)\right] \tag{11}$$

$$= i S_E(\boldsymbol{x}), \tag{12}$$

where we have redefined the notation slightly so that $x_i = x(\tau_i)$ since we now have a lattice in imaginary time. The quantity

$$S_E(\boldsymbol{x}) \equiv \sum_{i=0}^{N-1} a \left[\frac{1}{2} m \left(\frac{x_{i+1} - x_i}{a}\right)^2 + V(x_i)\right] \tag{13}$$

is the discretised *Euclidean action*; it has this name because the effect of the Wick transformation is that it turns the Minkowski metric ds_M on the coordinates (x, y, z, t) into the Euclidean metric ds_E on the coordinates (x, y, z, τ) and vice-versa:

$$ds_M^2 = -dt^2 + dx^2 + dy^2 + dz^2 \tag{14}$$
$$= d\tau^2 + dx^2 + dy^2 + dz^2 \tag{15}$$
$$= ds_E^2. \tag{16}$$

Equation 12 is a very useful result since upon substitution into the discrete path integral we find:

$$Z_{ba} \sim \int_{-\infty}^{+\infty} \prod_{i=1}^{N-1} dx_i \exp\left(-\frac{1}{\hbar} S_E(\boldsymbol{x})\right) \tag{17}$$

which we refer to as the *discrete euclidean path integral* and will converge since the integrand is now exponentially suppressed. We will impose periodic boundary conditions by identifying the first and last lattice sites (i.e. take $x_b = x_a$) and then integrate over that site, which can be quantified through:

$$Z = \mathrm{Tr}(Z_{ba}) = \int dx_a \int dx_b \delta(x_b - x_a) Z_{ba} \tag{18}$$
$$= \int_{-\infty}^{+\infty} \prod_{i=0}^{N-1} dx_i \exp\left(-\frac{1}{\hbar} S_E(\boldsymbol{x})\right), \tag{19}$$

where in equation 19 the identification of the first and last lattice sites is now implicit. We have the standard result from statistical physics that for a system with a fixed number N continuous degrees of freedom labelled by x_i for $i = 0, 1, \ldots N-1$, so that the vector $\boldsymbol{x} = (x_0, \ldots, x_{N-1})$ describes the system with a classical Hamiltonian H, then the canonical partition function is given by:

$$\mathcal{Z} \sim \int_{-\infty}^{+\infty} \prod_{i=0}^{N-1} dx_i \exp(-\beta H(\boldsymbol{x})), \tag{20}$$

with $\beta = \frac{1}{k_B T}$ where T is the system temperature and k_B is the Boltzmann constant. Comparing equation 19 to equation 20 we can see that the discretised Euclidean path integral is a classical canonical partition function on a system with N degrees of freedom, provided that we take:

$$\frac{1}{\hbar} S_E(\boldsymbol{x}) = \beta H(\boldsymbol{x}), \tag{21}$$

and impose periodic boundary conditions. We then have a Boltzmann factor given by $\exp\left(-\frac{1}{\hbar} S_E(\boldsymbol{x})\right)$. So in summary we now have a classical interpretation of our quantum calculation of the path integral; our lattice is essentially a one dimensional crystal of size N at temperature T with a continuous variable x_i at each crystal site, and its classical Hamiltonian (in the statistical system and units where $\hbar = \beta = 1$) is given by $S_E(\boldsymbol{x})$ which couples the nearest neighbour lattice variables x_i and places each variable in its own potential.

Since we are interested in calculating properties of the quantum system, e.g. position expectation, ground state energy, the first excited state energy etc. we will explore how the

quantum expectation values relate to those of the statistical system. Note that upon applying the Wick transformation and periodic boundary conditions to the continuum theory path integral in equation 3:

$$Z = \int_{-\infty}^{+\infty} dx \, \langle x | \, e^{-(\tau_b - \tau_a)\hat{H}/\hbar} \, | x \rangle = \mathrm{Tr}\left(e^{-(\tau_b - \tau_a)\hat{H}/\hbar} \right). \tag{22}$$

Following [11] we define for any operator $\hat{A}$ the quantity:

$$\langle \hat{A} \rangle = \frac{\mathrm{Tr}\left(e^{-(\tau_b - \tau_a)\hat{H}/\hbar} \hat{A} \right)}{\mathrm{Tr}\left(e^{-(\tau_b - \tau_a)\hat{H}/\hbar} \right)} \tag{23}$$

and after repeating the above discretisation theory equation 23 can be written as [11]:

$$\langle \hat{A} \rangle = \frac{\int_{-\infty}^{+\infty} \prod_{i=0}^{N-1} dx_i A\left(x_0, \ldots, x_{N-1}\right) \exp\left(-\frac{1}{\hbar} S_E\left(\boldsymbol{x}\right) \right)}{\int_{-\infty}^{+\infty} \prod_{i=0}^{N-1} dx_i \exp\left(-\frac{1}{\hbar} S_E\left(\boldsymbol{x}\right) \right)} \tag{24}$$

$$= \langle A \rangle \tag{25}$$

where $A\left(x_0, \ldots x_{N-1}\right)$ is a now a function on our lattice position variables. Equation 24 is precisely the statistical expectation value $\langle A \rangle$[1] of the function A on our system with the canonical distribution with density function:

$$P\left(\boldsymbol{x}\right) = \frac{1}{Z} \exp\left(-\frac{1}{\hbar} S_E\left(\boldsymbol{x}\right) \right) \tag{26}$$

where Z is the partition function for the discrete theory derived as above. We can see from equation 24 that the normalisation in Z is indeed not relevant for calculating such expressions as it would cancel between the numerator and the denominator in the discretisation. It can also be shown that in the limit that $\tau_b - \tau_a \to \infty$, that is, for a large enough lattice, equation 23 takes the form [11]

$$\langle \hat{A} \rangle = \frac{\sum_{n=0}^{\infty} e^{-\frac{1}{\hbar} E_n (\tau_b - \tau_a)} \langle n | \, \hat{A} \, | n \rangle}{\sum_{n=0}^{\infty} e^{-\frac{1}{\hbar} E_n (\tau_b - \tau_a)}} \tag{27}$$

$$= \langle 0 | \, \hat{A} \, | 0 \rangle . \tag{28}$$

So, in order to calculate the expectation of some operator e.g. position, position squared or the energy in the the ground state, we can utilise equation 24 and calculate the expectation of the corresponding function on the statistical system under the canonical distribution. In practice the high dimensional integral in 24 is not possible to compute analytically so we use Monte Carlo methods which we develop in the next chapter.

[1] There is a clash of notation with regards to using the angle brackets here. However, since it is useful to know in what context the values are being calculated, we will always use hats to denote quantum expectations and include the state with respect to which the expectation is being calculated, i.e. $\langle 0 | \, \hat{A} \, | 0 \rangle$ and use $\langle A \rangle$ for the statistical expectation. This distinction is somewhat redundant since we have seen the two quantities are equivalent for a large enough lattice, but it is useful to determine the context in which we are working.

Due to divergences in the kinetic term of the expectation of the action $\langle S_E \rangle$ of $\mathcal{O}(1/a)$[11] that occur in the calculation for the ground state energy given by:

$$E_0 = \langle 0| \hat{H} |0\rangle \tag{29}$$

$$= \lim_{\tau_b - \tau_a \to \infty} \frac{\mathrm{Tr}\left(e^{-(\tau_b - \tau_a)\hat{H}/\hbar}\hat{H}\right)}{\mathrm{Tr}\left(e^{-(\tau_b - \tau_a)\hat{H}/\hbar}\right)} \tag{30}$$

$$= \lim_{\tau_b - \tau_a \to \infty} \frac{-1}{\tau_b - \tau_a} \frac{\partial}{\partial\left(\hbar^{-1}\right)} \ln Z \tag{31}$$

$$= \lim_{\tau_b - \tau_a \to \infty} \frac{-1}{\tau_b - \tau_a} \frac{\partial}{\partial\left(\hbar^{-1}\right)} \ln \int_{-\infty}^{+\infty} \prod_{i=0}^{N-1} dx_i \exp\left(-\frac{1}{\hbar} S_E(\boldsymbol{x})\right), \tag{32}$$

in the discrete theory, we need an alternative method for calculating the lowest lying energy eigenvalues in our system as $a \to 0$. Rather than use the *point splitting* fix given in [16] we will follow the method used in [11], [13] and [14] and utilise the quantum *virial theorem* which relates the expectation value of kinetic energy to that of the potential for stationary states as:

$$2\langle n| \hat{T} |n\rangle = \langle n| \hat{x}\hat{V}'(\hat{x}) |n\rangle \tag{33}$$

the derivation of which can be found in [17] and [18]. Using equations 24 and we then have that the lowest lying energy eigenstates can be calculated as:

$$E_0 = \langle 0| \hat{H} |0\rangle \tag{34}$$

$$= \langle 0| \hat{T} + \hat{V} |0\rangle \tag{35}$$

$$= \langle 0| \frac{1}{2}\hat{x}\hat{V}'(\hat{x}) + \hat{V}(\hat{x}) |0\rangle \tag{36}$$

$$= \langle \frac{1}{2} x V'(x) + V(x)\rangle \tag{37}$$

for a large enough lattice. It should be noted that the virial theorem as stated in equation 33 is a continuum result and its applicability to the discrete case may not be valid, so correction terms may be needed. Indeed, we found that when calculating the first excited state energy via the naive method of computing $E_1 = \Delta E + E_0$ where the energy gap $\Delta E = E_1 - E_0$ was computed via the correlation function which is a result from discrete theory, and E_0 via the virial theorem, a continuum result, there was a discrepancy in the case of the harmonic oscillator with the analytic result for E_1 which can be computed exactly.

It is interesting to note that in quantum mechanics, due to the uncertainty principle $\Delta x \Delta p \geq \frac{\hbar}{2}$, $\hbar$ provides a measure of quantum fluctuations in our system. As $\hbar \to 0$ we recover classical physics and in this limit the only path in the path integral that contributes to the transition amplitude is the classical one. On the other hand, in statistical mechanics T provides a measure of statistical fluctuations in our system, and in the limit that $T \to 0$ these fluctuations go to zero. Hence taking the limit that $\hbar \to 0$ and $T \to 0$ we see statistical mechanics on a (real) crystal lattice is equivalent quantum mechanics in imaginary time.

3 Methods

3.1 Monte Carlo Methods

Monte Carlo methods provide a way of numerically evaluating integrals where the error is independent of the dimensionality of the integral. For the purposes of our work we will be interested in estimating expectation values of functions (which are the observables of our statistical system) $A(X)$ of some d-dimensional random variable X under some probability distribution with density function $f(\boldsymbol{x})$:

$$\langle A(X)\rangle = \int \prod_{i=1}^{d} dx_i f(\boldsymbol{x}) A(\boldsymbol{x}), \tag{38}$$

since this takes the form of the path integral we wish to calculate, where $\boldsymbol{x}$ correspond to the configurations on the lattice, $f(\boldsymbol{x})$ the canonical distribution and $A(\boldsymbol{x})$ some function on the lattice such as mean position. In order to estimate the integral we approximate the expectation as a sample mean on some finite set of M samples of X as (we will refer to this as the *Monte Carlo estimate*):

$$\langle A(X)\rangle \approx \frac{1}{M}\sum_{n=1}^{M} A(\boldsymbol{x}_n). \tag{39}$$

It can be shown that the error in equation 39 is always $\mathcal{O}\left(1/\sqrt{M}\right)$ independent of the dimension of the integral, and that the Monte Carlo method wins over quadrature methods (traditional numerical integration schemes) in terms of computational cost vs. accuracy for $d > 3$ [19]. It is now just a question of how to choose the samples $\boldsymbol{x}_n$ to get a expectation value with a small error. From the form of equation 38 drawing uniform samples from the support of f is clearly unwise, since this would give equal weight to contributions to our approximation of the expectation value for samples that are very unlikely to be realised by the random variable X under its distribution, as those which are very likely to be realised. We therefore choose to draw samples according to the distribution of X, this is known as *importance sampling*. In order to draw samples according to a probability distribution we use a *Markov chain* which for our purposes can be defined as a sequence of random variables (samples) for which the probability of the next random variable in the sequence $\boldsymbol{x}'$ taking on a particular value depends only on the value of the current random variable $\boldsymbol{x}$, we will denote this probability by $P(\boldsymbol{x} \to \boldsymbol{x}')$. By starting from some arbitrary state (which in our simulation is a configuration) through the stochastic sequence of configurations we will eventually end up sampling from the desired equilibrium distribution with density function $f(\boldsymbol{x})$. This idea of a *Markov Chain Monte Carlo* method was first introduced by Metropolis et al. in [2] via the Metropolis algorithm. Hybrid Monte Carlo is a method for constructing such a Markov chain and it is discussed in the next section.

A Markov chain will converge on a distribution with density function $P(\boldsymbol{x})$ provided the chain is ergodic and obeys detailed balance. Ergodicity is the condition that any sample

that can be realised under the probability distribution can be reached through the Markov chain and the detailed balance condition is given by:

$$P(\boldsymbol{x})\,P(\boldsymbol{x}\to\boldsymbol{x}') = P(\boldsymbol{x}')\,P(\boldsymbol{x}'\to\boldsymbol{x})\,. \tag{40}$$

During an actual simulation, one only starts to calculate the values of observables via equation 39 after an equilibration period, this is due to the fact that initially samples will not be drawn according to the correct probability distribution and it is only once the Markov chain has converged that they will.

We may consider our Monte Carlo simulation as consisting of three main stages. First to start the simulation we provide an initial configuration to begin the chain. We then have to perform a sufficient number of updates in the chain such that we are drawing from the correct distribution, which in our case is the canonical distribution for our statistical system. Once we are at equilibrium we may compute the values of any observables on any configuration and use them to calculate the Monte Carlo estimate via equation 39. In order to provide an initial lattice configuration we found that after some experimentation, a reasonable method was to provide a so called *hot start* by uniformly distributing the position values of the lattice sites on the interval $[-1, 1]$. To determine when and if samples are being drawn from the equilibrium distribution one can observe the evolution of some set of observables, for example position, and see when these values begin to converge. This method has potential issues in that there are the possibilities of *metastable states* [20] for which it may seem as if the system has reached equilibrium, when in fact it is sampling from region of the configuration space for which it is metastable, i.e. it will remain sampling from this region for a long period of time but will not remain indefinitely. This issue will arise for the case of the anharmonic oscillator, where the possibility of the particle getting "stuck" in one well can lead to Monte Carlo estimates on $\langle x\rangle$ being non-zero for a symmetric double well about zero. However, in this case we are able to identify when this metastability has occurred, since by symmetry we know the true answer $\langle x\rangle = 0$. Once we have achieved equilibrium we are able to evaluate observables via the configurations, however due to the possibility of correlations between samples in the Markov chain this is not entirely trivial; the analysis of recorded data is discussed at the end of this chapter.

3.2 Hybrid Monte Carlo

3.2.1 Hamiltonian Dynamics

Due to the fundamental importance of Hamiltonian dynamics in the HMC algorithm, in this section we provide a quick review of Hamilton's equations and explain the numerical method used to integrate these equations, so that they may be used in computational simulations.

Hamiltonian Dynamics enables us to solve the time evolution of a system of canonical coordinates $(\boldsymbol{q}, \boldsymbol{p})$ where the d-dimensional $\boldsymbol{q} = (q_1, q_2, \ldots, q_d)$ and $\boldsymbol{p} = (p_1, p_2, \ldots, p_d)$ vectors are

called *position* and *momentum* respectively. The system is then characterised by a Hamiltonian function $H(\boldsymbol{q}, \boldsymbol{p})$ on the $2d$-dimensional *phase space* occupied by the $(\boldsymbol{q}, \boldsymbol{p})$ vectors. **Hamilton's equations**:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i} \tag{41}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i} \tag{42}$$

for $i = 1, 2, \ldots d$, provide the time evolution of the vectors $\boldsymbol{q}$ and $\boldsymbol{p}$, so that if the state of the system is $(\boldsymbol{q}, \boldsymbol{p})$ at time t then Hamilton's equations give a mapping T_s to the state $(\boldsymbol{q}', \boldsymbol{p}')$ at time $t + s$. We will see in section 3.2.2 for the purposes of the implementing the HMC algorithm we may assume that the Hamiltonian is of the form:

$$H(\boldsymbol{q}, \boldsymbol{p}) = U(\boldsymbol{q}) + K(\boldsymbol{p}), \tag{43}$$

Where $U(\boldsymbol{q})$ and $K(\boldsymbol{p})$ are known as the *potential* and *kinetic* terms respectively. Equations 41 and 42 then become:

$$\frac{dq_i}{dt} = \frac{\partial K}{\partial p_i} \tag{44}$$

$$\frac{dp_i}{dt} = -\frac{\partial U}{\partial q_i}. \tag{45}$$

Since we intend to use the solutions to Hamilton's equations in the HMC simulation we need to find a way of integrating them numerically; in [3] it is argued that a successful method for this is *leapfrog integration*. In the leapfrog method to make the small time step ϵ from t to $t + \epsilon$ in position and momentum we use the equations:

$$p_i(t + \epsilon/2) = p_i(t) + \epsilon/2 \frac{dp_i}{dt}(t) = p_i(t) - \epsilon/2 \frac{\partial U}{\partial q_i}(\boldsymbol{q}(t)), \tag{46}$$

$$q_i(t + \epsilon) = q_i(t) + \epsilon \frac{dq_i}{dt}(t + \epsilon/2) = q_i(t) + \epsilon \frac{\partial K}{\partial p_i}(\boldsymbol{p}(t + \epsilon/2)), \tag{47}$$

$$p_i(t + \epsilon) = p_i(t + \epsilon/2) + \epsilon/2 \frac{dp_i}{dt}(t + \epsilon/2) = p_i(t + \epsilon/2) - \epsilon/2 \frac{\partial U}{\partial q_i}(\boldsymbol{q}(t + \epsilon)), \tag{48}$$

where in the second equality in each step we have made the substitution for Hamilton's equations. In equation 46 we begin with a half step in each momentum component to go from t to $t+\epsilon/2$. Using the half stepped momenta we then do a full step from t to $t+\epsilon$ in each position component in equation 47. We end in equation 48 with a second half step in the momentum components to go from $t + \epsilon/2$ to $t + \epsilon$ using the updated position components. Iterating this process we do as many steps of size ϵ as we like.

This generalises to making l steps in position and momentum and we can combine any two half steps in momentum that follow one another to simplify the algorithm which is often more convenient for computation purposes. Starting at $t = 0$:

1. We begin with a half step in momentum:

$$p_i(\epsilon/2) = p_i(0) - \epsilon/2 \frac{\partial U}{\partial q_i}(\boldsymbol{q}(0)), \tag{49}$$

and then a full step in position:

$$q_i(\epsilon) = q_i(0) + \epsilon \frac{\partial K}{\partial p_i}(\boldsymbol{p}(\epsilon/2)). \tag{50}$$

2. Then make $l-1$ alternating full steps in momentum and position:

$$p_i(n\epsilon + \epsilon/2) = p_i(n\epsilon - \epsilon/2) - \epsilon \frac{\partial U}{\partial q_i}(\boldsymbol{q}(n\epsilon)), \tag{51}$$

$$q_i(n\epsilon + \epsilon) = q_i(n\epsilon) + \epsilon \frac{\partial K}{\partial p_i}(\boldsymbol{p}(n\epsilon + \epsilon/2)), \tag{52}$$

for $n = 1, \ldots, l-1$.

3. Then a final half step in momentum:

$$p_i(l\epsilon) = p_i(l\epsilon - \epsilon/2) - \epsilon/2 \frac{\partial U}{\partial q_i}(\boldsymbol{q}(l\epsilon)). \tag{53}$$

The reasoning for the name *leapfrog* becomes apparent here since apart from the initial and final half steps in momentum, the momentum and position values are used alternatively to calculate the position and momentum respectively at the next step and the variables "leap" over one another at on offset of $\epsilon/2$.

An important feature of the leapfrog integration scheme that makes it applicable to HMC is that it preserves volume in phase space exactly, which we will see leads to detailed balance. For Hamiltonian dynamics in continuous time, preservation of volume in phase space is known as Louville's theorem and its proof is given in [21]. For the discrete dynamics integrated via the leapfrog scheme we can easily verify that volume preservation is still the case. First let us define the mappings $\mathcal{T}_q(\epsilon) : (\boldsymbol{q}, \boldsymbol{p}) \to (\boldsymbol{q}', \boldsymbol{p}')$ and $\mathcal{T}_p(\epsilon) : (\boldsymbol{q}, \boldsymbol{p}) \to (\boldsymbol{q}', \boldsymbol{p}')$ on phase space such that:

$$\mathcal{T}_q(\epsilon) \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \boldsymbol{q} + \epsilon \underline{\nabla}_p K(\boldsymbol{p}) \\ \boldsymbol{p} \end{pmatrix}, \tag{54}$$

$$\mathcal{T}_p(\epsilon) \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{p} - \epsilon \underline{\nabla}_q U(\boldsymbol{q}) \end{pmatrix}, \tag{55}$$

then (with a slight abuse of notation) the Jacobians of these mappings will be of the form:

$$J = \det \begin{pmatrix} \frac{\partial q_i'}{\partial q_j} & \frac{\partial q_i'}{\partial p_j} \\ \frac{\partial p_i'}{\partial q_j} & \frac{\partial p_i'}{\partial p_j} \end{pmatrix} \tag{56}$$

so that:

$$J_q = \det \begin{pmatrix} \delta_{ij} & \epsilon \partial_{p_i} \partial_{p_j} K(\boldsymbol{p}) \\ 0 & \delta_{ij} \end{pmatrix} = 1 \tag{57}$$

$$J_p = \det \begin{pmatrix} \delta_{ij} & 0 \\ -\epsilon \partial_{q_i} \partial_{q_j} U(\boldsymbol{q}) & \delta_{ij} \end{pmatrix} = 1. \tag{58}$$

For a single step of size ϵ the leapfrog algorithm equations 46, 47 and 48 define a mapping on phase space $\mathcal{T}(\epsilon) : (\boldsymbol{q}(t), \boldsymbol{p}(t)) \to (\boldsymbol{q}(t+\epsilon), \boldsymbol{p}(t+\epsilon))$ that can be written using the mappings given above as:

$$\mathcal{T}(\epsilon) = \mathcal{T}_p(\epsilon/2) \circ \mathcal{T}_q(\epsilon) \circ \mathcal{T}_p(\epsilon/2). \tag{59}$$

Since each map in the composition on the right hand side of equation 59 has a Jacobian of 1, we have that the Jacobian of $\mathcal{T}(\epsilon)$ is 1. For the total trajectory which consists of l steps of size ϵ, we may write the mapping that via leapfrog integration takes us from the start to the end of the trajectory as:

$$\text{traj}(\epsilon, l) = (\mathcal{T}(\epsilon))^l, \tag{60}$$

which of course also a Jacobian of 1 since each mapping in the composition has Jacobian 1. Hence leapfrog integration preserves volume on phase space exactly, since as a mapping its Jacobian is 1. We will return to this argument in section 4.2.2, where we will show that the leapfrog integration scheme preserves volume even for the case of tempered dynamics.

For a Hamiltonian of the form in equation 43 with a symmetric kinetic term $K(\boldsymbol{p}) = K(-\boldsymbol{p})$ (which we will see is always the case for our work), then by negating the momenta at the end of a trajectory and applying the leapfrog equations a second time, we will return to the initial point in phase space and hence the dynamics is reversible. Negating the momenta at the end of a leapfrog trajectory will also preserve the volume.

3.2.2 Sampling and the Hamiltonian

For a physical system in thermodynamic equilibrium with a fixed number of degrees of freedom and with energy function $E(\boldsymbol{x})$, where the vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ denotes the state of the system that depends on d variables, the canonical distribution in units where $k_b = 1$ is defined by the density function:

$$P(\boldsymbol{x}) = \frac{1}{\mathcal{Z}} \exp\left(-\frac{E(\boldsymbol{x})}{T}\right), \tag{61}$$

where $\mathcal{Z}$ is the canonical partition function for the system, which provides the normalisation and is given by

$$\mathcal{Z} = \int \prod_{i=1}^{d} dx_i \exp\left(-\frac{E(\boldsymbol{x})}{T}\right). \tag{62}$$

Alternatively any probability density function $P(\boldsymbol{x})$ can be constructed as the canonical distribution for the energy function $E(\boldsymbol{x}) = -T(\log P(\boldsymbol{x}) + \log \mathcal{Z})$ for some choice of T and $\mathcal{Z}$ [3].

A Hamiltonian of the form $H(\boldsymbol{q},\boldsymbol{p}) = U(\boldsymbol{q}) + K(\boldsymbol{p})$ is an energy function on the $2d$-dimensional phase space given by position and momentum $(\boldsymbol{q},\boldsymbol{p})$, with $\boldsymbol{q} = (q_1, q_2, \ldots, q_d)$ and $\boldsymbol{p} = (p_1, p_2, \ldots, p_d)$. We may define a canonical distribution on those variables via the density function:

$$P(\boldsymbol{q},\boldsymbol{p}) = \frac{1}{\mathcal{Z}} \exp\left(-\frac{H(\boldsymbol{q},\boldsymbol{p})}{T}\right), \tag{63}$$

with

$$\mathcal{Z} = \int \prod_{i=1}^{d} dx_i dp_i \exp\left(-\frac{H(\boldsymbol{q},\boldsymbol{p})}{T}\right). \tag{64}$$

Then, if $H(\boldsymbol{q},\boldsymbol{p}) = U(\boldsymbol{q}) + K(\boldsymbol{p})$ this factorises as:

$$P(\boldsymbol{q},\boldsymbol{p}) = \frac{1}{\mathcal{Z}_{\boldsymbol{q}}} \exp\left(-\frac{U(\boldsymbol{q})}{T}\right) \frac{1}{\mathcal{Z}_{\boldsymbol{p}}} \exp\left(-\frac{K(\boldsymbol{p})}{T}\right) = P(\boldsymbol{q})\, P(\boldsymbol{p}), \tag{65}$$

(where $\mathcal{Z}_{\boldsymbol{q}}$ and $\mathcal{Z}_{\boldsymbol{p}}$ denote the marginal partition functions) so the variables $\boldsymbol{q}$ and $\boldsymbol{p}$ and their respective probability distributions are independent with energy functions $U(\boldsymbol{q})$ and $K(\boldsymbol{p})$. This means that in order to sample $\boldsymbol{q}$ according to the marginal distribution of $\boldsymbol{q}$ with density function $P(\boldsymbol{q})$, we can sample from the joint distribution of $\boldsymbol{q}$ and $\boldsymbol{p}$ with density function $P(\boldsymbol{q},\boldsymbol{p})$ and disregard the $\boldsymbol{p}$.

In the HMC algorithm we employ Hamiltonian dynamics to sample from the joint distribution of $(\boldsymbol{q},\boldsymbol{p})$. If we have some probability distribution $P(\boldsymbol{x})$ for states $\boldsymbol{x}$ we wish to sample, we may through equation 61 define an energy function $E(\boldsymbol{x})$ for that distribution. We then define $\boldsymbol{q} \equiv \boldsymbol{x}$ and $U(\boldsymbol{q}) \equiv E(\boldsymbol{x})$, introduce "fictitious momenta" $\boldsymbol{p}$ and define a kinetic energy $K(\boldsymbol{p})$ and the Hamiltonian $H(\boldsymbol{q},\boldsymbol{p}) = U(\boldsymbol{q}) + K(\boldsymbol{p})$. Running the HMC algorithm generates samples according to the distribution given by equation 63, but by the factorization in equation 65 this gives samples of $\boldsymbol{x}$ according to our original distribution given by $P(\boldsymbol{x})$. In the HMC algorithm, we think of the variables of interest $\boldsymbol{x}$ as position $\boldsymbol{q}$ with conjugate momenta $\boldsymbol{p}$ which of course matches up nicely in physical applications where we are actually sampling position, although the algorithm is equally applicable in non-physical cases.

Since the fictitious momenta $\boldsymbol{p}$ are introduced artificially, we need to choose their probability distribution with density function $P(\boldsymbol{p})$ by specifying the kinetic energy function $K(\boldsymbol{p})$. For the purposes of our work we may take:

$$K(\boldsymbol{p}) = \sum_{i=1}^{d} \frac{p_i^2}{2} \tag{66}$$

and $T = 1$ from now on.

3.2.3 Steps of the HMC Algorithm

Here we will outline the main steps of the HMC algorithm as defined in [1], [22] and [3] then give a more detailed explanation of each stage and its implementation. The two key steps

are 1 and 2. In step 1 momentum is changed, where as in step 2 can change position as well as momentum. The canonical joint distribution for $(\boldsymbol{q}, \boldsymbol{p})$ is left invariant in both steps, therefore is also invariant under their combination; this is the detailed balance condition. The steps of the algorithm are:

0. Provide an initial sample $\boldsymbol{q}$.
1. Generate $\boldsymbol{p}$ from the multivariate Gaussian Distribution $\mathcal{N}(\mathbf{0}, I_{d\times d})$.
2. Evolve the variables from $(\boldsymbol{q}, \boldsymbol{p})$ to $(\boldsymbol{q}', \boldsymbol{p}')$ by simulating Hamiltonian dynamics for l steps of size ϵ and then negate the momentum variables at the end of the trajectory. Accept the proposed state $(\boldsymbol{q}', -\boldsymbol{p}')$ as the next state in the Markov chain with probability:
$$\min\left[1, \exp\left(-H_{HMC}\left(\boldsymbol{q}', -\boldsymbol{p}'\right) + H_{HMC}\left(\boldsymbol{q}, \boldsymbol{p}\right)\right)\right]$$
If the proposed state is rejected, record current state $\boldsymbol{q}$ as sample (i.e. state started with), if successful set current state to proposed state $\boldsymbol{q} = \boldsymbol{q}'$ and record it as a sample.
3. Return to step 1 with $\boldsymbol{q}$.

In step 0 we provide an initial state. This can be chosen or generated randomly, however, since we normally disregard the samples generated in the early HMC iterations as discussed above, this choice is not critical.

In step 1 we draw the fictitious momenta according to their probability distribution. The choice of kinetic energy in equation 66 and the form of equation 61 means that:

$$P\left(\boldsymbol{p}\right) = \frac{1}{\mathcal{Z}_{\boldsymbol{p}}} \exp\left(-\sum_{i=1}^{d} \frac{p_i^2}{2}\right) \tag{67}$$

$$= \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{p_i^2}{2}\right) \tag{68}$$

$$= \prod_{i=1}^{d} P\left(p_i\right) \tag{69}$$

where each $P\left(p_i\right)$ is the density function of a Gaussian distribution of mean 0 and variance 1. So the d momentum variables are independent with each component p_i of $\boldsymbol{p}$ drawn from a Gaussian $\mathcal{N}(0, 1)$, so we draw $\boldsymbol{p}$ from a multivariate Gaussian distribution of mean $\mathbf{0}$ and covariance $I_{d\times d}$. In this step $\boldsymbol{q}$ is not changed, and $\boldsymbol{p}$ gets drawn from the it's correct conditional distribution given $\boldsymbol{q}$ which is it's marginal distribution by the above independence, so the canonical joint distribution is left invariant.

In step 2 we evolve the variables $(\boldsymbol{q}, \boldsymbol{p})$ by integrating Hamilton's equations, this is often referred to as the *molecular dynamics* and we will define the map that carries out this evolution as $\mathcal{H} : (\boldsymbol{q}, \boldsymbol{p}) \to (\boldsymbol{q}', \boldsymbol{p}')$. In order to implement this in a computer simulation we use the leapfrog method described in section 3.2.1, although in general any numerical integration scheme that is exactly area preserving and reversible is valid [22]. We have a

choice of the number of time steps l and their size ϵ; it is common practice in HMC to choose ϵ and l such that $l\epsilon = 1$. At the end of the trajectory, negating the momentum variables makes the Metropolis proposal symmetric, which is needed for detailed balance to hold. However, because the choice of kinetic energy in equation 66 is quadratic in $\boldsymbol{p}$ so that the HMC Hamiltonian is symmetric in $\boldsymbol{p}$ and the momentum is replaced anyway in the next iteration, there is no need for the negation in a simulation. We can define a mapping for the momentum flip as $\mathcal{F} : (\boldsymbol{q}', \boldsymbol{p}') \to (\boldsymbol{q}', -\boldsymbol{p}')$ so that the mapping that gives the proposed update is $\mathcal{F} \circ \mathcal{H} : (\boldsymbol{q}, \boldsymbol{p}) \to (\boldsymbol{q}', -\boldsymbol{p}')$. Since $\mathcal{H}$ preserves volume, and trivially negating momenta with $\mathcal{F}$ also preserves volume the mapping $\mathcal{F} \circ \mathcal{H}$ will preserve volume. We accept or reject the proposed state with a probability given by:

$$P\left(\boldsymbol{q}', \boldsymbol{p}', \boldsymbol{q}, \boldsymbol{p}\right) = \min\left[1, \exp\left(-H_{HMC}\left(\boldsymbol{q}', \boldsymbol{p}'\right) + H_{HMC}\left(\boldsymbol{q}, \boldsymbol{p}\right)\right)\right], \tag{70}$$

which is known as a *Metropolis update*. The interpretation of equation 70 is that if the Hamiltonian decreases (or stays constant) as a result of moving to the proposed state the update is accepted with certainty. However, if the Hamiltonian increases as a result of the moving to the proposed state, the proposal is accepted with a probability that decreases exponentially for larger increases in the Hamiltonian. If the state is accepted then it becomes the current state i.e. $\boldsymbol{q} = \boldsymbol{q}'$, otherwise the current state remains as it was in step 1. In either case we then record the current state as a sample and return to step 1 with that state.

From the Metropolis update in equation 70 we can show that for smaller value of ϵ and larger l (i.e. smaller time steps and more of them) the probability of accepting an update approaches 1. To see this note that as $\epsilon \to 0$ and $l \to \infty$ the leapfrog method approaches an exact integration of Hamilton's equations. It is easily shown that for continuous time the Hamiltonian is a conserved quantity:

$$\frac{dH}{dt} = \sum_{i=1}^{d}\left[\frac{dq_i}{dt}\frac{\partial H}{\partial q_i} + \frac{dp_i}{dt}\frac{\partial H}{\partial p_i}\right] = \sum_{i=1}^{d}\left[\frac{\partial H}{\partial p_i}\frac{\partial H}{\partial q_i} - \frac{\partial H}{\partial q_i}\frac{\partial H}{\partial p_i}\right] = 0 \tag{71}$$

where in the second equality we have used Hamilton's equations. Therefore as $\epsilon \to 0$ and $l \to \infty$, $H_{HMC}\left(\boldsymbol{q}, \boldsymbol{p}\right) \to H_{HMC}\left(\boldsymbol{q}', \boldsymbol{p}'\right)$ and so the probability of accepting a state is always 1.

As ϵ decreases l increases provided $l\epsilon$ is kept fixed, therefore more steps and hence a larger computational cost is required for proposal states with smaller ϵ in the trajectory. Conversely if we have a fixed computation time, then for smaller ϵ we will generate less proposals with a higher acceptance rate and hence the error in the Monte Carlo estimate on any observable will be larger. It is shown in [23] that an acceptance rate of $\sim 65.1\%$ provides an optimal balance between the cost of generating proposals and the total number of samples.

HMC will be typically ergodic [3] however proving ergodicity is highly non-trivial, we will therefore assume that ergodicity of the HMC algorithm holds for our simulation and refer the reader to the mathematical literature on the subject such as [24], [25] and [26] for justification.

We can however show that HMC obeys detailed balance. In order to do this we will (for notational simplicity) adopt the field theory notation and set $\boldsymbol{q} \to \phi$ and $\boldsymbol{p} \to \pi$ and redefine

the density function of the canonical distribution in equation 63 as $P(\boldsymbol{q}, \boldsymbol{p}) \to p(\phi, \pi)$ where we are using a lower case p to emphasise the fact we are dealing with a probability density. $p(\phi, \pi)\, d\phi d\pi$ is then the probability of being in the volume given by the intervals $[\phi, \phi + d\phi]$ and $[\pi, \pi + d\pi]$ and since the kinetic term in the Hamiltonian is quadratic and therefore symmetric in π, $p(\phi, \pi) = p(\phi, -\pi)$. If $P((\phi, \pi) \to (\phi', \pi'))$ is the probability of moving from (ϕ, π) to (ϕ', π') the detailed balance condition for HMC is given by equation 40:

$$d\phi d\pi p(\phi, \pi) P((\phi, \pi) \to (\phi', -\pi')) = d\phi' d(-\pi') p(\phi', -\pi') P((\phi', -\pi') \to (\phi, \pi)) \quad (72)$$
$$= d\phi' d\pi' p(\phi', \pi') P((\phi', -\pi') \to (\phi, \pi)), \quad (73)$$

where in the second equality we have used preservation of volume under negation of momenta and the symmetry of the density function in π. We have shown preservation of volume in phase space for the leapfrog method, so that $d\phi d\pi = d\phi' d\pi'$. We may decompose the second factor in the LHS of equation 72 as:

$$P((\phi, \pi) \to (\phi', -\pi')) = P_{\mathcal{F} \circ \mathcal{H}}((\phi, \pi) \to (\phi', -\pi'))\, P_{\mathcal{A}}((\phi, \pi) \to (\phi', -\pi')) \quad (74)$$

where $P_{\mathcal{F} \circ \mathcal{H}}$ is the probability of proposing the update under the molecular dynamics and momentum flip and $P_{\mathcal{A}}$ is the probability of actually accepting that move. By the fact that leapfrog is reversible we have that

$$P_{\mathcal{F} \circ \mathcal{H}}((\phi, \pi) \to (\phi', -\pi')) = P_{\mathcal{F} \circ \mathcal{H}}((\phi', -\pi') \to (\phi, \pi)) \quad (75)$$

and since we are using the Metropolis update step and the Hamiltonian is symmetric in π:

$$P_{\mathcal{A}}((\phi, \pi) \to (\phi', -\pi')) = \min[1, \exp(-(H(\phi', -\pi') - H(\phi, \pi)))] \quad (76)$$
$$= \min[1, \exp(-(H(\phi', \pi') - H(\phi, \pi)))] \quad (77)$$
$$= P_{\mathcal{A}}((\phi, \pi) \to (\phi', \pi')). \quad (78)$$

Starting from the LHS of equation 72 we get that:

$$d\phi d\pi p(\phi, \pi) P((\phi, \pi) \to (\phi', -\pi')) \quad (79)$$
$$= d\phi d\pi \frac{1}{\mathcal{Z}} e^{-H(\phi, \pi)} P_{\mathcal{F} \circ \mathcal{H}}((\phi, \pi) \to (\phi', -\pi')) \min\left[1, e^{(-(H(\phi', \pi') - H(\phi, \pi)))}\right] \quad (80)$$
$$= d\phi d\pi \frac{1}{\mathcal{Z}} e^{-H(\phi', \pi')} P_{\mathcal{F} \circ \mathcal{H}}((\phi', -\pi') \to (\phi, \pi)) \min\left[e^{(-(H(\phi, \pi) - H(\phi', \pi')))}, 1\right] \quad (81)$$
$$= d\phi' d\pi' \frac{1}{\mathcal{Z}} e^{-H(\phi', \pi')} P_{\mathcal{F} \circ \mathcal{H}}((\phi', -\pi') \to (\phi, \pi)) \min\left[1, e^{(-(H(\phi, \pi) - H(\phi', \pi')))}\right] \quad (82)$$
$$= d\phi' d\pi' p(\phi', \pi') P((\phi', -\pi') \to (\phi, \pi)) \quad (83)$$

where in the first equality we have just used the definitions, in the second we have used reversibility of the leapfrog integration scheme and factored out an exponent from the min function and in the third equality we have used the volume preservation, so detailed balance holds for HMC.

3.3 Data Analysis

The results of our simulation will ultimately be average values with accompanying statistical errors for several measured observables. Because Monte Carlo simulations generate samples using a Markov chain, subsequent samples in the chain can be correlated, in which case naive error estimates can be wrong. In what follows we will briefly explore this issue and explain our method for dealing with this problem in the simulation.

Uncorrelated Data If via our HMC simulation we have generated configurations at equilibrium and calculated the values of $x_1, \ldots, x_n$ of some observable (e.g. the ground state energy, or position expectation in the ground state) on each configuration, then since the Markov chain is a sequence of random variables, each sample x_i is the realisation of some random variable X_i. The lattice configurations and therefore the samples that are functions of those configurations are generated at equilibrium according to the canonical distribution, so these random variables have the same expectation value and variance:

$$\langle X_i \rangle = \langle X \rangle, \tag{84}$$

$$\sigma_{X_i}^2 = \langle (X_i - \langle X_i \rangle)^2 \rangle = \sigma_X^2 \tag{85}$$

and unbiased estimators for these values are [19]

$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{86}$$

$$\overline{\sigma}_X^2 = \frac{1}{n-1} \sum_{i=1}^{n} \left(X_i - \overline{X} \right)^2 . \tag{87}$$

For X_i uncorrelated and $i \neq j$, $\langle X_i X_j \rangle = \langle X_i \rangle \langle X_j \rangle = \langle X \rangle^2$ from which it can be shown [19], that:

$$\sigma_{\overline{X}}^2 = \frac{1}{n} \sigma_X^2, \tag{88}$$

so that the statistical error (standard deviation) in the observable $\overline{X}$ is $\sigma_{\overline{X}}$ and by approximating σ_X by the (now biased) estimator $\overline{\sigma}_X$, for n uncorrelated measurements of some observable X our final estimate will be:

$$\overline{X} \pm \frac{\overline{\sigma}_X}{\sqrt{n}} \tag{89}$$

from which we can see the error is $\mathcal{O}\left(1/\sqrt{n}\right)$.

Correlated Data Subsequent samples generated in the HMC simulation will be correlated due to the fact that they are generated via a Markov chain and hence the above error analysis will not be valid in general. In order to quantify the correlation between samples we can use the *autocorrelation function*. If the random variables are correlated, they will have a non-vanishing autocorrelation function which is given by:

$$C_X \left(X_i, X_{i+t} \right) = \langle \left(X_i - \langle X_i \rangle \right) \left(X_{i+t} - \langle X_{i+t} \rangle \right) \rangle = \langle X_i X_{i+t} \rangle - \langle X_i \rangle \langle X_{i+t} \rangle. \tag{90}$$

For invariance in the shift of the index $\langle X_i \rangle = \langle X \rangle$ (which we can assume here since we are always sampling from the canonical distribution at equilibrium) the correlation function only depends on the separation time t between samples in the chain, and utilising the translational invariance in the index i we are able to get a good estimate on the correlation function from the sum:

$$C_X(t) \approx \frac{1}{n} \sum_{i=1}^{n} C_X(X_i, X_{i+t}). \tag{91}$$

It is useful to normalise the autocorrelation function by:

$$\Gamma_X(t) \equiv \frac{C_X(t)}{C_X(0)}, \tag{92}$$

since equation 92 typically takes the form of a sum of exponentials however for large t we may truncate to the asymptomatically leading term [19]:

$$\Gamma_X(t) \sim \exp\left(-\frac{t}{\tau_{X,\text{exp}}}\right), \tag{93}$$

where $\tau_{X,\text{exp}}$ is the *exponential autocorrelation time* for X, which gives us a measure of the number of configurations in the Markov chain, before measurements we are working with become uncorrelated. It is important to note that different observables will have in general, different autocorrelation times. For example, we may on each configuration of the lattice compute position expectation, and position squared expectation, which may have different autocorrelation times. Therefore in order to get an overall measure of when all measurements become uncorrelated we should compute:

$$\tau_{\text{exp}} = \sup_X \tau_{X,\text{exp}} \tag{94}$$

for all observables X.

For correlated random variables, equation 88 is no longer valid, however it can be shown for the correlated case [19] that:

$$\sigma_{\overline{X}}^2 \approx \frac{\sigma_X^2}{n} 2\tau_{X,\text{int}} \tag{95}$$

where the *integrated autocorrelation time* $\tau_{X,\text{int}}$ is defined by:

$$\tau_{X,\text{int}} = \frac{1}{2} + \sum_{t=1}^{n} \Gamma_X(t), \tag{96}$$

and provides a second measure (like the exponential autocorrelation time) of the number of configurations before the samples become uncorrelated. As before, the integrated autocorrelation time will vary for different measured quantities, so to find the number of configurations before measurements are completely uncorrelated one should consider $\sup_X \tau_{X,\text{int}}$. For the case of correlated samples, our estimates will be now:

$$\overline{X} \pm \sqrt{\frac{1}{n} 2\tau_{X,\text{int}} \overline{\sigma}_X^2}, \tag{97}$$

and we can see that the error will be larger than in equation 89 where we assumed the samples were uncorrelated.

In practice obtaining the values of either the exponential, or integrated autocorrelation times is not always easy. Due to the statistical noise that appears in the autocorrelation function at larger times one needs to first plot the autocorrelation function, then either fit an exponential for $\tau_{X,\text{exp}}$, or compute the sum in $\tau_{X,\text{int}}$ for an appropriate range. For the systems we simulated computational costs are (relatively) small, so rather than compute the autocorrelation times after the simulation to determine the errors in any quantities, we introduce into the simulation a parameter that determines at what frequency measurements are actually recorded from configurations in the Markov chain. By increasing this parameter on preliminary test runs until $\tau_{X,\text{int}}$ and $\tau_{X,\text{exp}}$ are both less than one, we know that our measurements are always uncorrelated and we are therefore free to use the naive error in equation 89 with confidence. This makes the simulation somewhat simpler, since it allows us to do statistical analysis during the simulation, rather than saving all generated configurations and performing data analysis after.

Finally in section 3.2.2 we claimed the optimum acceptance rate for HMC was $\sim 65.1\%$, in the results that follow we tuned our input parameters of the number of leapfrog steps l and step size ϵ such that the success rate is close to or greater than this value. In each simulation we also ran an initial test to see how long the system took to reach equilibrium by measuring the evolution of the value of an observable during the simulation. In all results that follow configurations generated during the equilibration period have been disregarded, and we can therefore assume that all measurements were made at equilibrium.

4 Results and Discussion

4.1 Quantum Oscillators

In section 2.1.1 we showed that by discretising the path integral and performing a Wick rotation into imaginary time, the path integral of quantum mechanics is a partition function. Working now in units where $\hbar = T = k_B = 1$ we have a statistical system with the canonical distribution and density function:

$$P(\boldsymbol{x}) = \frac{1}{Z} \exp\left(-S_E(\boldsymbol{x})\right). \tag{98}$$

The probability distribution of equation 98 is the distribution we wish to draw our samples according to for the Monte Carlo simulations, where the samples $\boldsymbol{x}_i = (x_{i_1}, \ldots, x_{i_n})$ are lattice configurations, therefore we follow the method of section 3.2.2 with this distribution, taking $\boldsymbol{q} \equiv \boldsymbol{x}$,

$$U(\boldsymbol{q}) \equiv S_E(\boldsymbol{x}) \tag{99}$$

$$= \sum_{i=0}^{N-1} a \left[\frac{1}{2} m \left(\frac{x_{i+1} - x_i}{a} \right)^2 + V(x_i) \right] \tag{100}$$

and the kinetic energy $K(\boldsymbol{p})$ as in equation 66. We define our HMC Hamiltonian:

$$H_{HMC}(\boldsymbol{q},\boldsymbol{p}) \equiv K(\boldsymbol{p}) + U(\boldsymbol{q}) \tag{101}$$

$$= \sum_{i=0}^{N-1} \frac{p_i^2}{2} + S_E(\boldsymbol{q}) \tag{102}$$

$$= \sum_{i=0}^{N-1} \frac{p_i^2}{2} + \sum_{i=0}^{N-1} a\left[\frac{1}{2}m\left(\frac{q_{i+1}-q_i}{a}\right)^2 + V(q_i)\right], \tag{103}$$

we can then use this Hamiltonian to generate samples with the HMC algorithm for any 1-dimensional quantum mechanical system, by specifying the potential $V(x)$. Once we reach equilibrium, we can compute the value of any functions (observables) on the configurations which via the Monte Carlo estimate of equation 39, gives an approximation of the true expectation value under the canonical distribution and we know that these statistical expectations correspond to quantum expectation values in the ground state for a large enough lattice. With the choice of kinetic energy given in equation 66 and $U(\boldsymbol{q}) \equiv S_E(\boldsymbol{x})$, this leads to the approximate solutions to Hamilton's equations in leapfrog implementation for l steps of size ϵ taking the form:

1. Half step in momentum:

$$p_i(\epsilon/2) = p_i(0) - \epsilon/2\left[\frac{m}{a}(2q_i(0) - q_{i-1}(0) - q_{i+1}(0)) + a\frac{\partial V(q_i(0))}{\partial q_i}\right], \tag{104}$$

   then full step in position:

$$q_i(\epsilon) = q_i(0) + \epsilon p_i(\epsilon/2). \tag{105}$$

2. Make $l-1$ alternating full steps in momentum and position:

$$p_i(n\epsilon+\epsilon/2) = p_i(n\epsilon-\epsilon/2) - \epsilon\left[\frac{m}{a}(2q_i(n\epsilon) - q_{i-1}(n\epsilon) - q_{i+1}(n\epsilon)) + a\frac{\partial V(q_i(n\epsilon))}{\partial q_i}\right], \tag{106}$$

$$q_i(n\epsilon+\epsilon) = q_i(n\epsilon) + \epsilon p_i(n\epsilon+\epsilon/2), \tag{107}$$

   for $n = 1, \ldots, l-1$.

3. Final half step in momentum:

$$p_i(\epsilon l) = p_i(l\epsilon-\epsilon/2) - \epsilon/2\left[\frac{m}{a}(2q_i(\epsilon l) - q_{i-1}(\epsilon l) - q_{i+1}(\epsilon l)) + a\frac{\partial V(q_i(\epsilon l))}{\partial q_i}\right], \tag{108}$$

where we are as mentioned previously, imposing the periodic boundary conditions that $q_N = q_0$.

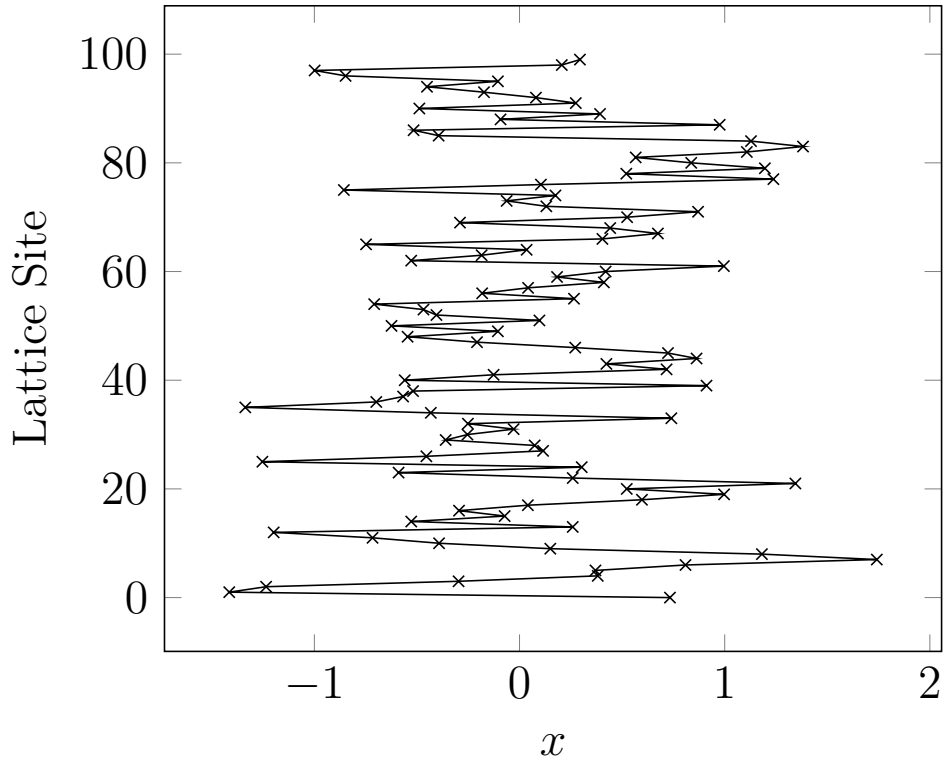


Figure 3. Typical configuration for the harmonic oscillator with $\mu^2 = 1, m = 1$ on a lattice of $N = 100$ sites at spacing $a = 1$.

4.1.1 Quantum Harmonic Oscillator

We begin by applying the HMC algorithm to the quantum harmonic oscillator. Since this system has an analytic solution it is a good testing ground for checking our simulation is correct. The potential for the quantum harmonic oscillator can be parametrised as:

$$V(x) = \frac{1}{2}\mu^2 x^2 \tag{109}$$

so that

$$H_{HMC}(\boldsymbol{q}, \boldsymbol{p}) = \sum_{i=0}^{N-1} \frac{p_i^2}{2} + \sum_{i=0}^{N-1} a \left[\frac{1}{2} m \left(\frac{q_{i+1} - q_i}{a} \right)^2 + \frac{1}{2}\mu^2 q_i^2 \right] \tag{110}$$

where $\mu \in \mathbb{R}$.

Typical Configuration Figure 3 shows a typical configuration for the HMC simulation of the harmonic oscillator. This is just a plot of the position variable at each lattice site taken from a configuration generated at equilibrium. As the particle moves through imaginary time in the y axis, its position at each lattice site is specified in the configuration, and this gives the path for the particle.

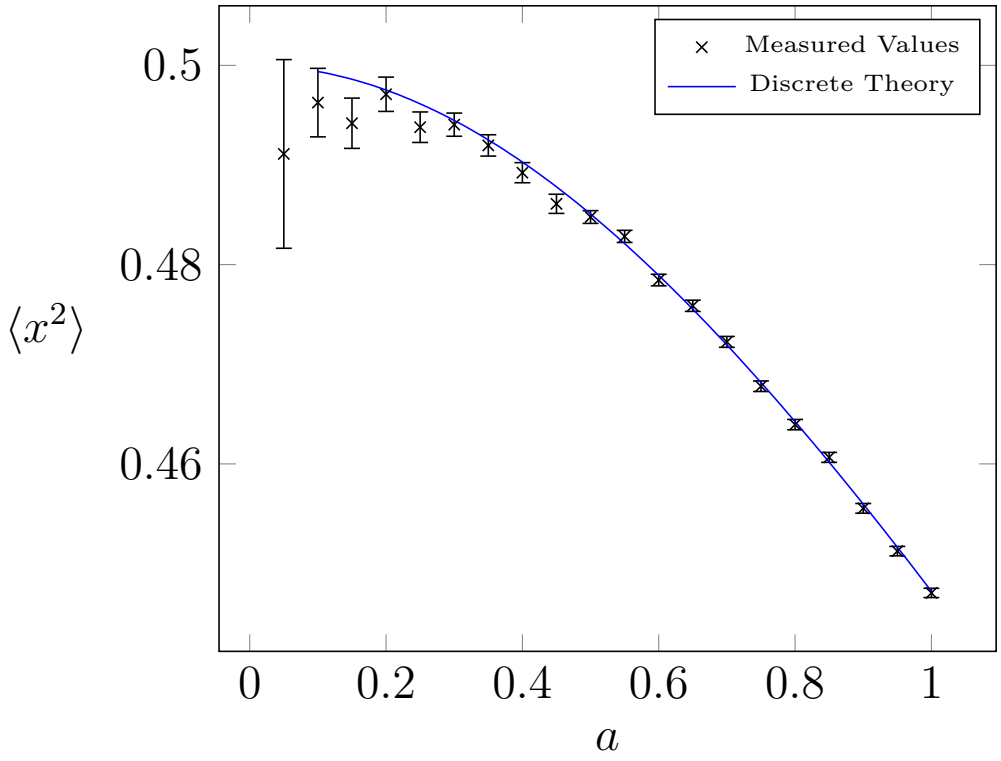


Figure 4. Relationship between lattice spacing and expectation of position squared for the harmonic oscillator with $\mu^2 = 1, m = 1$. Measurements were made on a $N = 100$ site lattice for a range of lattice spacings a.

Mean Square Position The first quantity we calculate is the mean square position which for the statistical mechanical system is $\langle x^2 \rangle$ and corresponds to $\langle 0 | \hat{x}^2 | 0 \rangle$ for the quantum mechanical system. For the discrete quantum theory, this quantity can be calculated exactly for an imaginary time lattice of spacing a and size N to be:

$$\langle x^2 \rangle = \frac{1}{2\mu \left(m + \frac{a^2\mu^2}{4} \right)^{\frac{1}{2}}} \left(\frac{1 + R^N}{1 - R^N} \right), \tag{111}$$

where

$$R = 1 + \frac{a^2\mu^2}{2m} - \frac{a\mu}{\sqrt{m}} \left(1 + \frac{a^2\mu^2}{4m} \right)^{\frac{1}{2}}, \tag{112}$$

see [11], [14], [12] or appendix B for a full derivation of equation 111. Running the HMC simulation at a range of lattice spacings we obtain the results in figure 4. The errors grow as we approach the continuum limit which is due to the fact that for smaller lattice spacing the autocorrelation time increases, so we save samples at a lower frequency and hence for a fixed number of generated configurations we have less samples and so by equation 89, the error is larger.

Energy Levels In order to calculate the ground state energy of the system we use the virial theorem as discussed in section 2.1.1. Applied to the harmonic potential and remembering

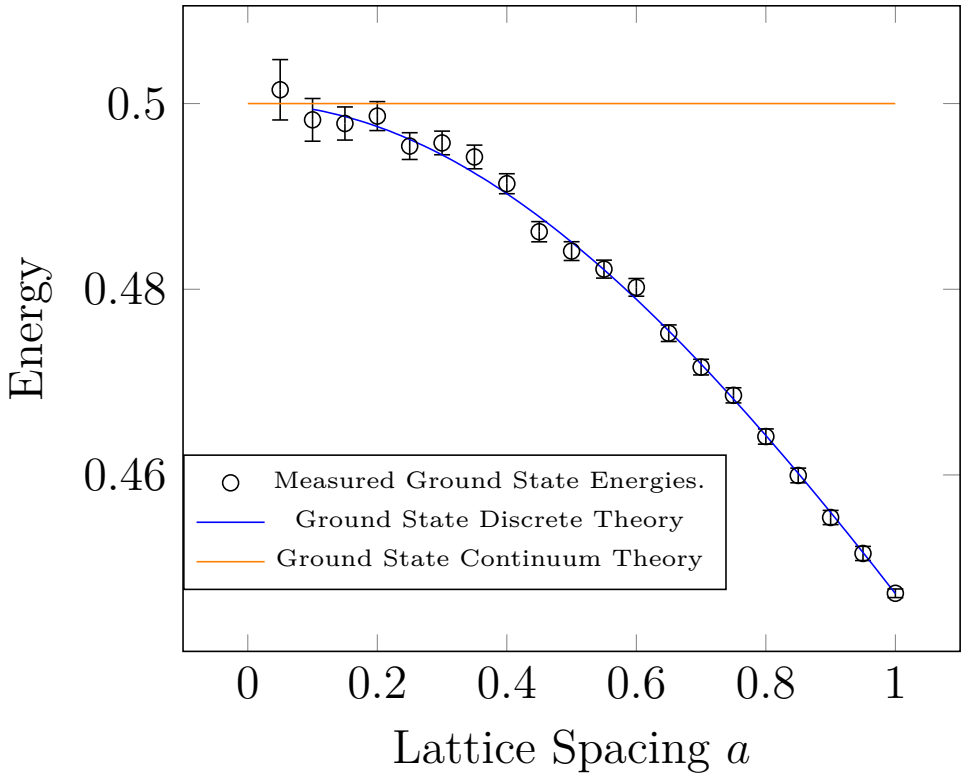


Figure 5. Ground state energies for the harmonic oscillator with $\mu^2 = 1, m = 1$ on a lattice of $N = 100$ sites for a range of lattice spacings a.

that measurements of moments $\langle x^p \rangle$ correspond to quantum expectations in the ground state the virial theorem gives:

$$E_0 = \mu^2 \langle x^2 \rangle \tag{113}$$

Of course when $\mu = 1$ the calculated values for the ground state energy are the same as those for $\langle x^2 \rangle$ in figure 4, however figure 5 shows the ground state energy on a larger lattice with the accompanying discrete and continuum theory results for comparison.

As $a \to 0$ and we approach the continuum we see that E_0 approaches the value of $\frac{1}{2}$ which is given by the continuum formula for the energy levels of a harmonic oscillator:

$$E_n = \mu \left(n + \frac{1}{2} \right), \tag{114}$$

when $n = 0$.

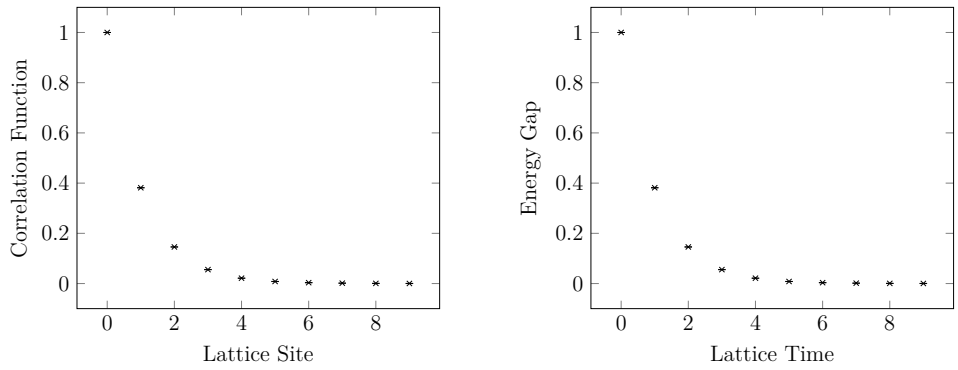
To calculate the energy gap between the first and ground state we use:

get help with this, I do not know why this is true

$$\Delta E \left(\delta\tau \right) = -\frac{1}{\delta\tau} \ln C \left(\delta\tau \right), \tag{115}$$

where:

$$C \left(\delta\tau \right) = \frac{\sum_{i=0}^{N-1} x_i x_{i+\delta\tau}}{\sum_{i=0}^{N-1} x_i^2} \tag{116}$$



(a) Correlation function. (b) Energy gap

Figure 6. Energy gap and correlation function for the harmonic oscillator with $a = 1$, $\mu = 1$ and $m = 1$.

is the *correlation function*. In equation 116 we have utilized the periodic boundary conditions of the lattice and calculated the correlation at each lattice site then averaged over them to increase the statistics.For an derivation of equation 115 see appendix Figures **??** and **??** show the energy gap and correlation function respectively on an HMC simulation of the harmonic oscillator. For small values of $\delta\tau$ we can see that $\Delta E \approx 1$ which is what we expect from equation 114. By averaging over the values for which $\Delta E \approx 1$ we can estimate the the first excited energies by subtracting the estimated ground state energy for various values of μ^2. These have been plotted in figure **??** alongside the ground state energies for a selection of lattice spacings along with the results from the discrete and continuous theory.

Is this okay?

Insert reference to appendix.

Why doesn't it stay constant?

Make data for the plot and include the plot

Ground State Probability Density Function Our method for measuring the density function of the ground state follows that of [11]. We discretise position into K bins of size Δx. On any given configuration we may then bin the position coordinate at each lattice site and create a histogram. Normalising the histogram gives the probability mass density for finding the particle in any interval $[x, x + \Delta x]$ and as $\Delta x \to 0$, this becomes the probability density function that is the modulus squared of the wave function. Since we know that measurements made in our simulation correspond to ground state expectation values, this density function is that of the ground state. It can be shown (see appendix B or [11]) that for the discrete quantum harmonic oscillator the ground state density function is given exactly by:

$$|\psi(x)|^2 = \left(\frac{\sqrt{5}}{2\pi}\right)^{\frac{1}{2}} \exp\left(-\frac{\sqrt{5}}{2}x^2\right) \tag{117}$$

when $m = \mu = 1$ and with a lattice spacing of $a = 1$, which should be contrast with the continuum result when $a \to 0$:

$$|\psi(x)|^2 = \frac{1}{\sqrt{\pi}} \exp\left(-x^2\right). \tag{118}$$

Plots of the density functions for the discrete and continuum cases are shown in figure 8 along with the simulation results at the given parameters. There is a divergence from the

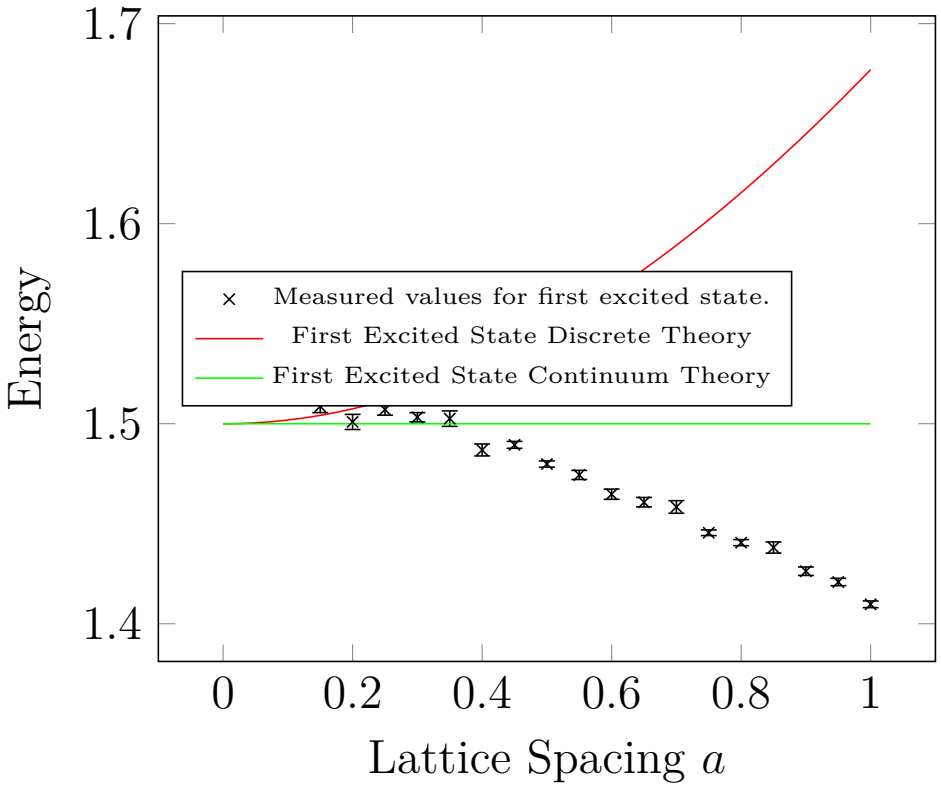


Figure 7. First excited state of the harmonic oscillator on a 1000 site lattice of varying spacings.

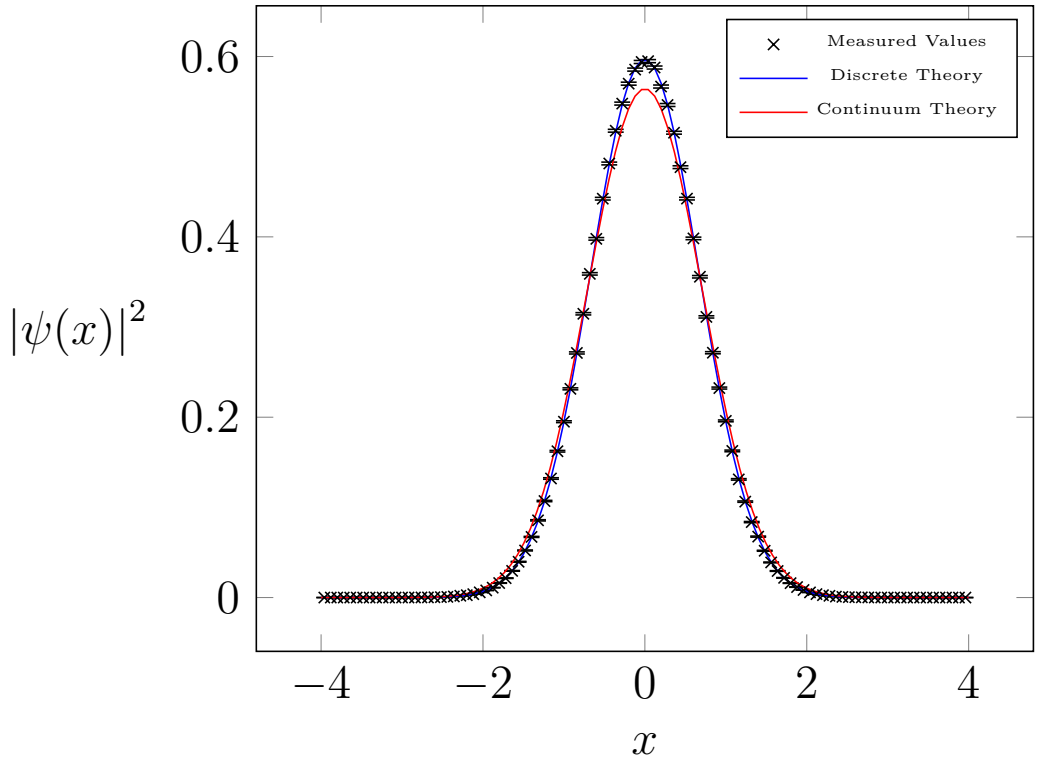


Figure 8. Continuum, discrete and measured ground state density functions for the harmonic oscillator with $\mu^2 = 1, m = 1$ on a lattice of $N = 100$ sites at spacing $a = 1$.

continuum result which is due to the finite lattice spacing of $a = 1$ in this simulation. As $a \to 0$ the discrete theory and simulation results will converge on the continuum ground state density function.

4.1.2 Quantum Anharmonic Oscillator

Since the anharmonic oscillator does not have an analytic solution for the properties we wish to calculate, it is a good first application of out HMC simulation for calculations we do not already know the answer to. Rather than use the more common form of the anharmonic quartic potential:

$$V(x) = \frac{1}{2} m\mu^2 x^2 + \lambda x^4 \tag{119}$$

where $\mu^2 \in \mathbb{R}$ and $\lambda \in \mathbb{R}_{>0}$, we use the parameterisation:

$$V(x) = \lambda \left(x^2 - f^2\right)^2 \tag{120}$$

with $\lambda \in \mathbb{R}_{>0}$ and $f^2 \in \mathbb{R}$. This form of the potential has the advantage that its zeros coincide with its minima at $x = \pm f$ when $f > 0$ which can be seen in figure 9 that illustrates the symmetric *double well* structure of the system. Since equation 120 is equivalent to equation 119 up to the additive constant λf^4, the Euclidean action and consequently

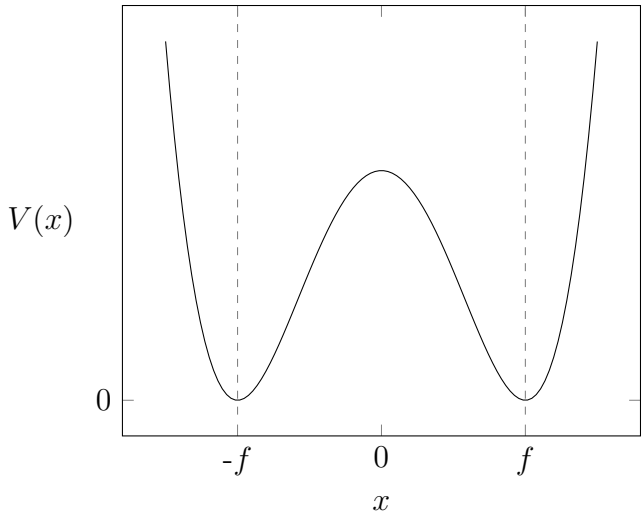


Figure 9. Anharmonic oscillator double well potential with the parameterisation $V(x) = \lambda\left(x^2 - f^2\right)^2$.

the HMC Hamiltonian for the system is the same up to the additive constant λf^4. The constant in the Hamiltonian vanishes when we take its derivatives in Hamilton's equations, and any constant term in the action amounts to rescaling the path integral in equation 1 by a complex exponential factor, which will therefore vanish when we take the magnitude to get the probability, so this re-parameterisation of the potential is valid. Of course, this re-parameterisation will have the effect of trivially rescaling the energy eigenvalues of the system by the additive constant λf^4.

The HMC Hamiltonian for this system is given by:

$$H_{HMC}\left(\boldsymbol{q},\boldsymbol{p}\right) = \sum_{i=0}^{N-1}\frac{p_i^2}{2} + \sum_{i=0}^{N-1} a\left[\frac{1}{2}m\left(\frac{q_{i+1}-q_i}{a}\right)^2 + \lambda\left(q_i^2 - f^2\right)^2\right]. \tag{121}$$

Typical configuration Figure 10 shows a typical configuration for the HMC simulation of the anharmonic oscillator. As the particle moves up the y axis in imaginary time we now observe the tunnelling behaviour between different lattice sites in different minima we expect for a quantum double well system, that is the particle is either localised in one well at f, or the other at $-f$ and is very rarely outside of these regions.

Energy Levels In order to calculate the ground state energy we proceed with the same method we used in the case of the harmonic oscillator in section 4.1.1 and use the virial theorem of equation 33 which gives:

$$E_0 = 3\lambda\left\langle x^4\right\rangle - 4\lambda f^2\left\langle x^2\right\rangle + \lambda f^4. \tag{122}$$

To calculate the first excited state we. The results of the HMC simulation of the anharmonic oscillator for the ground and first excited states are shown in 11 for a range of values of f^2.

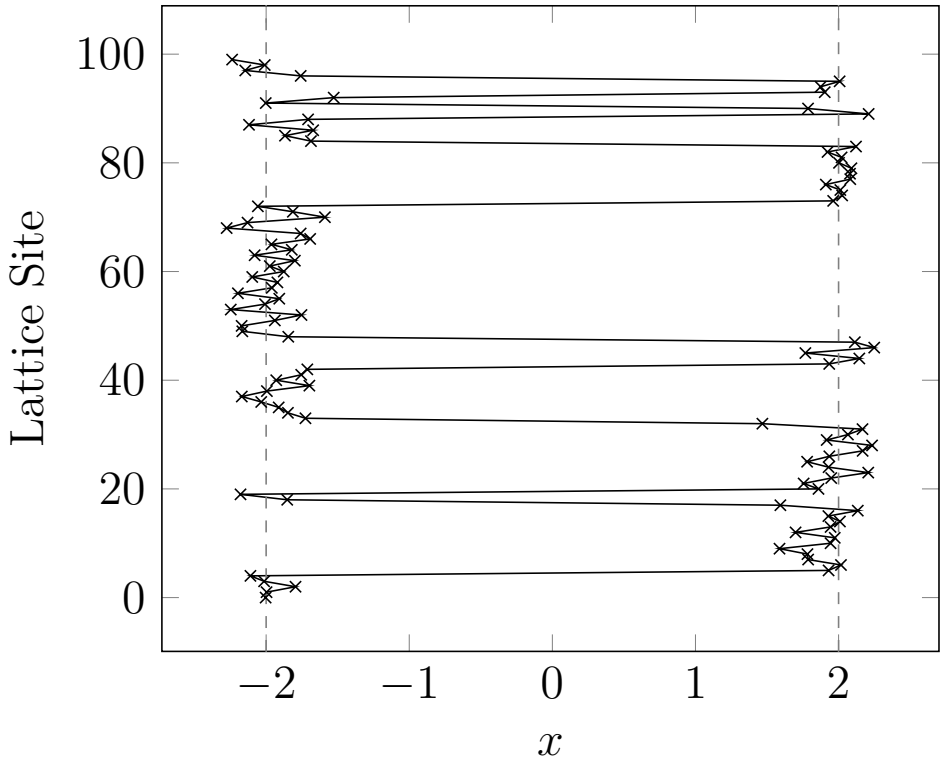


Figure 10. Typical configuration for the anharmonic oscillator with $\lambda = 1, f^2 = 4, m = 1$ on a lattice of $N = 100$ sites at spacing $a = 1$.

Since there is no analytic solution to the anharmonic oscillator, the continuum values in figure 11 are from [15] where approximations of the exact energy eigenvalues are given. Our results follow the same pattern as those given in [15], however since we are working with finite lattice spacings we should expect some divergence. The deviation from the approximated exact values for larger values of f^2 at the chosen lattice spacing matches the deviation given in [13] where the Metropolis algorithm is used for the same calculations.

Ground State Probability Density Function To measure the ground state probability density function we follow the same method as we did for the harmonic oscillator in section 4.1.1. The results can be seen in figure 12 where we now observe the double Gaussian type behaviour we would expect; two regions of high probability around the minima of the potential separated by a region of lower probability, i.e. a high probability of finding the particle in either well and low probability of finding it anywhere else.

4.1.3 Isolated Modes

In figure 10 we saw the tunnelling behaviour displayed by the anharmonic oscillator as the particle tunnelled between the minima of the symmetric potential wells at $x = \pm f$ along its path in imaginary time. In figure 13 we investigate the effect of increasing the value of f^2

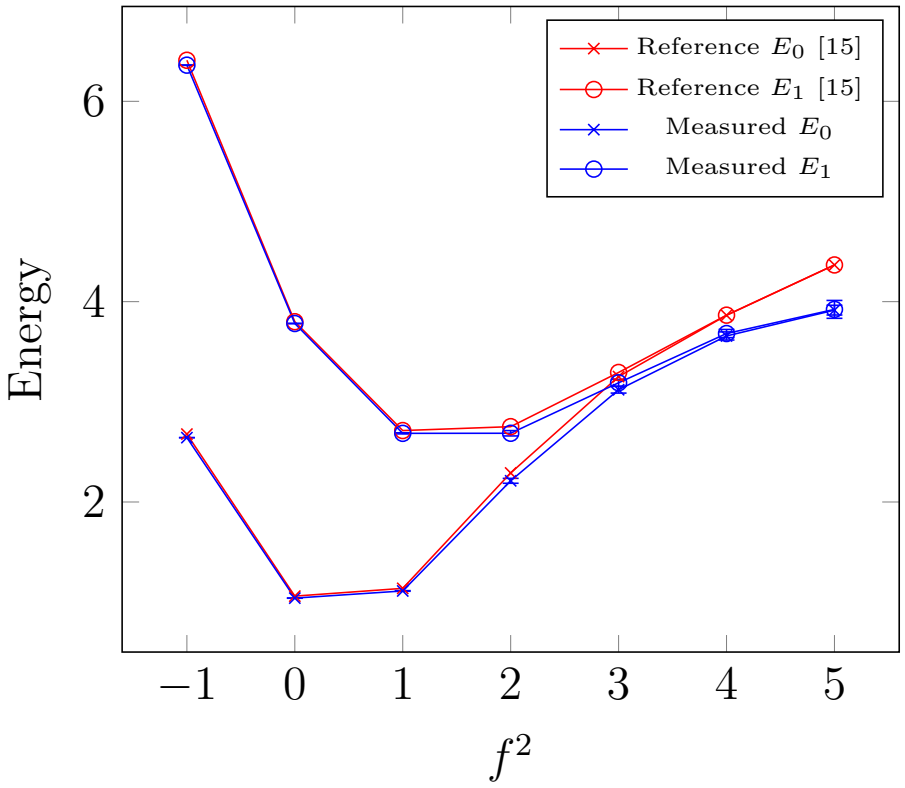


Figure 11. Ground and first excited state energies for the anharmonic oscillator with $\lambda = 1, m = 1$ on a lattice of $N = 1000$ sites at spacing $a = 0.1$. Values are shown for a range of f^2 values.

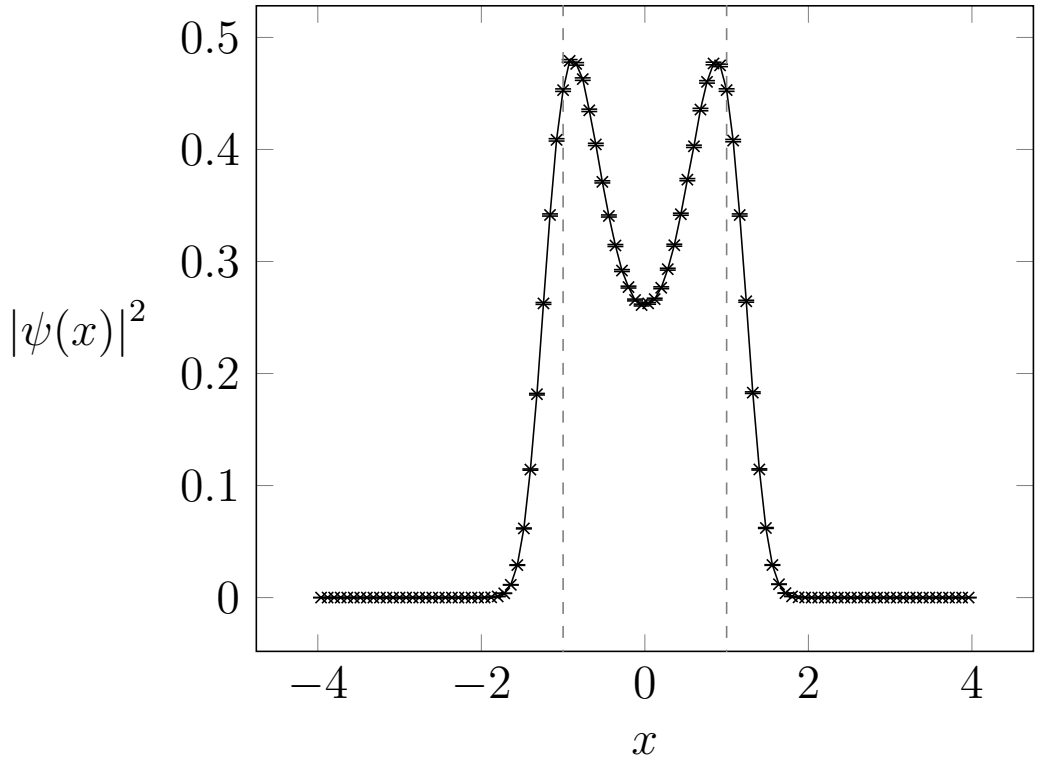
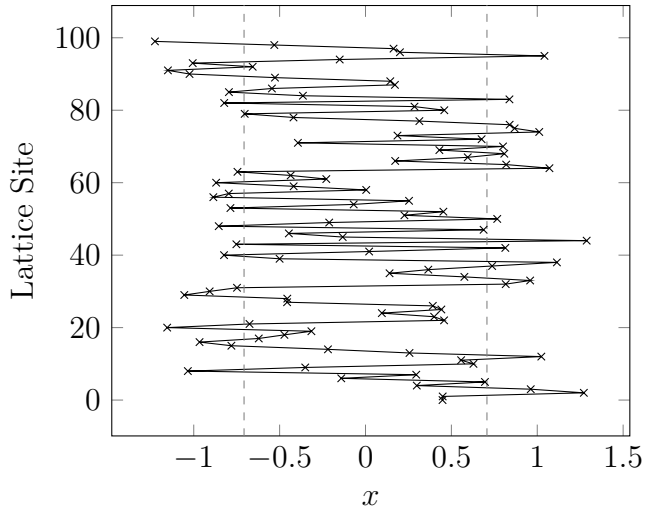


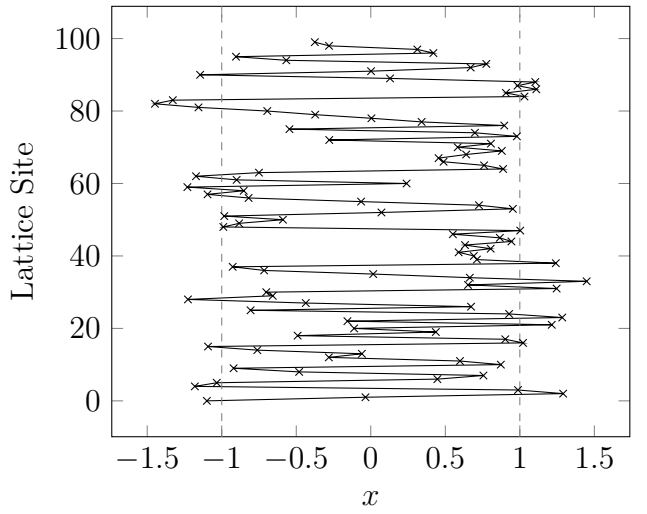
Figure 12. Measured ground state density function for the anharmonic oscillator with $\lambda = 1, f^2 = 1, m = 1$ on a lattice of $N = 100$ sites at spacing $a = 1$.

which increases the distance between the centre of the wells and increases their depth. We can clearly see that even a small increase in f^2 results in less tunnelling along the particles path on a given configuration and lattice variables tend to cluster together in one well or the other, due to the coupling term in the Euclidean action. This in itself is not a problem; measurements made on a configuration where more lattice variables lie in one potential well than the other do contribute a bias value to the statistical average in equation 39 for quantities such as $\langle x \rangle$. However, as long as subsequently generated configurations upon which we make measurements are uncorrelated, and the lattice variables x_i have been able to tunnel between the minima at $\pm f$, these bias contributions will cancel.

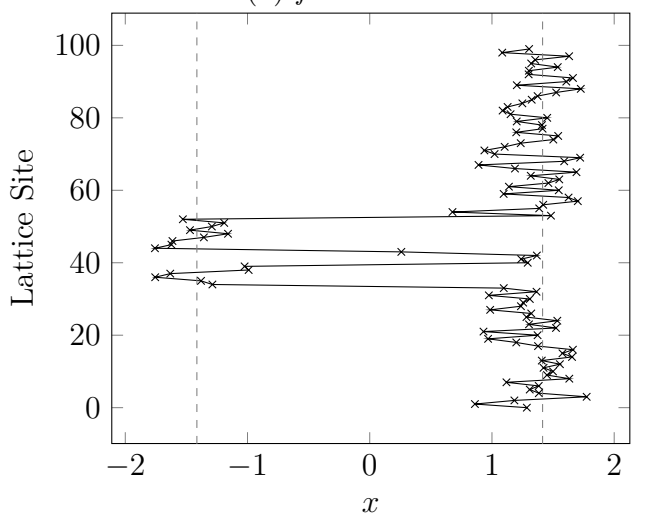
We found that for $f^2 \geq 2$ and keeping $\lambda = 1$ fixed the lattice variables x_i had difficulty moving between the minima of the potential on proposal configurations and hence subsequently generated configurations were highly correlated. In our simulation this is the problem of isolated modes; when lattice variables x_i on subsequent configurations in the HMC algorithm have been unable to move out of the well they were in in the original configuration, and into the other well in the proposal configuration. This problem manifests itself in various ways in our simulation. One good indicator that we have had a tunnelling problem are non zero (to within the error bar) estimates of $\langle x \rangle$ since by symmetry the quantum particles mean position should be zero. Another indicator of a tunnelling problem is an asymmetric ground state density function $|\psi(x)|^2$ that appears to favour the particle being in one well more than the other. These anomalous results are due to the fact that we may be stuck



(a) $f^2 = 0.5$



(b) $f^2 = 1$

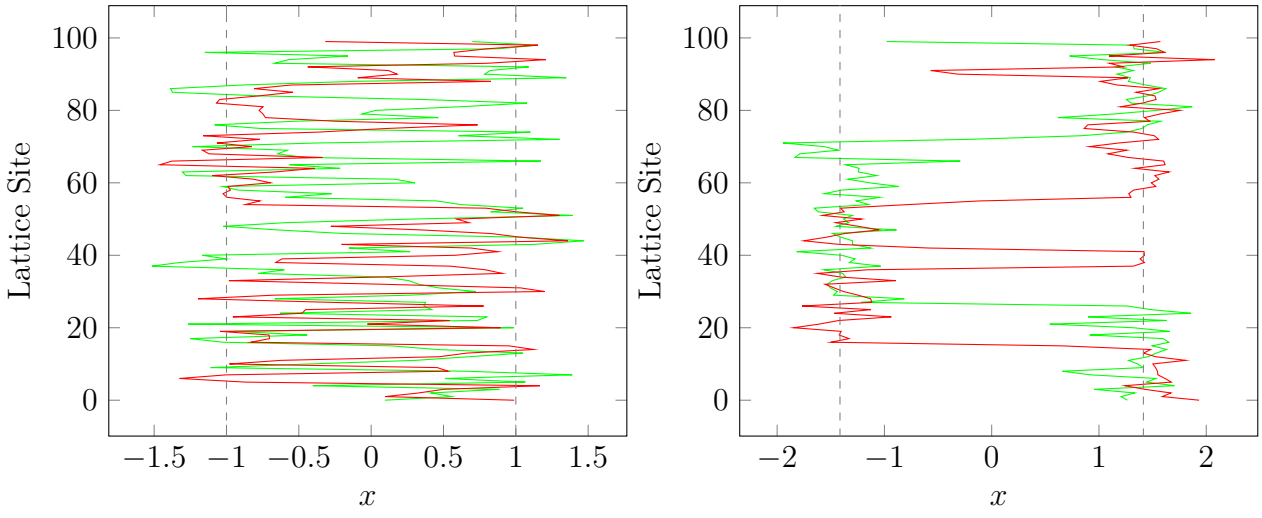


(c) $f^2 = 2.0$

Figure 13. Typical configurations for a anharmonic oscillator with $\lambda = 1, m = 1$ on a lattice of $N = 100$ sites at spacing $a = 1$ when varying f^2.

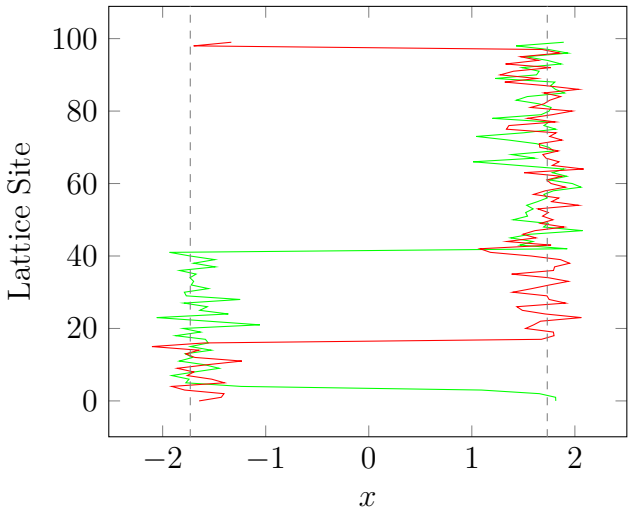
generating configurations where more lattice variables are in one minima than the other; due to the fact individual lattice variables have difficulty tunnelling, if on a given configuration more lattice variables are in one minima than the other, then since the variables have difficulty tunnelling, subsequent configurations will also have this same bias and so estimates of $\langle x \rangle$ and $|\psi(x)|^2$ will be wrong. It should however be noted that it is possible to have estimates of $\langle x \rangle$ close to zero and symmetric density functions $|\psi(x)|^2$ and still have highly correlated configurations resulting from a tunnelling problem. This can arise if initially for a configuration in equilibrium, half the lattice variables are trapped in one well and half in another, if there is no tunnelling of these lattice variables on subsequently generated configurations, then this will lead to a symmetric result, however the subsequent configurations will be highly correlated since we are not exploring the entire configuration space. Typically measuring the value of the ground state energy E_0 and energy gap ΔE will not necessarily give an indication of a tunnelling problem for the symmetric anharmonic potential. In the case of E_0 as calculated via the virial theorem in equation 122, up to the additive constant the value depends only on the second and fourth moments of the position, and is therefore insensitive to the change of sign that is associated with the tunnelling between symmetric wells. As long as configurations are measured at equilibrium values of $\langle x^2 \rangle$ and $\langle x^4 \rangle$ will be correct but with underestimated errors. For the energy gap, due to the fact that lattice variables tend to cluster in the wells because of the coupling term in the Euclidean action, most terms in the correlation function of the form $x_i x_{i+t}$ will also be insensitive to the change of sign for small values of t, and so the effect of the tunnelling problem on the energy gap will be small. Of course we can also measure autocorrelation times which should be very large if these correlated configurations persists throughout the simulation; the inaccuracy in our result would then manifest itself as the error in equation 97. In practice however it is likely that the measured integrated and exponential autocorrelation times will significantly underestimate the correlations between subsequent configurations resulting from the tunnelling problem. To see why this is the case, note that if all lattice variables are stuck in the same well with no tunnelling, then the system knows nothing of the other well, and so measured integrated and exponential autocorrelation times will be much less than than they should be. For the situation when $\langle x \rangle$ is close to zero but there is very little tunnelling, so there are approximately even numbers of lattice variables in each well, but each variable stays trapped in its well on subsequent generated configurations, the system will not be exploring the whole configuration space, and so the measured integrated and exponential autocorrelation times will be underestimated. We can also observe the difference in the lattice configurations generated at the start (after the equilibration period) and end of the simulation to determine qualitatively how bad the correlation really is, and this is shown in figure 14 for a range of f^2 values to demonstrate how the isolated modes become a problem.

We can see that on a typical simulation the first and last equilibrium configurations when $f^2 = 1$ and $f^2 = 2$ look to be uncorrelated suggesting there has been some tunnelling of the lattice variables x_i in both simulations. At $f^2 = 3$ there appears to be a correlation between the first and last configurations. Finally at $f^2 = 4$ it appears qualitatively as if there have been very few instances where a lattice variables have been able to tunnel throughout the entire simulation of 100000 generated configurations; most lattice variables in the final configuration are in the same well they were in in the initial configuration, these

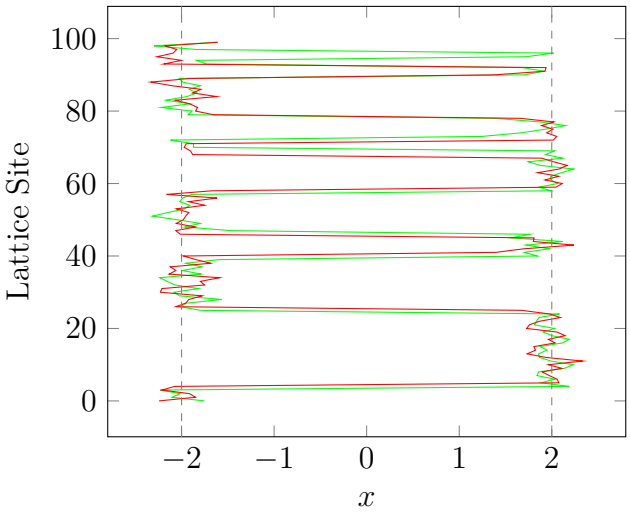


(a) $f^2 = 1$

(b) $f^2 = 2$



(c) $f^2 = 3$



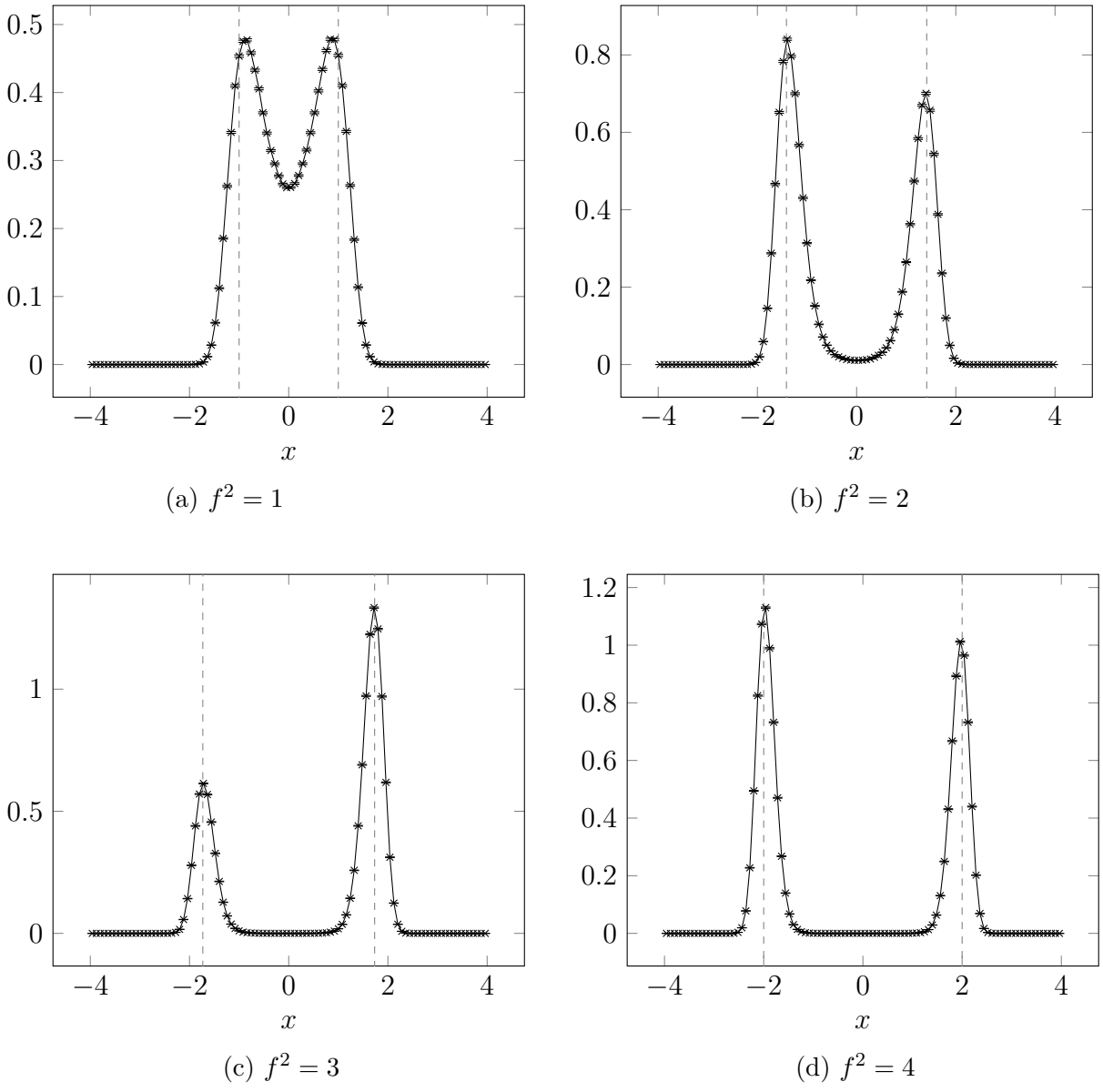
(d) $f^2 = 4$

Figure 14. Typical first equilibrium configuration in green and final configuration after 100000 iterations of HMC algorithm in red for anharmonic oscillator with $\lambda = 1, m = 1$ at a range of f^2 values on a lattice of $N = 100$ sites at spacing $a = 1$.

configurations are therefore highly correlated.

When $f^2 = 1$, $\langle x \rangle = 0.000760909 \pm 0.000930068$ so to within one error bar the value is correct, we also have the ground state density function shown in figure 15a which is symmetric, so tunnelling of lattice variables between configurations for which we made measurements has not been a problem. In the case at $f^2 = 2$, $\langle x \rangle = -0.116344 \pm 0.00317193$ and the measured density function $|\psi(x)|^2$ is shown in figure 15b. Clearly this result is wrong since the error is two orders of magnitude smaller than the correction needed in the estimate and the density function is asymmetric. So, despite the fact that when $f^2 = 2$ the first and last configurations in the trajectory look to be fairly uncorrelated, there has been a tunnelling problem in subsequent measured configurations generated in the simulation that have lead to a bias where more lattice variables have been stuck on the well at $-\sqrt{2}$. For $f^2 = 3$, $\langle x \rangle = 0.618997 \pm 0.00135925$ with the density function $|\psi(x)|^2$ in figure 15c, suggesting the same problem has occurred as when $f^2 = 2$ except now more lattice variables have been stuck in the minima at $\sqrt{3}$ on subsequently generated configurations. However, when $f^2 = 4$, $\langle x \rangle = -0.102023 \pm 0.000516574$ which is closer to zero than when $f^2 = 3$ with the density function $|\psi(x)|^2$ in figure 15d which is more symmetric, however from figure 14d it looks as if there has been a severe tunnelling problem in this system, with only a few lattice variables tunnelling in the entire simulation. The reason for this seemingly better estimate is that the initial configuration had a more even distribution of lattice variables at $\pm f$ and so despite the fact that there is no tunnelling, we get a better estimate of $\langle x \rangle$ and a more symmetric density function $|\psi(x)|^2$. The errors in the above estimates where we have identified a tunnelling problem are of course all too small due to the high correlation between measured results and our naive method of calculating error. In general we found that these *lucky* cases when there is an even distribution of lattice variables in the wells at $\pm f$ were fairly common in our simulation. This is due to the fact that we used the initial conditions of uniformly distributing the lattice variables on the interval $[-1, +1]$, so by the the end of the equilibration period even proportions of the variables will have moved into the wells at $\pm f$ via the molecular dynamics. One method to avoid this problem is to start the simulation with all lattice variables in one well, then if there is a severe tunnelling problem in the simulation we will find that $\langle x \rangle \approx \pm f$ depending on which well we chose and a ground state density that is entirely asymmetric. This method enables us to identify cases when isolated modes are a severe problem for the system.

It should be noted here that the correlation between subsequent configurations is not due to proposals being rejected (although this would lead to an increase in correlation between configurations), indeed even though figure 14 shows only the first and last configurations, we can see that the lattice variables have moved about within the minima they are stuck in. The problem is that the molecular dynamics which evolves the configuration to the proposed state evolves $\boldsymbol{q}$ in an energy landscape given by the potential in the HMC algorithm that is the Euclidean action. This action places each lattice variable in its own double well potential but also couples it to its nearest neighbours, and so for a deep and wide enough potential e.g. when $f^2 = 4$, the probability that the randomised momenta $\boldsymbol{p}$ are large enough to move the variables between the wells is very small and the molecular dynamics will tend to propose update configurations with each lattice variable in the same well it started in. The proposed



(a) $f^2 = 1$ (b) $f^2 = 2$

(c) $f^2 = 3$ (d) $f^2 = 4$

Figure 15. Ground state density functions when varying f^2 for the simulations corresponding to figure 14.

configurations will still likely be accepted since they are in a region of low euclidean action and therefore have a high probability of acceptance.

An obvious but naive solution to the problem of isolated modes in our quantum system is to simply run a much longer simulation and take samples at a lower frequency. For a system where some tunnelling does occur, such as when $f^2 = 2$ this may be feasible, since the probability of a lattice variable tunnelling is non-negligible, so if we wait for a long enough time between measurements, our configurations will become uncorrelated. However, for a system where the tunnelling is a much greater problem, such as when $f^2 = 4$ figure 14d suggests in 100000 configuration updates, only a few tunnelling events occurred. Clearly running longer simulations at a lower sampling frequency will be infeasible. In the next section we explore a method to deal with this problem more efficiently.

4.2 Accelerated Dynamics (Tempering)

4.2.1 Tempering in HMC

We have seen above that HMC much like other MCMC methods [3] has an issue in that it struggles to sample from isolated modes in a probability distribution, such as the wells in the double well potential. To solve this problem various methods, such as *parallel tempering* [27], [28], *simulated tempering* [29], *tempered transitions* [7] and *annealed importance sampling* [30] have been developed where one samples according to a distribution that is more diffuse than the target distribution that has the isolated modes. These methods work by varying the temperature T in the canonical distribution of equation 61, so if $T = 1$ gives our target distributions, increasing T will give a more diffuse one. In HMC we can incorporate tempering directly into the molecular dynamics which proposes update states and here we follow the method given in [3] and [7].

In order to implement tempering on a leapfrog trajectory that is used to propose an update in the HMC algorithm we will follow the symmetric method given in [3]. Defining the tempering parameter α as a number close to but larger than one, multiply each momentum variable before the first half step (equation 46) and after the second half step (equation 48) in the leapfrog update by $\sqrt{\alpha}$ in the first half of the trajectory and then in the second half of the trajectory replace this multiplication with division by $\sqrt{\alpha}$. If there is an odd number of steps in the leapfrog trajectory then on the middle step before the first half step in momenta we multiply by $\sqrt{\alpha}$ and on the second half step we do a corresponding division so that the number of multiplications and divisions by $\sqrt{\alpha}$ is equal.

With tempering introduced the leapfrog equations 46, 47 and 48 for a small time step ϵ to

go from t to $t+\epsilon$ in position and momentum become:

$$p_i\left(t+\epsilon/2\right) = \sqrt{\alpha}p_i\left(t\right) - \epsilon/2\frac{\partial U}{\partial q_i}\left(\boldsymbol{q}\left(t\right)\right), \tag{123}$$

$$q_i\left(t+\epsilon\right) = q_i\left(t\right) + \epsilon\frac{\partial K}{\partial p_i}\left(\boldsymbol{p}\left(t+\epsilon/2\right)\right) \tag{124}$$

$$p_i\left(t+\epsilon\right) = \sqrt{\alpha}\left(p_i\left(t+\epsilon/2\right) - \epsilon/2\frac{\partial U}{\partial q_i}\left(\boldsymbol{q}\left(t+\epsilon\right)\right)\right), \tag{125}$$

for steps in the first half of the trajectory and

$$p_i\left(t+\epsilon/2\right) = \frac{1}{\sqrt{\alpha}}p_i\left(t\right) - \epsilon/2\frac{\partial U}{\partial q_i}\left(\boldsymbol{q}\left(t\right)\right), \tag{126}$$

$$q_i\left(t+\epsilon\right) = q_i\left(t\right) + \epsilon\frac{\partial K}{\partial p_i}\left(\boldsymbol{p}\left(t+\epsilon/2\right)\right) \tag{127}$$

$$p_i\left(t+\epsilon\right) = \frac{1}{\sqrt{\alpha}}\left(p_i\left(t+\epsilon/2\right) - \epsilon/2\frac{\partial U}{\partial q_i}\left(\boldsymbol{q}\left(t+\epsilon\right)\right)\right), \tag{128}$$

in the second half of the trajectory, with the possible exception that for an odd number of leapfrog steps on the middle step:

$$p_i\left(t+\epsilon/2\right) = \sqrt{\alpha}p_i\left(t\right) - \epsilon/2\frac{\partial U}{\partial q_i}\left(\boldsymbol{q}\left(t\right)\right), \tag{129}$$

$$q_i\left(t+\epsilon\right) = q_i\left(t\right) + \epsilon\frac{\partial K}{\partial p_i}\left(\boldsymbol{p}\left(t+\epsilon/2\right)\right) \tag{130}$$

$$p_i\left(t+\epsilon\right) = \frac{1}{\sqrt{\alpha}}\left(p_i\left(t+\epsilon/2\right) - \epsilon/2\frac{\partial U}{\partial q_i}\left(\boldsymbol{q}\left(t+\epsilon\right)\right)\right). \tag{131}$$

4.2.2 Volume Preservation Under Tempering Dynamics

Just as in section 3.2.1 we still require that volume is preserved by the leapfrog integration scheme since this is required for detailed balance, even with tempering present. The argument carries over from the non-tempered case, however on any given leapfrog step the determinant of the Jacobian will not necessarily be 1 and therefore volume will not be preserved; it will only be the transformation for the whole trajectory that preserves volume. To see why this is the case note that as before we can define the mappings on phase space $\mathcal{T}_{p_1}\left(\epsilon,\alpha\right):\left(\boldsymbol{q},\boldsymbol{p}\right)\rightarrow\left(\boldsymbol{q}',\boldsymbol{p}'\right)$, $\mathcal{T}_q\left(\epsilon\right):\left(\boldsymbol{q},\boldsymbol{p}\right)\rightarrow\left(\boldsymbol{q}',\boldsymbol{p}'\right)$ (this is the same map as in the non-tempered case) and $\mathcal{T}_{p_2}\left(\epsilon,\alpha\right):\left(\boldsymbol{q},\boldsymbol{p}\right)\rightarrow\left(\boldsymbol{q}',\boldsymbol{p}'\right)$ such that:

$$\mathcal{T}_{p_1}\left(\epsilon,\alpha\right)\begin{pmatrix}\boldsymbol{q}\\\boldsymbol{p}\end{pmatrix} = \begin{pmatrix}\boldsymbol{q}\\\sqrt{\alpha}\boldsymbol{p} - \epsilon\underline{\nabla}_q U\left(\boldsymbol{q}\right)\end{pmatrix}, \tag{132}$$

$$\mathcal{T}_q\left(\epsilon\right)\begin{pmatrix}\boldsymbol{q}\\\boldsymbol{p}\end{pmatrix} = \begin{pmatrix}\boldsymbol{q} + \epsilon\underline{\nabla}_p K\left(\boldsymbol{p}\right)\\\boldsymbol{p}\end{pmatrix}, \tag{133}$$

$$\mathcal{T}_{p_2}\left(\epsilon,\alpha\right)\begin{pmatrix}\boldsymbol{q}\\\boldsymbol{p}\end{pmatrix} = \begin{pmatrix}\boldsymbol{q}\\\sqrt{\alpha}\left(\boldsymbol{p} - \epsilon\underline{\nabla}_q U\left(\boldsymbol{q}\right)\right)\end{pmatrix}, \tag{134}$$

where we have now had to define three maps due to the fact the tempering multiplication happens before and after the half steps in the momenta so the forms of the half steps in momentum in the leapfrog equations are no longer the same. Having defined these mappings we can calculate their Jacobians to be:

$$J_{p_1} = \det \begin{pmatrix} \delta_{ij} & 0 \\ -\epsilon \partial q_i \partial q_j U(\boldsymbol{q}) & \sqrt{\alpha}\delta_{ij} \end{pmatrix} = \alpha^{\frac{d}{2}} \tag{135}$$

$$J_q = \det \begin{pmatrix} \delta_{ij} & \epsilon \partial p_i \partial p_j K(\boldsymbol{p}) \\ 0 & \delta_{ij} \end{pmatrix} = 1 \tag{136}$$

$$J_{p_2} = \det \begin{pmatrix} \delta_{ij} & 0 \\ -\sqrt{\alpha}\epsilon \partial q_i \partial q_j U(\boldsymbol{q}) & \sqrt{\alpha}\delta_{ij} \end{pmatrix} = \alpha^{\frac{d}{2}}. \tag{137}$$

Let us assume that we have an even number l of leapfrog steps in our trajectory. A leapfrog step in the first half of the trajectory can be written in terms of the above mappings as $\mathcal{T}(\alpha, \epsilon) = \mathcal{T}_{p_2}(\alpha, \epsilon/2) \circ \mathcal{T}_q(\epsilon) \circ \mathcal{T}_{p_1}(\alpha, \epsilon/2)$, which from the compositions of the above maps will have Jacobian α^d and in the second half trajectory as $\mathcal{T}(1/\alpha, \epsilon) = \mathcal{T}_{p_2}(1/\alpha, \epsilon/2) \circ \mathcal{T}_q(\epsilon) \circ \mathcal{T}_{p_1}(1/\alpha, \epsilon/2)$ which will have Jacobian $1/\alpha^d$. The mapping which via the leapfrog equations takes us from the start to the end of the trajectory can be written as:

$$\text{traj}(\alpha, \epsilon, l) = [\mathcal{T}(1/\alpha, \epsilon)]^{\frac{l}{2}} \circ [\mathcal{T}(\alpha, \epsilon)]^{\frac{l}{2}} \tag{138}$$

which using the Jacobians of the mappings in the composition will have Jacobian $J_{\text{traj}} = \left[\alpha^d\right]^{\frac{l}{2}} \left[1/\alpha^d\right]^{\frac{l}{2}} = 1$ and so as in the case of non-tempered dynamics we have preservation of volume on phase space if we integrate Hamilton's equations via the leapfrog scheme and include a tempering factor. Note that for an odd number of steps l we would have $\text{traj}(\alpha, \epsilon, l) = [\mathcal{T}(1/\alpha, \epsilon)]^{\frac{l-1}{2}} \circ \mathcal{T}_{p_2}(1/\alpha, \epsilon/2) \circ \mathcal{T}_q(\epsilon) \circ \mathcal{T}_{p_1}(\alpha, \epsilon/2) \circ [\mathcal{T}(\alpha, \epsilon)]^{\frac{l-1}{2}}$, however the factors of alpha will still cancel in the Jacobian and so volume is still preserved.

The multiplication of the momenta variables by $\sqrt{\alpha}$ will increase the value of the HMC Hamiltonian H_{HMC}, so that if at the beginning of the trajectory H_{HMC} has a typical value for the canonical distribution when $T = 1$ (which it will if taken at equilibrium), then after the multiplication H_{HMC} will have a typical value for a higher T; this is how the "tempering" arises from the dynamics. As the momenta $\boldsymbol{p}$ are increased there will be an increase in the kinetic term $K(\boldsymbol{p})$, which will lead to an increase in $H_{HMC}(\boldsymbol{q}, \boldsymbol{p}) = K(\boldsymbol{p}) + U(\boldsymbol{q})$. However, successive leapfrog steps will distribute the increase in H_{HMC} between K and U [3], this gives a more diffuse distribution for $\boldsymbol{q}$ than the original one at $T = 1$. In fact, following lots of tempered steps $\boldsymbol{q}$ values can be reached which would have been very improbable in the target distribution. In the second half of the trajectory divisions by $\sqrt{\alpha}$ return H_{HMC} to values for the original typical distribution at $T = 1$, but the $\boldsymbol{q}$ may now be in a different region of the probability distribution than before, so we may be able to tunnel between modes of the distribution separated by regions of low probability.

Applying this idea to our quantum mechanical double well potential we remember that our variables of interests are the lattice configurations, i.e. $\boldsymbol{q} \equiv \boldsymbol{x} = (x_0, \ldots x_{N-1})$ and the problem is that for a deep and wide enough well i.e. $f > 1$, the individual x_i variables have

difficulty tunnelling between the wells when we generate a proposal configuration, this leads to a large correlation between subsequent samples and hence a bad estimate on observables. We have the HMC Hamiltonian for the double well system given by:

$$H_{HMC}\left(\boldsymbol{q},\boldsymbol{p}\right)=\sum_{i=0}^{N-1}\frac{p_i^2}{2}+\sum_{i=0}^{N-1}a\left[\frac{1}{2}m\left(\frac{q_{i+1}-q_i}{a}\right)^2+\lambda\left(q_i^2-f^2\right)\right], \tag{139}$$

where the mass term in the action couples nearest neighbour variables on the lattice. Ignoring the mass term temporarily will give the Hamiltonian for a system of N an anharmonic oscillators, each moving in its own double well potential, each with momenta p_i and position q_i. As we evolve the system via the leapfrog integration we can imagine that by multiplying each p_i by $\sqrt{\alpha}$ we may increasing the kinetic energy of each oscillator enough so it can move into the other well. This picture is somewhat naive since it neglects the coupling term in the Hamiltonian which means the harmonic oscillator is actually moving in a more complex potential where each x_i is coupled to its nearest neighbours. Nevertheless, this intuitive picture provides a conceptual idea of how tempering would work in our lattice system. Alternatively we can think of the entire configuration $\boldsymbol{x}$ as being in a potential landscape given by the Euclidean action term in the Hamiltonian, where the isolated modes consist of the regions where different x_i move between the wells in the potential. By increasing the Hamiltonian through multiplication of the momentum variables as described above we may be able to move between these regions of the landscape, which correspond to the isolated modes in the distribution. Before the first tempered step we would expect each x_i variable to be localised around $\pm f$, as we begin to make tempered steps in the molecular dynamics trajectory we expect to see the variables x_i spread out and move far away from the potential minima by the time we reach half way through the trajectory. Then as we begin the divisions in the second half trajectory we should see the variables move back towards the minima at $\pm f$, however they may move back to a different minima to the one they started in.

In [3] it is argued that for large α, $H\left(\boldsymbol{q}',\boldsymbol{p}'\right)$ will typically be much larger than $H\left(\boldsymbol{q},\boldsymbol{p}\right)$, so the probability of accepting the proposed configuration via the Metropolis step will be small. Here lies a key difficulty in tempering, one needs to provide a large enough α so that the Hamiltonian increases enough on the trajectory such that the system is able to move between the isolated modes, however at the end of the trajectory the Hamiltonian needs to have been reduced again to a value similar to that of where it started if it is to have any chance of being accepted in the Metropolis update. There is a further complication in that increasing the number of leapfrog steps may no longer increase the acceptance rate, since by doing more steps in the leapfrog algorithm we are in effect performing more multiplications by α and so doing more tempering, which may give an even larger Hamiltonian at the end of the trajectory. In principle, a slightly lower acceptance rate in a simulation with tempered HMC than in normal HMC may not be a problem. If one could show that despite the fact that less proposals are accepted, the proposals that are (in tempered HMC) are highly uncorrelated then if the number of accepted proposals is still reasonably high we may get better estimates on our observables than in the non tempered case, where highly correlated samples would lead to larger errors.

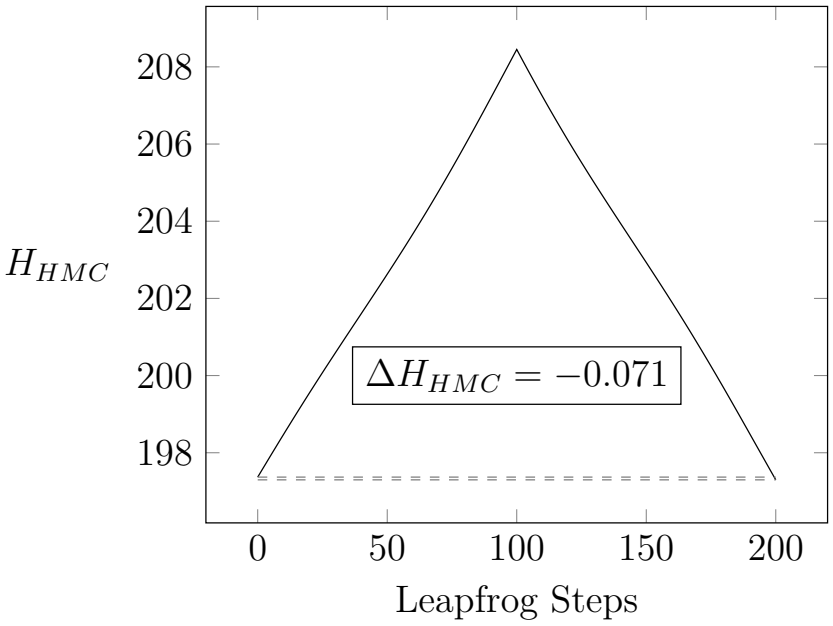
The appeal of tempering as an optimization of HMC is that if it could be successfully implemented in our simulation for a fixed number of leapfrog steps, it would in effect be cost

free. If we were able to find a system in which tunnelling was infrequent or not occurring at all for normal HMC with a number of leapfrog steps l in our simulation, then introduce the tempering parameter to the same system and observe tunnelling for the same number of leapfrog steps l without a significant drop in the acceptance rate, the only cost incurred would have been the multiplication of the momenta variables, which is negligible in comparison to the cost of generating more configurations until the system moves between the modes.

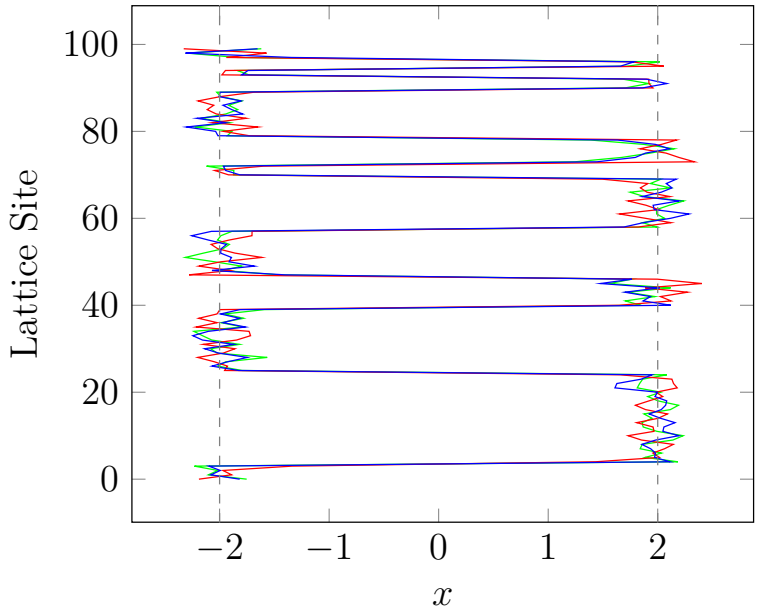
4.2.3 Tempered Simulation Results

After implementing tempered dynamics in our simulation of the anharmonic oscillator we found that rather than observing a better estimates for $\langle x \rangle$ and the ground state density function $|\psi(x)|^2$, as well as a qualitative decorrelation between subsequently generated configurations as in figure 14, we found that the acceptance rate of the algorithm quickly dropped to zero as we increased α. To investigate why this is the case we worked with an anharmonic potential where $\lambda = 1, f^2 = 4$ and $m = 1$ on a lattice of $N = 100$ sites at a spacing of $a = 1$ since we know this system exhibits the issues related to isolated modes, in that the tunnelling probability of lattice sites is very low, see figure 14d. We first evolved our system to an equilibrium configuration via non-tempered dynamics, then from the equilibrium configuration applied tempering dynamics in a leapfrog trajectory for a range of α values. We evolved this same initial configuration for $l = 200$ leapfrog steps of size $\epsilon = 0.005$ so that $\epsilon l = 1$, but were also able to observe the evolution of the Hamiltonian along the trajectory for the range of α values. Due to the complex nature of the potential the system is in, it is not clear how much the Hamiltonian would need to increase in order for it to begin tunnelling the lattice variables x_i, however, by observing the evolution of the configuration at half way through the trajectory we will be able to see if the x_i have been able to move out of their minima.

Figures 16, 17 and 18 show the evolution of the Hamiltonian and the same initial equilibrium configuration under the molecular dynamics for tempering parameters $\alpha = 1.001$, $\alpha = 1.01$ and $\alpha = 1.05$ respectively. In each case during the first half of the trajectory we see the Hamiltonian grow, corresponding to the multiplications by $\sqrt{\alpha}$, then during the second half of the trajectory it shrinks, due to the corresponding divisions by $\sqrt{\alpha}$. In figure 16a with $\alpha = 1.001$ we see the Hamiltonian returns to approximately the same place it started, and indeed in this example there is actually an overall decrease such that for $\Delta H_{HMC} = H_{HMC}(\boldsymbol{q}', \boldsymbol{p}') - H_{HMC}(\boldsymbol{q}, \boldsymbol{p})$, $\Delta H_{HMC} = -0.071$, so the proposed trajectory will be accepted with certainty via the Metropolis update. However, we can see from the plots of the configuration evolution in figure 16b that at the centre of the leapfrog trajectory, where the Hamiltonian is at its maximum value, the lattice variables have been unable to leave the wells they were initially localised in and that at the end of the trajectory, no tunnelling has occurred on the proposed configuration since all lattice variables are in the same wells they started in. Figure 17 shows results for that same initial equilibrium configuration, but now with $\alpha = 1.01$. However, now there is an increase in the Hamiltonian with $\Delta H_{HMC} = 16.005$ which corresponds to an acceptance probability of the order of 10^{-7} which is incredibly low. The issue that makes tempering as suggested in [3] and [7] ineffective in our lattice system lies in this result. In the configuration evolution in figure 17b we can see as in the tempered

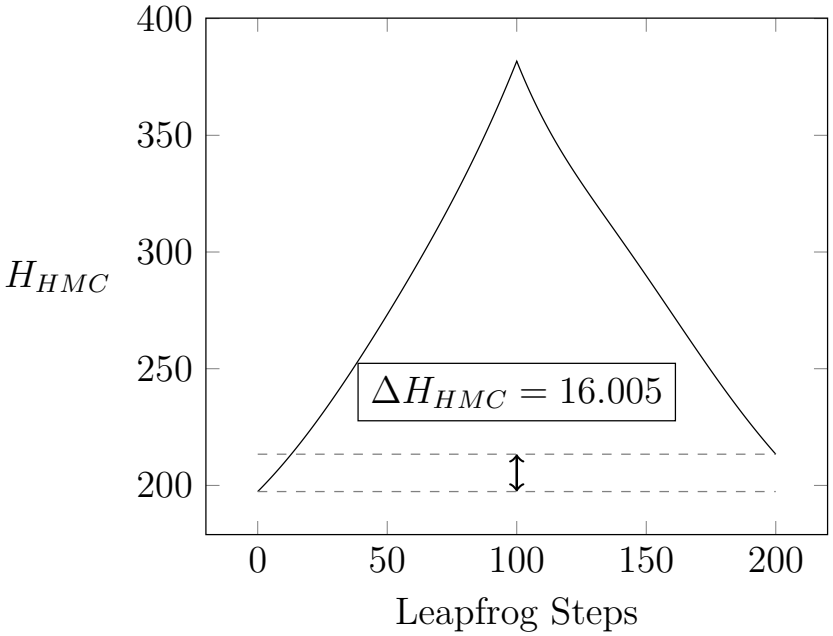


(a) Evolution of Hamiltonian along the leapfrog trajectory.

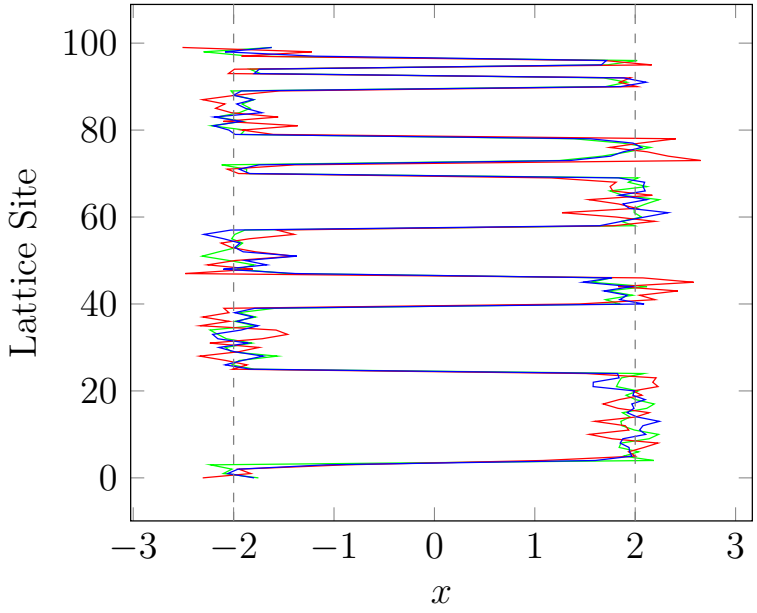


(b) Evolution of the configuration with initial configuration in green, the configuration after 100 leapfrog steps in red and at the end of the trajectory (the proposal configuration) in blue.

Figure 16. Evolution of Hamiltonian and configuration along leapfrog trajectory for tempering parameter $\alpha = 1.001$

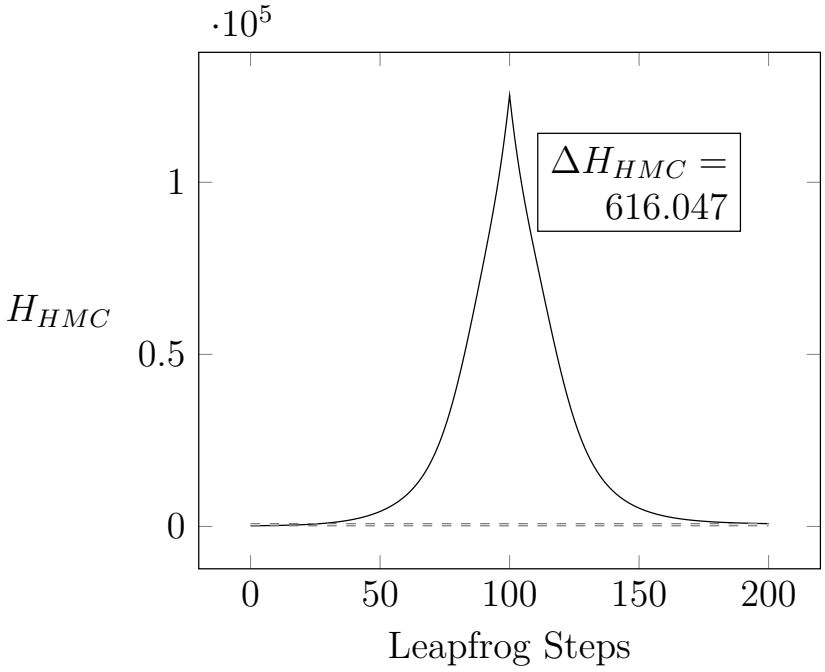


(a) Evolution of Hamiltonian along the leapfrog trajectory.

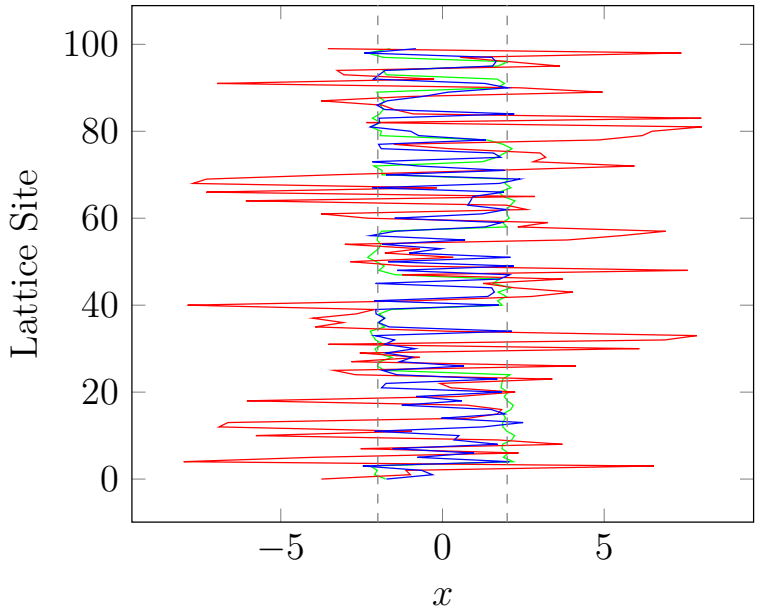


(b) Evolution of the configuration with initial configuration in green, the configuration after 100 leapfrog steps in red and at the end of the trajectory (the proposal configuration) in blue.

Figure 17. Evolution of Hamiltonian and configuration along leapfrog trajectory for tempering parameter $\alpha = 1.01$



(a) Evolution of Hamiltonian along the leapfrog trajectory.



(b) Evolution of the configuration with initial configuration in green, the configuration after 100 leapfrog steps in red and at the end of the trajectory (the proposal configuration) in blue.

Figure 18. Evolution of Hamiltonian and configuration along leapfrog trajectory for tempering parameter $\alpha = 1.05$

case for $\alpha = 1.001$ there has still been no tunnelling despite the larger increase in H_{HMC} that occurs on the trajectory in figure 17a. ΔH_{HMC} on the trajectory is however now too big and typical proposals will not be accepted. So, even for tempering parameters that result in the Hamiltonian being too large at the end of the trajectory for the proposed configuration to be accepted, the lattice variables have still been unable to move between the wells. To see that the intuition around the effects of tempering on our simulation were valid, that is we can move the lattice variables x_i between the minima of the potential for a large enough α, observe figure 18. Here we can see that with $\alpha = 1.05$ the Hamiltonian increases massively in figure 18a and indeed on the middle leapfrog step in figure 18b the lattice variables have been able to move well out of the wells they were localised in on the initial configuration. We can also see that at the end of the trajectory as lattice variables are moved back towards the minima during the divisions by $\sqrt{\alpha}$, variables that were initially in one well have in some cases been brought back down into the other well, so we have induced tunnelling of the lattice variables. However, with $\alpha = 1.05$, $\Delta H = 616.047$ which will give an acceptance rate of zero in the simulation.

Although the above result is for a specific lattice and a very small leapfrog step in order to demonstrate the evolution of the Hamiltonian; this tendency for ΔH_{HMC} to be so large that the probability of accepting a proposed trajectory is always zero for any tempering parameter α that actually induces tunnelling in the system, is a general problem we found for all tempered anharmonic simulations. In [3] it is suggested that in order to maintain the effective temperature i.e. the diffuse distribution at the midpoint of the trajectory and the magnitude of the Hamiltonian at that point, and reduce the tendency for ΔH_{HMC} to be so large, one should reduce α to $\alpha^{\frac{1}{R}}$ and increase the number of leapfrog steps by a factor of R. In practice we found this suggestion had no effect on the system for which we applied tempered dynamics. The idea of increasing the number of leapfrog steps until the tempering is successful raises another issue; There is a computational cost of performing more leapfrog steps. It may be that for a system that has a low probability of samples moving between isolated modes, the cost of increasing the number of leapfrog steps outweighs the cost of simply generating more configurations and waiting for the system to tunnel by itself then making measurements at a very low frequency. So even if tempering was successful in such a scenario a cost analysis would be required before one could make any claims about its benefits.

The reason that the example of tempering in [3] was shown to be successful where as in our case it has failed is likely due to the difference in the number of degrees of freedom of the systems. The example in [3] is for a system of two degrees of freedom, so tempering both variables with a value of α large enough to move the system between the isolated modes results in a relatively small increase in the Hamiltonian at the centre of the trajectory, which means that during the second half of the trajectory the tempered dynamics is able to decrease the Hamiltonian back into the region it started in. However, in our system we have for a lattice of 100 sites, 100 degrees of freedom. As we have seen, tempering each of these variables with a value of α large enough to induce tunnelling results in a huge increase of the Hamiltonian at the centre of the trajectory due to the fact each variable contributes to the Hamiltonian, and the tendency for the Hamiltonian not to decrease in the second half

of the trajectory as much as it increased in the first half of the trajectory, for large increases in the first half of the trajectory, means there is a very low probability of acceptance.

4.2.4 Suggestions for Future Tempering Investigations

A possible solution to the issue of ΔH_{HMC} always being too large on tempered trajectories that actually induce tunnelling, and a suggestion for future investigations, would be *local tempering*. The huge increase that we currently require in the Hamiltonian at the midpoint of the molecular dynamics trajectory for tunnelling events to occur, and the tendency for the the Hamiltonian not to decrease as much as it increases for large α means the ΔH_{HMC} is too large on the proposed configuration for it to be accepted. This is because our system has a large number of degrees of freedom, corresponding to the lattice variables, so tempering all these variables results in much larger changes in the Hamiltonian. However, if we only applied tempering dynamics to a select few lattice variables, and evolved the rest of the variables via standard leapfrog molecular dynamics, this might have the effect of increasing the Hamiltonian for those tempered variables enough locally that they may tunnel through into another well. Globally however, the Hamiltonian would not have increased as much at the midpoint of the trajectory, since less variables are being tempered, and so it may have the ability to return to the range it started in during the divisions in the second half trajectory. Since tunnelling would only occur with the variables chosen for tempering, one would need to find a method of selecting lattice sites for tempering such that throughout the simulation each lattice site has the possibility of tunnelling on subsequent proposal configurations. Options for this could be choosing sites systematically or randomly on subsequent iterations of the HMC algorithm; one would need to show that volume preservation and hence detailed balance still holds.

5 Conclusion

In the background section we began the report by considering the path integral of quantum mechanics and showed through discretisation of time and a Wick rotation, that it can be considered as a canonical partition function on a periodic lattice with continuous position variables at each lattice site. We went on to discuss the relationship between quantum and statistical expectation values, and saw how quantum expectation values of operators in ground state for a large enough lattice could be calculated via the statistical expectation value under the canonical distribution.

Since in practice the statistical expectation values cannot be calculated analytically we introduced in the methods section the notion of a Monte Carlo estimate which we could use to approximate these expectation values. We then discussed the Hybrid Monte Carlo algorithm, which was used in the simulation, and explained how this algorithm uses fictitious momentum variables and Hamiltonian dynamics to propose update states in the simulation. A proof was given that the leapfrog method used to numerically integrate Hamilton's

equations preserves volume on phase space exactly and using this fact detailed balance for the HMC algorithm was also proved. Due to the correlations that can arise between samples in the Markov chain generated via the HMC algorithm, errors can be underestimated and we discussed how this can be taken into account via the exponential and integrated autocorrelation times.

In the results section we applied the HMC algorithm to the cases of the quantum harmonic and anharmonic oscillators, using it to generate lattice configurations which we then used to estimate the values of observables in the ground state via the Monte Carlo estimate. We found here excellent agreement of our results with the discrete theory in the case of the harmonic oscillator for quantities such as $\langle x^2 \rangle$ and $|\psi(x)|^2$ and discussed the divergence from the continuum result due to finite lattice spacings. Ultimately the harmonic oscillator provided a good method for testing the validity of our simulation against theory. Applying the HMC algorithm to the anharmonic oscillator we were able to obtain the same quantities for the case of the harmonic oscillator, however, since the anharmonic system is not solvable analytically we were only able to compare our estimates of energy eigenvalues E_0, E_1 to approximations of the exact results where we found good agreement. More importantly, we were in the case of the anharmonic oscillator able to identify a problem with isolated modes in our simulation. We found that for deep and wide wells lattice variables could have serious difficult tunnelling between wells on subsequently generated configurations leading to incorrect estimates in our results for quantities such as $\langle x \rangle$ and $|\psi(x)|^2$.

As a potential solution to the problem of isolated modes in our simulation we investigated the effect of introducing a tempering parameter α into our simulation with the aim of increasing the tunnelling of our lattice variables. It was explained how this tempering parameter is incorporated directly into the leapfrog integration scheme and importantly we proved that volume preservation still holds for tempered dynamics. This is an important proof since without volume preservation detailed balance would not hold, and so tempering would not have been a valid modification of the HMC algorithm. Ultimately we found that the presence of a tempering parameter only had the effect of lowering the acceptance rate of the algorithm, without desired increase in tunnelling. Through tracking the evolution of the Hamiltonian and an equilibrium configuration under tempering dynamics for several values of α we found that for our system the only values of α for which tunnelling of lattice variables was induced, were values for which the acceptance rate in the simulation would always be zero; this is likely due to the large number of degrees of freedom in our lattice system.

It is for the above reasons we can conclude that tempering as applied to our lattice system does not improve our simulation therefore should not be perused as a solution to the issues of isolated modes in more complex systems such as lattice field theories, where the fact that there are even more degrees of freedom in the lattice systems will likely lead to the same problem. Instead, as discussed at the end of the results section, a next step for tempering as applied to lattice systems should be an investigation into local tempering. Local tempering may have the ability to tunnel a subset of the lattice variables in proposal configurations without the zero acceptance probability that has shown to arise for normal tempering in our simulations.

References

[1] Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987.

[2] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6), Jun 1953.

[3] Radford Neal. Mcmc using hamiltonian dynamics. *Chapman & Hall/CRC Handbooks of Modern Statistical Methods Handbook of Markov Chain Monte Carlo*, Oct 2011.

[4] Radford M. Neal. Bayesian learning for neural networks. *Lecture Notes in Statistics*, 1996.

[5] Hemant Ishwaran. Applications of hybrid monte carlo to bayesian generalized linear models: Quasicomplete separation and neural networks. *Journal of Computational and Graphical Statistics*, 8(4):779, 1999.

[6] Mikkel N. Schmidt. Function factorization using warped gaussian processes. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML 09*, 2009.

[7] Radford M. Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6(4):353–366, 1996.

[8] Wolfgang Bietenholz. Hadron physics from lattice qcd. *International Journal of Modern Physics E*, 25(07):1642008, 2016.

[9] Luigi Del Debbio, Haralambos Panagopoulos, and Ettore Vicari. Îÿ dependence of su(n) gauge theories. *Journal of High Energy Physics*, 2002(08):044–044, 2002.

[10] Luigi Del Debbio, Gian Mario Manca, and Ettore Vicari. Critical slowing down of topological modes. *Physics Letters B*, 594(3-4):315–323, 2004.

[11] M Creutz and B Freedman. A statistical approach to quantum mechanics. *Annals of Physics*, 132(2):427–462, 1981.

[12] Marise J. E. Westbroek, Peter R. King, Dimitri D. Vvedensky, and Stephan Durr. User's guide to monte carlo methods for evaluating path integrals, Dec 2017.

[13] Ronnnie Rodgers and Laura Raes. Monte carlo simulations of harmonic and anharmonic oscillators in discrete euclidean time, 2014.

[14] Aleksandra Slapik and William Serenone. Lattice monte carlo study of the harmonic oscillator in ..., 2012.

[15] R. Blankenbecler, T. Degrand, and R. L. Sugar. Moment method for eigenvalues and expectation values. *Physical Review D*, 21(4):1055–1061, 1980.

[16] Richard P. Feynman and A. R. Hibbs. *Quantum mechanics and path integrals R.P. Feynman A.R. Hibbs*. McGraw-Hill, 1965.

[17] James Binney and David Benjamin. Skinner. *The physics of quantum mechanics*. Oxford University Press, 2015.

[18] V. Fock. Bemerkung zum virialsatz. *Zeitschrift fur Physik*, 63(11-12):855–858, 1930.

[19] Christof Gattringer and Christian B. Lang. *Quantum Chromodynamics on the Lattice An Introductory Presentation*. Springer Berlin, 2013.

[20] A. Sokal. Monte carlo methods in statistical mechanics: Foundations and new algorithms. *Functional Integration NATO ASI Series*, pages 131–192, 1997.

[21] Herbert Goldstein, Charles P. Poole, and John Safko. *Classical mechanics*. Pearson, 2014.

[22] A.d. Kennedy and Brian Pendleton. Cost of the generalised hybrid monte carlo algorithm for free field theory. *Nuclear Physics B*, 607(3):456–510, 2001.

[23] Alexandros Beskos, Natesh Pillai, Gareth Roberts, Jesus-Maria Sanz-Serna, and Andrew Stuart. Optimal tuning of the hybrid monte carlo algorithm. *Bernoulli*, 19(5A):1501–1534, 2013.

[24] Nawaf Bou-Rabee and Jesús María Sanz-Serna. Randomized hamiltonian monte carlo. *The Annals of Applied Probability*, 27(4):2159–2194, 2017.

[25] Alain Durmus, Eric Moulines, and Eero Saksman. On the convergence of hamiltonian monte carlo, 2017.

[26] Samuel Livingstone, Michael Betancourt, Simon Byrne, and Mark Girolami. On the geometric ergodicity of hamiltonian monte carlo, 2016.

[27] Charles J. Geyer. Markov chain monte carlo maximum likelihood. *Interface Foundation of North America*, 1991.

[28] David J. Earl and Michael W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910, 2005.

[29] E Marinari and G Parisi. Simulated tempering: A new monte carlo scheme. *Europhysics Letters (EPL)*, 19(6):451–458, 1992.

[30] Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

Appendices

Note — Appendices are provided for completeness only and any content included in them will be disregarded for the purposes of assessment.

A Quantum Virial Theorem

The quantum virial theorem relates the expectation value of the kinetic energy to that of the potential. For a single particle of mass m can be derived as follows. We begin with a Hamiltonian of the form:

$$\hat{H}\left(\hat{x},\hat{p}\right)=\hat{T}\left(\hat{p}\right)+\hat{V}\left(\hat{x}\right) \tag{140}$$

, where

$$\hat{T}\left(\hat{p}\right)=\frac{1}{2m}\hat{p}^{2}, \tag{141}$$

$$\hat{p}=-i\frac{d}{dx} \tag{142}$$

(in units where $\hbar=1$) and

$$\hat{V}\left(\hat{x}\right)=V\left(x\right) \tag{143}$$

so the potential is time independent.

Using the identity:

$$\left[\hat{A},\hat{B}\hat{C}\right]=\left[\hat{A},\hat{B}\right]\hat{C}+\hat{B}\left[\hat{A},\hat{C}\right], \tag{144}$$

it can be shown that:

$$\begin{aligned}\left[\hat{H},\hat{x}\hat{p}\right]&=\left[\hat{H},\hat{x}\right]\hat{p}+\hat{x}\left[\hat{H},\hat{p}\right] &(145)\\ &=\frac{1}{2m}\left[\hat{p}^{2},\hat{x}\right]\hat{p}+\hat{x}\left[V\left(x\right),\hat{p}\right] &(146)\\ &=-i\frac{\hat{p}^{2}}{m}+ixV'\left(x\right), &(147)\end{aligned}$$

so that:

$$i\left[\hat{H},\hat{x}\hat{p}\right]=2\hat{T}\left(\hat{p}\right)-xV'\left(x\right). \tag{148}$$

The Heisenberg equation of motion gives for any operator $\hat{A}\left(t\right)$:

$$\frac{d\hat{A}\left(t\right)}{dt}=-i\left[\hat{H},\hat{A}\left(t\right)\right], \tag{149}$$

so that equation 148 becomes:

$$\frac{d\left(\hat{x}\hat{p}\right)}{dt}=2\hat{T}\left(\hat{p}\right)-xV'\left(x\right) \tag{150}$$

Then taking the expectation value of both sides of equation 150 and noting that for stationary states:

$$\langle n| \frac{d\left(\hat{x}\hat{p}\right)}{dt} |n\rangle = 0, \tag{151}$$

we get the virial theorem:

$$2\langle n| \hat{T} |n\rangle = \langle n| \hat{x}\hat{V}'\left(\hat{x}\right) |n\rangle \tag{152}$$

B Derivation of the discrete path integral for quantum harmonic oscillator

Here we follow the derivation given in [11] for the exact result of the path integral for discrete theory and a particle of mass $m = 1$. Since in [11] the derivation has several typographical mistakes and jumps in the algebra that make it difficult to follow for the reader, we have chosen to reproduce it with corrections and alterations.

For a quantum harmonic oscillator of mass $m = 1$, the discrete Euclidean path integral is given in equation 18 as:

$$Z = \int_{-\infty}^{+\infty} \prod_{i=0}^{N-1} dx_i \exp\left(-\sum_{j=0}^{N-1} a\left[\frac{1}{2}\left(\frac{x_{j+1}-x_j}{a}\right)^2 + \frac{1}{2}\mu^2 x_j^2\right]\right). \tag{153}$$

We begin by defining an operator T such that its matrix elements in the Schrödinger picture are given by:

$$\langle x'| \hat{T} |x\rangle = \exp\left(-\frac{1}{2a}\left(x'-x\right)^2 - \frac{\mu^2 a}{4}\left(x^2+x'^2\right)\right). \tag{154}$$

Then we can use the completeness relation that:

$$\hat{1} = \int_{-\infty}^{\infty} |x\rangle \langle x| . \tag{155}$$

By inserting $N-1$ copies of this relation into the expression (one between each pair of $\hat{T}$s):

$$\mathrm{Tr}\left(\hat{T}^N\right), \tag{156}$$

and using the definition of the matrix elements in equation 154, we recover the path integral in equation 153, that is:

$$Z = \mathrm{Tr}\left(\hat{T}^N\right). \tag{157}$$

Here we take the trace over the physical Hilbert space, so that for any operator $\hat{A}$ in the Schrödinger eigenbasis, the trace is:

$$\mathrm{Tr}\left(\hat{A}\right) = \int_{-\infty}^{\infty} dx \langle x| \hat{A} |x\rangle , \tag{158}$$

which is of course basis independent.

The next step is to make the ansatz that:

$$\hat{T} = \int_{-\infty}^{\infty} d\omega e^{\frac{-\mu^2 a}{4}\hat{x}^2} e^{-i\hat{p}\omega} e^{-1\frac{1}{2a}\omega^2} e^{\frac{-\mu^2 a}{4}\hat{x^2}}, \tag{159}$$

then, using that canonical momentum generates translations on position, that is:

$$e^{-i\hat{p}\Delta} |x\rangle = |x + \Delta\rangle \tag{160}$$

which can easily be shown by Fourier transforming the position eigenbasis into momentum space, acting with the momentum operator in the exponential then Fourier transforming back into position space; we can calculate the matrix element $\langle x'| T |x\rangle$ and recover equation 154. We observe that the integral over ω in equation 159 is Gaussian; we may use the standard result:

$$\int_{-\infty}^{+\infty} dx e^{-\alpha x^2 + i\beta x} = \sqrt{\frac{\pi}{\alpha}} e^{-\frac{\beta^2}{4a}}, \tag{161}$$

this result follows from completing the square on x, analytically continuing the integral over x into the complex plane and creating a rectangular contour in the lower half plane, then applying residue theory, it is valid provided $\alpha \in \mathbb{R}_{>0}$ and $\beta \in \mathbb{R}$. Applying the identity in equation 161 to equation 159 gives:

$$\hat{T} = \sqrt{2\pi a} e^{-\frac{-\mu^2 a}{4}\hat{x}^2} e^{-\frac{a}{2}\hat{p}^2} e^{-\frac{\mu^2 a}{4}\hat{x}^2}. \tag{162}$$

Notice at this stage in our derivation we could apply the Baker-Campbell-Hausdorff formula to combine the exponentials; dropping $\mathcal{O}(a^2)$ would then give the harmonic oscillator Hamiltonian exponentiated. Since this system is exactly solvable in the discrete case, we will avoid doing this and keep all terms.

The canonical commutation relation of quantum mechanics gives:

$$[\hat{x}, \hat{p}] = i\hbar, \tag{163}$$

which can be along with the identities that for operators $\hat{A}$ and $\hat{B}$:

$$\left[\hat{A}, \hat{B}^n\right] = n\hat{B}^{n-1}\left[\hat{A}, \hat{B}\right] \tag{164}$$

and

$$\left[\hat{A}^n, \hat{B}\right] = n\hat{A}^{n-1}\left[\hat{A}, \hat{B}\right], \tag{165}$$

if $\left[\hat{A}, \left[\hat{A}, \hat{B}\right]\right] = \left[\hat{B}, \left[\hat{A}, \hat{B}\right]\right] = 0$, to show that:

$$\left[\hat{x}, e^{-\frac{1}{2}a\hat{p}^2}\right] = -ia\hat{p} \tag{166}$$

and

$$\left[\hat{p}, e^{-\frac{\mu^2 a}{4}\hat{x}^2}\right] = i\frac{\mu^2 a}{2}\hat{x} \tag{167}$$

Using identities 166 and 167 we can easily show that:

$$\hat{x}\hat{T} = \hat{T}\left[\left(1 + \frac{a^2\mu^2}{2}\right)\hat{x} - ia\hat{p}\right], \tag{168}$$

and

$$\hat{p}\hat{T} = \hat{T}\left[\left(1 + \frac{a^2\mu^2}{2}\right)\hat{p} + ia\mu^2\left(1 + \frac{a^2\mu^2}{4}\right)\hat{x}\right]. \tag{169}$$

Iterating equations 168 and 169 a second time gives:

$$\left[\hat{p}^2 + \mu^2\left(1 + \frac{a^2\mu^2}{4}\right)\hat{x^2}, \hat{T}\right] = 0. \tag{170}$$

Defining a new angular frequency parameter ω by:

$$\omega^2 = \mu^2\left(1 + \frac{a^2\mu^2}{4}\right), \tag{171}$$

then from equation 170 we have that $\hat{T}$ commutes with the simple harmonic oscillator Hamiltonian:

$$\hat{H} = \frac{1}{2}\hat{p}^2 + \frac{1}{2}\omega^2\hat{x}^2. \tag{172}$$

Since $\hat{H}$ and $\hat{T}$ commute we know $\hat{T}$ is diagonalized by the eigenstates of $\hat{H}$.

The Hamiltonian is in the form of a harmonic oscillator with angular frequency ω, therefore we may define the corresponding ladder operators:

$$\hat{a}^\dagger = \frac{1}{\sqrt{\omega}}\left(\hat{p} + i\omega\hat{x}\right) \tag{173}$$

and

$$\hat{a} = \frac{1}{\sqrt{\omega}}\left(\hat{p} - i\omega\hat{x}\right) \tag{174}$$

which allows us to write:

$$\hat{H} = \left(\hat{a}^\dagger\hat{a} + \frac{1}{2}\right)\omega \tag{175}$$

The eigenstates of $\hat{H}$ satisfy the standard relations:

$$\hat{a}\left|0\right\rangle = 0, \tag{176}$$

$$\left(\hat{a}^\dagger\right)^n\left|0\right\rangle = \left|n\right\rangle, \tag{177}$$

and

$$\langle n|n\rangle = n!. \tag{178}$$

Using the identities of equations 168 and 169 we can show that:

$$\hat{a}\hat{T} = \hat{T}\hat{a}\left(1 + \frac{a^2\mu^2}{2} - a\mu\left(1 + \frac{a^2\mu^2}{4}\right)^{\frac{1}{2}}\right). \tag{179}$$

Since the eigenstates of $\hat{H}$ given as $|n\rangle$ diagonalise $\hat{T}$, for λ_i the eigenvalues of $\hat{T}$:

$$\hat{T}\left|n\right\rangle = \lambda_n\left|n\right\rangle. \tag{180}$$

We can then use that $\hat{a}\,|n\rangle = \sqrt{n}\,|n-1\rangle$ and equation 179 to show the ratio:

$$\frac{\lambda_n}{\lambda_{n-1}} = 1 + \frac{a^2\mu^2}{2} - a\mu\left(1 + \frac{a^2\mu^2}{4}\right)^{\frac{1}{2}} \tag{181}$$

$$= \left(\left(1 + \frac{a^2\mu^2}{4}\right)^{\frac{1}{2}} - \frac{1}{2}a\mu\right)^2 \tag{182}$$

$$\coloneqq R, \tag{183}$$

so that

$$\lambda_n = R\lambda_{n-1} = R^2\lambda_{n-2} = \cdots = R^n\lambda_0, \tag{184}$$

which gives:

$$\hat{T}\,|n\rangle = \lambda_n\,|n\rangle \tag{185}$$

$$= R^n\lambda_0\,|n\rangle \tag{186}$$

$$= \lambda_0 e^{n\log R}\,|n\rangle\,. \tag{187}$$

However, the energy eigenstates $|n\rangle$ are defined through:

$$\hat{H}\,|n\rangle = \left(n + \frac{1}{2}\right)\omega\,|n\rangle\,, \tag{188}$$

we therefore make the ansatz that for constants C and K:

$$\hat{T} = Ce^{K\hat{H}}, \tag{189}$$

which can be shown to be consistent with equation 187 provided we take $K\omega = \log R$ and $Ce^{\frac{K\omega}{2}} = \lambda_0$ and can conclude that:

$$\hat{T} = Ce^{\frac{\hat{H}}{\omega}\log R} \tag{190}$$

$$= CR^{\frac{1}{\omega}\hat{H}}. \tag{191}$$

We may then calculate C by first taking the trace of $\hat{T}$ over the energy eigenbasis $|n\rangle$:

$$\mathrm{Tr}\left(\hat{T}\right) = \sum_n \langle n|\,\hat{T}\,|n\rangle \tag{192}$$

$$= C\sum_n R^{n+\frac{1}{2}} \tag{193}$$

$$= C\frac{R^{\frac{1}{2}}}{1-R} \tag{194}$$

$$= C\frac{1}{a\mu} \tag{195}$$

where we have used that $\hat{H}\,|n\rangle = E_n\,|n\rangle = \left(n + \frac{1}{2}\right)\omega$ for the harmonic oscillator in units where $\hbar = 1$, and in order to compute the sum we have used that $|R| < 1$. We then take the

trace of $\hat{T}$ over the position eigenbasis according to equation 158, and using equation 154:

$$\mathrm{Tr}\left(\hat{T}\right) = \int dx \, \langle x| \, \hat{T} \, |n\rangle \tag{196}$$

$$= \int dx e^{-\frac{a\mu^2}{2}x^2} \tag{197}$$

$$= \frac{1}{\mu}\sqrt{\frac{2\pi}{a}}, \tag{198}$$

comparing equation 198 and 195 we conclude $C = \sqrt{2\pi a}$, and so

$$T = \sqrt{2\pi a} R^{\frac{\hat{H}}{\omega}}. \tag{199}$$

We may now explicitly compute the trace expression for the discrete path integral given in equation 156 using the result for $\hat{T}$ in equation 199. We again take the trace over the energy eigenstates and use the fact that $|R| < 1$ to compute the resulting sum, this gives:

$$Z = (2\pi a R)^{\frac{N}{2}} \frac{1}{1 - R^N}. \tag{200}$$

This is the exact expression for the discrete path integral of a quantum harmonic oscillator of mass $m = 1$, angular frequency μ on an imaginary time lattice of $N - 1$ sites with lattice spacing a.

The correlation functions are given by:

$$\langle x_i x_{i+j} \rangle = \frac{1}{Z} Tr\left(\hat{x}\hat{T}^j \hat{x}\hat{T}^{N-j}\right) \tag{201}$$

$$= \frac{1}{2\mu\left(1 + \frac{a^2\mu^2}{4}\right)^{\frac{1}{2}}\left(1 - R^n\right)} \left(R^j + R^{N-j}\right). \tag{202}$$

Taking $j = 0$, equation 201 becomes:

$$\langle x^2 \rangle = \frac{1}{2\omega}\left(\frac{1 + R^N}{1 - R^N}\right) \tag{203}$$

$$= \frac{1}{2\mu\left(1 + \frac{a^2\mu^2}{4}\right)^{\frac{1}{2}}} \left(\frac{1 + R^N}{1 - R^N}\right) \tag{204}$$

We have shown in above that the discrete theory for the harmonic oscillator leads to a quantum system with the Hamiltonian given in equation 172. This is the Hamiltonian of a quantum harmonic oscillator of angular frequency ω rather than μ and hence we employ the standard result from quantum theory that for a a quantum harmonic oscillator of angular frequency ω the ground state wave function is given by:

$$\psi(x) = \left(\frac{\omega}{\pi}\right)^{\frac{1}{4}} \exp\left(-\frac{1}{2}\omega x^2\right) \tag{205}$$

$$= \left(\frac{\mu\left(1 + \frac{a^2\mu^2}{4}\right)^{\frac{1}{2}}}{\pi}\right)^{\frac{1}{4}} \exp\left(-\frac{1}{2}\mu\left(1 + \frac{a^2\mu^2}{4}\right)^{\frac{1}{2}} x^2\right) \tag{206}$$

In order order to get the discrete path integral results for a particle of any mass (not just unity), we note that the action for a particle of mass m:

$$S_M(\boldsymbol{x}) = \sum_{i=0}^{N-1} a\left[\frac{1}{2}m\left(\frac{x_{i+1}-x_i}{a}\right)^2 + \frac{1}{2}\mu^2 x_i^2\right] \tag{207}$$

can be obtained from of the discrete Euclidean action of a particle of mass unity:

$$S_M(\boldsymbol{x}) = \sum_{i=0}^{N-1} a\left[\frac{1}{2}\left(\frac{x_{i+1}-x_i}{a}\right)^2 + \frac{1}{2}\eta^2 x_i^2\right]. \tag{208}$$

by making the replacements

$$x \to x\sqrt{m} \tag{209}$$

and

$$\mu \to \frac{\mu}{\sqrt{m}}. \tag{210}$$

Making these replacements in the above results gives:

$$\langle x_i x_{i+j}\rangle = \frac{1}{2\omega m}\left(\frac{R^j + R^{N-j}}{1-R^N}\right) \tag{211}$$

$$\langle x^2\rangle = \frac{1}{2m\omega}\left(\frac{1+R^N}{1-R^N}\right) \tag{212}$$

$$\psi(x) = \left(\frac{\omega}{\pi}\right)^{\frac{1}{4}} \exp\left(-\frac{1}{2}\omega x^2\right) \tag{213}$$

but where now:

$$R = 1 + \frac{a^2\mu^2}{2m} - \frac{a\mu}{\sqrt{m}}\left(1+\frac{a^2\mu^2}{4m}\right)^{\frac{1}{2}} \tag{214}$$

$$\omega = \frac{\mu}{\sqrt{m}}\sqrt{1+\frac{a^2\mu^2}{4m}}. \tag{215}$$

C Simulation Code

In order to save space only the main method has been included from the simulation written in C++, this gives an idea of the structure of the program and how it works. For the entire simulation, along with various utilities such as Makefiles please see `https://github.com/FranklandJack/HMC-Oscillator` where the interested reader may pull the repository and run the simulation themselves as well as view the rest of the project files.

```
1 #include <iostream>
2 #include <fstream>
3 #include <cmath>
4 #include <random>
```

```
 5 #include <chrono>
 6 #include <string>
 7 #include <vector>
 8 #include <algorithm>
 9 #include <boost/program_options.hpp>
10 #include "Ipotential.hpp"
11 #include "HarmonicPotential.hpp"
12 #include "AnharmonicPotential.hpp"
13 #include "LatticeFunctions.hpp"
14 #include "Histogram.hpp"
15 #include "ProgressBar.hpp"
16 #include "Timer.hpp"
17 #include "makeDirectory.hpp"
18 #include "getTimeStamp.hpp"
19 #include "HMCLattice1D.hpp"
20 #include "DataArray.hpp"
21 #include "HMCInput.hpp"
22 #include "HMCOutput.hpp"
23 #include "metropolisUpdate.hpp"
24
25
26
27 int main(int argc, const char * argv[])
28 {
29     // Start Timing.
30     Timer timer;
31
32 /*
    ********************************************************************************
33 ********************************************** Input
    Parameters **********************************************
34 ********************************************************************************
    */
35
36     // Lattice Parameters.
37     int    latticeSize;
38     double latticeSpacing;
39
40     // Oscillator Parameters.
41     double mass;
42     double muSquared;
43     double lambda;
44     double fSquared;
45
```

```
46      // HMC Parameters.
47      int     lfStepCount;
48      double  lfStepSize;
49
50      // Other Parameters.
51      int configCount;
52      int burnPeriod;
53      int mInterval;
54
55      // Choice of potential.
56      HMCInput::PotentialType potentialChoice;
57
58      // Histogram parameters.
59      int     numBins;
60      double  histMaxValue;
61      double  histMinValue;
62
63      // Tempering parameter sqrt(alpha)
64      double temperingParameter;
65
66      // Correlation range calculator.
67      int correlationRange;
68
69      // Set up optional command line argument.
70      boost::program_options::options_description desc("Options for
           hmc oscillator program");
71
72      // Add all optional command line arguments.
73      desc.add_options()
74
75          ("lattice-size,L", boost::program_options::value<int>(&
               latticeSize)->default_value(100), "The number of
               lattice sites")
76          ("lattice-spacing,a", boost::program_options::value<double
               >(&latticeSpacing)->default_value(1.0), "The spacing
               between lattice sites")
77          ("mass,m", boost::program_options::value<double>(&mass)->
               default_value(1.0), "The mass of the oscillator")
78          ("mu-squared,u", boost::program_options::value<double>(&
               muSquared)->default_value(1), "The mu^2 of the
               oscillator")
79          ("lambda,l", boost::program_options::value<double>(&lambda
               )->default_value(0.0), "The lambda value of the
               oscillator")
80          ("f-squared,f", boost::program_options::value<double>(&
```

```
            fSquared)->default_value(0.0), "The f^2 value of the
            oscillator")
81          ("lf-step-count,N", boost::program_options::value<int>(&
            lfStepCount)->default_value(5), "The number of leapfrog
             steps")
82          ("lf-step-size,d", boost::program_options::value<double>(&
            lfStepSize)->default_value(0.2), "The leapfrog step
            size")
83          ("configuration-count,c", boost::program_options::value<
            int>(&configCount)->default_value(100000), "The number
            of configurations")
84          ("burn-period,b", boost::program_options::value<int>(&
            burnPeriod)->default_value(10000), "The burn period")
85          ("measurement-interval,i", boost::program_options::value<
            int>(&mInterval)->default_value(4), "The number of
            steps between measurements")
86          ("number-bins,B", boost::program_options::value<int>(&
            numBins)->default_value(100), "The number of bins in
            the position histogram")
87          ("histogram-max-value,R", boost::program_options::value<
            double>(&histMaxValue)->default_value(4.0), "Maximum
            value in histogram range")
88          ("histogram-min-value,r", boost::program_options::value<
            double>(&histMinValue)->default_value(-4.0), "Minimum
            value in histogram range")
89          ("tempering-parameter,T", boost::program_options::value<
            double>(&temperingParameter)->default_value(1.0), "sqrt
            (alpha) that is the tempering parameter")
90          ("correlation-range", boost::program_options::value<int>(&
            correlationRange)->default_value(10), "range to
            calculate the correlation function for.")
91          ("anharmonic", "use alternative potential")
92          ("help,h", "produce help message");
93
94      // Make arguments available to program
95      boost::program_options::variables_map vm;
96      boost::program_options::store(boost::program_options::
           parse_command_line(argc,argv,desc), vm);
97      boost::program_options::notify(vm);
98
99      // If the user asks for help display it then exit.
100     if(vm.count("help"))
101     {
102         std::cout << desc << "\n";
103         return 1;
```

```
104     }
105
106     // If the user specifies alternate potential need to let the
        program know.
107     if(vm.count("anharmonic"))
108     {
109         potentialChoice = HMCInput::Potential_Anharmonic;
110     }
111
112
113     else
114     {
115         potentialChoice = HMCInput::Potential_Harmonic;
116     }
117
118     // Construct an input object and print the values to the
        command line.
119     HMCInput inputParameters
120     {
121         latticeSize,
122         latticeSpacing,
123         mass,
124         muSquared,
125         lambda,
126         fSquared,
127         lfStepCount,
128         lfStepSize,
129         configCount,
130         burnPeriod,
131         mInterval,
132         potentialChoice,
133         numBins,
134         histMaxValue,
135         histMinValue,
136         temperingParameter,
137         correlationRange
138     };
139
140     std::cout << inputParameters << '\n';
141     int outputColumnWidth = 10;
142
143 /*
    ************************************************************************

144 ************************************************ Output Set
```

```
    Up ***************************************************
145 ***************************************************************************
    */
146
147
148     // Create string which holds unique time/date stamp.
149     std::string outputName(makeDirectory(getTimeStamp()));
150
151     // Create output file to hold the input parameters.
152     std::ofstream inputParametersOutput(outputName + "/input.txt")
            ;
153
154     // Create output file to hold numerical values calculated
            during the simulation.
155     std::ofstream resultsOutput(outputName + "/results.txt");
156
157     // Create output file to hold the wave function.
158     std::ofstream wavefunctionOutput(outputName+"/wavefunction.dat
            ");
159
160     // Create output file to hold the correlation function data.
161     std::ofstream correlationOutput(outputName+"/correlation.dat")
            ;
162
163     // Create output file to hold the final configuration so it
            can be resused in future simulations.
164     std::ofstream finalConfigOutput(outputName+"/
            finalConfiguration.dat");
165
166     // Create output file for the mean position on each
            configuration.
167     std::ofstream positionOutput(outputName+"/position.dat");
168
169     // Create output file for the mean position squared on each
            configuration.
170     std::ofstream positionSquaredOutput(outputName+"/
            positionSquared.dat");
171
172     // Create output file for the mean energy gs on each
            configuration.
173     std::ofstream gsEnergyOutput(outputName+"/gsEnergy.dat");
174
175     // Create output file for the autocorrealtion in position.
176     std::ofstream positionAutoCorrelationOutput(outputName+"/
            autocorrelationPosition.dat");
```

```
177
178     // Create output file for the autocorrelation in position
            squared.
179     std::ofstream positionSquaredAutoCorrelationOutput(outputName+
            "/autocorrelationPositionSquared.dat");
180
181     // Create output file for the autocorrelation in position
            fourth.
182     std::ofstream positionFourthAutoCorrelationOutput(outputName+"
            /autocorrelationPositionFourth.dat");
183
184     // Create output file for the Hamiltonian on the first
            leapfrog update at equilibrium so we can see its evolution.
185     std::ofstream hamtilonianEvolutionOutput(outputName+"/
            hamiltonianEvolution.dat");
186
187     // Create output file for the configuration on the leap frog
            update so we can see what happens to it.
188     std::ofstream configurationEvolutionOutput(outputName+"/
            configurationEvolution.dat");
189
190
191
192     // Create potentials for each type.
193     HarmonicPotential       harmonicPotential(muSquared, lambda);
194     AnharmonicPotential     anharmonicPotential(lambda, fSquared);
195     Ipotential              *potential = nullptr;
196
197     switch(potentialChoice)
198     {
199         case HMCInput::Potential_Harmonic:
200             potential = &harmonicPotential;
201             break;
202
203         case HMCInput::Potential_Anharmonic:
204             potential = &anharmonicPotential;
205             break;
206
207         default:
208             std::cout << "No potential selected, exiting program";
209             return 1;
210     }
211
212 /*
    ************************************************************************
```

```
213 ********************************************** Set up
    Measurements **********************************************
214 ********************************************************************************
    */
215
216     int     mCount                      = configCount/mInterval;
217
218     int     acceptance                  = 0;
219
220     DataArray positionData;
221     positionData.reserve(mCount);
222
223     DataArray positionSquaredData;
224     positionSquaredData.reserve(mCount);
225
226     DataArray positionFourthData;
227     positionFourthData.reserve(mCount);
228
229     DataArray actionData;
230     actionData.reserve(mCount);
231
232     DataArray keData;
233     keData.reserve(mCount);
234
235     DataArray dhData;
236     dhData.reserve(mCount);
237
238     DataArray expdhData;
239     expdhData.reserve(mCount);
240
241     DataArray gsEnergyData;
242     gsEnergyData.reserve(mCount);
243
244
245
246
247     std::vector<double> correlation(correlationRange,0);
248     std::vector<double> correlationSquared(correlationRange,0);
249
250
251     // Set up arrays to hold the wavefunction calculated on each
        measured configuration.
252     std::vector<double> wavefunction(numBins,0.0);
253     std::vector<double> wavefunctionSquared(numBins,0.0);
```

```
254     std::vector<double> wavefunctionError(numBins,0.0);
255
256
257
258
259 /*
    ************************************************************************
    ********************************************** Prepare PRN
    generation ********************************************
    ************************************************************************
    */
262
263     unsigned int seed = static_cast<unsigned int>(std::chrono::
        system_clock::now().time_since_epoch().count());
264     // Test seed = 1.
265     std::default_random_engine generator(1);
266
267
268
269 /*
    ************************************************************************
    ********************************************** Prepare
    Lattice ***********************************************************
    ************************************************************************
    */
272
273     // Create a lattice that can be updated.
274     HMCLattice1D lattice(latticeSize, latticeSpacing, mass,
        potential);
275
276     // Initialise the lattice with uniformly distributed position
        coordinates between -1 and +1.
277     lattice.initialise(generator);
278
279     // TEST TO SEE WHAT HAPPENS IF WE INITIALISE LATTICE IN ONE
        MINIMA.
280     /*
281     if(vm.count("anharmonic"))
282     {
283         for(int i = 0; i < lattice.getSize(); ++i)
284         {
285             lattice[i] = -sqrt(fSquared);
286         }
```

```
287     }
288     */
289
290     // Create a lattice to represent the current state of the
            system
291     HMCLattice1D currentLattice = lattice;
292
293 /*
        ************************************************************************
294 ************************************************ Do HMC
        ************************************************************
295 ************************************************************************
        */
296
297     // Progress bar to inform use how far through simulation they
            are.
298
299     ProgressBar progressBar(configCount+burnPeriod,0.5, 2, 0.0);
300
301     // Tempering will only begin once we are in equilibrium, so we
            need a second tempering parameter which will change
302     // once we reach equilibrium.
303     double tempering = 1;
304     for(int config = 0; config < configCount+burnPeriod; ++config)
305     {
306         std::cout << progressBar;
307         progressBar.increment();
308
309         // If we are in equilibrium update the tempering parameter
                .
310         if(config>=burnPeriod)
311         {
312             tempering = temperingParameter;
313         }
314
315         // Randomise the Momenta.
316         lattice.randomiseMomenta(generator);
317
318         // Calculate the Hamiltonian.
319         double hamiltonianBefore = lattice.hamiltonian();
320
321         // Save original lattice.
322         currentLattice = lattice;
323
```

```
324         if(config == burnPeriod)
325         {
326             // If we are on the first equilibrium update then
                print the Hamiltonian along the trajectory.
327             lattice.leapFrog(lfStepCount, lfStepSize, tempering,
                hamtilonianEvolutionOutput,
                configurationEvolutionOutput);
328         }
329         else
330         {
331             // Do a leapfrog update on the lattice.
332             lattice.leapFrog(lfStepCount, lfStepSize, tempering);
333         }
334
335         // Calculate Hamiltonian after.
336         double hamiltonianAfter = lattice.hamiltonian();
337
338         // Do a Metropolis update and if it fails make sure we
            restore the original lattice.
339         if(!(metropolisUpdate(hamiltonianBefore, hamiltonianAfter,
            generator)))
340         {
341             lattice = currentLattice;
342
343         }
344         // Otherwise if we are out of the burn period record the
            acceptance.
345         else if(config >= burnPeriod)
346         {
347             ++acceptance;
348         }
349         // Measurements are made at the interval specified by the
            user once we have exceeded the burn period.
350         if(0 == config%mInterval && config >= burnPeriod)
351         {
352             // Numerical values.
353             positionData.push_back(lattice.meanX());
354             positionSquaredData.push_back(lattice.meanXSquared());
355             positionFourthData.push_back(lattice.meanXFourth());
356             actionData.push_back(lattice.action()/latticeSize);
357             keData.push_back(lattice.kineticEnergy()/latticeSize);
358             dhData.push_back(hamiltonianAfter - hamiltonianBefore)
                ;
359             expdhData.push_back(exp(hamiltonianBefore -
                hamiltonianAfter));
```

```
360
361             // Wave Function.
362
363             // Create histogram to record the wavefunction for
                   this configuration.
364             Histogram positionHistogram(histMinValue, histMaxValue
                   , numBins);
365
366             for(int index = 0; index < latticeSize; ++index)
367             {
368                 positionHistogram(lattice[index]);
369             }
370
371             // Normalise the wavefunction and store it in the
                   vector of wavefunctions.
372             positionHistogram.normalise();
373
374             for(int i = 0; i < wavefunction.size(); ++i)
375             {
376                 double probabilty = positionHistogram.count(i);
377                 wavefunction[i] += probabilty/mCount;
378                 wavefunctionSquared[i] += probabilty * probabilty/
                       mCount;
379             }
380
381             // Correlation function.
382
383             // Store first n values of the correlation function on
                    the lattice.
384             for(int i = 0; i < correlation.size(); ++i)
385             {
386                 double correlationValue = lattice.correlation(i);
387                 correlation[i]         += correlationValue/mCount;
388                 correlationSquared[i]  += correlationValue*
                       correlationValue/mCount;
389
390             }
391
392             // Ground state energy.
393             gsEnergyData.push_back((*potential).groundStateEnergy(
                   lattice.meanXSquared(),lattice.meanXFourth()));
394
395         }
396
397     }
```

```
398         progressBar.increment();
399         std::cout << progressBar;
400         std::cout << "\nDone!...\n" << std::endl;
401 
402 
403 
404 /*
        ***************************************************************************
405 ***************************************** Calculate Observables
        ***********************************************************
406 ***************************************************************************
        */
407 
408         double acceptanceRate = static_cast<double>(acceptance)/(
              configCount);
409 
410         // Calculate variance and standard error using the normal
              formulas.
411         double varianceAcceptance  = (acceptanceRate - acceptanceRate
              * acceptanceRate) * mCount/(mCount-1);
412         double sdAcceptance        = sqrt(varianceAcceptance)/sqrt(
              mCount);
413 
414         for(int i = 0; i < wavefunction.size(); ++i)
415         {
416             double varianceWavefunction =   (wavefunctionSquared[i] -
                  wavefunction[i] * wavefunction[i]) * mCount/(mCount-1);
417             wavefunctionError[i]        = sqrt(varianceWavefunction)/
                  sqrt(mCount);
418         }
419 
420         std::vector<double> correlationError(correlation.size(),0);
421         for(int i = 1; i < correlationError.size();++i)
422         {
423             double varianceCorrelation    = (correlationSquared[i] -
                  correlation[i]*correlation[i]) * mCount/(mCount-1);
424             correlationError[i]           = sqrt(varianceCorrelation)/
                  sqrt(mCount);
425         }
426 
427 
428         double position             = positionData.mean();
429         double positionError        = positionData.error();
430         double positionIAC          = positionData.
```

```
        integratedAutocorrelationTime(10);
431
432     double positionSquared      = positionSquaredData.mean();
433     double positionSquaredError = positionSquaredData.error();
434     double positionSquaredIAC   = positionSquaredData.
        integratedAutocorrelationTime(10);
435
436     double positionFourth       = positionFourthData.mean();
437     double positionFourthError  = positionFourthData.error();
438     double positionFourthIAC    = positionFourthData.
        integratedAutocorrelationTime(10);
439
440     double kineticEnergy        = keData.mean();
441     double kineticEnergyError   = keData.error();
442
443     double action               = actionData.mean();
444     double actionError          = actionData.error();
445
446     double dh                   = dhData.mean();
447     double dhError              = dhData.error();
448
449     double expdh                = expdhData.mean();
450     double expdhError           = expdhData.error();
451
452     double gsEnergy             = gsEnergyData.mean();
453     double gsEnergyError        = gsEnergyData.error();
454
455
456 /*
    ************************************************************************

457 ************************************ Output to command line
    ************************************************************
458 ************************************************************************
    */
459     // Construct an object to hold the results.
460     HMCOutput results
461     {
462         acceptanceRate*100,
463         sdAcceptance*100,
464         action,
465         actionError,
466         kineticEnergy,
467         kineticEnergyError,
468         dh,
```

```
469             dhError,
470             expdh,
471             expdhError,
472             position,
473             positionError,
474             positionIAC,
475             positionSquared,
476             positionSquaredError,
477             positionSquaredIAC,
478             positionFourth,
479             positionFourthError,
480             positionFourthIAC,
481             gsEnergy,
482             gsEnergyError
483         };
484
485         // Output the results to the command line
486
487         std::cout << results << '\n';
488
489 /*
    ************************************************************************
490 ********************************************** File Output
    ************************************************************
491 ************************************************************************
    */
492
493
494         double histSpacing = (histMaxValue - histMinValue) / numBins;
495         for(int i = 0;i < wavefunction.size();++i)
496         {
497             wavefunctionOutput << (histMinValue + histSpacing/2) + i *
                    histSpacing << ' ' << wavefunction[i] << ' ' <<
                wavefunctionError[i] << '\n';
498         }
499
500         for(int i = 0; i < correlation.size(); ++i)
501         {
502             correlationOutput << i << " " << correlation[i] << ' ' <<
                correlationError[i] << '\n';
503         }
504
505         // Calcualte the autocorrelation in the measurements.
506         std::vector<double> positionAutoCorrelation = positionData.
```

```
        autoCorrelation(0, 100);
507     std::vector<double> positionSquaredAutoCorrelation =
        positionSquaredData.autoCorrelation(0,100);
508     std::vector<double> positionFourthAutoCorrelation =
        positionFourthData.autoCorrelation(0,100);
509
510     for(int i = 0; i < positionAutoCorrelation.size(); ++i)
511     {
512         positionAutoCorrelationOutput << i << ' ' <<
            positionAutoCorrelation[i] << '\n';
513         positionSquaredAutoCorrelationOutput << i << ' ' <<
            positionSquaredAutoCorrelation[i] << '\n';
514         positionFourthAutoCorrelationOutput << i << ' ' <<
            positionFourthAutoCorrelation[i] << '\n';
515
516
517     }
518
519     // Output the input parameters to their file.
520     inputParametersOutput << inputParameters;
521
522     // Output the numerical results to the file.
523     resultsOutput << results;
524
525     // Output final configuration.
526     finalConfigOutput << lattice;
527
528     // Output position data.
529     positionOutput << positionData;
530
531     // Output position squared data.
532     positionSquaredOutput << positionSquaredData;
533
534     // Output gs energy data.
535     gsEnergyOutput << gsEnergyData;
536
537
538 /*
    ***************************************************************************

539 ********************************************** End Program
    ************************************************
540 ***************************************************************************
    */
541     std::cout << "Simulation Complete! Results have been outputed
```

```
            to the directory " << outputName << '\n';
542         std::cout << "Time take to execute (s):    " << timer.elapsed()
                << std::endl << std::endl;
543
544
545
546         return 0;
547     }
```