# Monte Carlo Applications and Lattice QCD

**Jacob Scott**

Advisor: Dr. Tom Blum

Spring 2016

**Abstract**

The Monte Carlo method is a broad class of random sampling techniques. One facet of its power arises in its ability to compute complex multidimensional integrals simply through large sample sizes. In this paper, we explore the use of Monte Carlo techniques and their advantage in modeling physical statistical systems such as the Ising model, Feynman path integrals and lattice QCD. The Metropolis algorithm, one type of Monte Carlo method, is applied to evolve these systems while other methods are used to compute and measure different observables. What we will see is that these measurements are an accurate representation of the real-world counterpart whose error arises from simplifying assumptions made on the models, approximations of the measurements and the finite simple sizes. These above sources of error can be rectified by more accurate models and larger sample size, the former of which we will see in modeling of lattice QCD.

# Contents

## 1 Introduction

The myriad methods developed in the sciences and mathematics, no matter how esoteric, come about due to a problem that is unsolved that scientists or mathematicians want to solve. Problems arise due to new information, new measurements or even just a new idea that one person may have and these problems are solved using one or a number of these methods or new methods are discovered.

The field of lattice quantum chromodynamics (QCD) is a good example of this. Lattice QCD is the study of the strong force through the use of discretized spacetime lattice simulations and is considered a nonperturbative theory. Because of this, more traditional methods can not be used. In electrodynamics, the coupling constant (the constant that determines the strength of the force) is the fine-structure constant:

$$\alpha = \frac{1}{4\pi\epsilon_0}\frac{e^2}{\hbar c} \approx \frac{1}{137}. \tag{1}$$

Because of its extremely small value, perturbative methods can be implemented and approximations can be made that are in the form of series of powers of $\alpha$. Such series would be dominated by the first couple terms and even first-order solutions can be very accurate. Whereas in QCD, the coupling constant $g$ is of the order of one and therefore perturbation theory is not a valid method for low energy systems but lattice QCD is. Lattice QCD is not perturbative but rather finds its power in the form of computers.

Lattice QCD is a discrete and numerical approximation of the continuum and as such, when the spacing between lattice sites $a \to 0$, we return to the continuous theory. Because of this, the error that arises comes from this discretization and also from the numerical methods used in this paper. A common compromise is the choice for lattice size. Too large and the program can take too long to run to be practical. Too small and the error on the data is large.

The numerical methods that are used in this paper are Monte Carlo methods. These methods rely on random sampling in the hopes that doing so will eventually approximate the equilibrium state of a system. The method used specifically in the evolution of the systems is the Metropolis algorithm, an algorithm that decides whether a change in the system should be accepted or rejected. This decision is based on the physics of the system. For example, if a system tends towards the lowest energy state, then the Metropolis algorithm is used to accept states that are a lower energy. Monte Carlo methods are also used to approximate multi-dimensional integrals to measure specific observables of a system. As such, this paper highlights the flexibility and power of the aforementioned methods and shows how measurements are relatively simple once a system is properly modeled.

In this paper, I will explore simpler systems using the methods I will then use for lattice QCD. These systems are the Ising model and the harmonic oscillator. In every simulation, the same methods are used with the only changes coming in the form of the physics of the specific systems. Even though these systems are vastly different, the methods are the same. Known methods, in this case Monte Carlo methods, are used to explore a newer problem, nonperturbative QCD.

## 2 Monte Carlo and the Metropolis Algorithm

The Monte Carlo method is a broad class of computational algorithms where the use of random sampling yields numerical results. Any problem that has a probabilistic interpretation can be solved with the Monte Carlo method. And, due to the law of large numbers, with a sufficiently large sample size, expectation values (or, analogously, any integral) can be approximated by a simple mean. For example, we can approximate the following integral by [2]:

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{N}\sum_{i=1}^{N} f(x_i) \tag{2}$$

for $N$ samples. This integral can be generalized to any finite dimensional integral over some volume $\Omega$ as

$$\int_\Omega f(x)\,\mathrm{d}x = \frac{V(\Omega)}{N}\sum_{i=1}^{N} f(x_i) \tag{3}$$

where $V(\Omega)$ is the volume of $\Omega$. This is valid for any dimension and has little effect on computational time. In this paper, high dimensional integrals are computed numerically and so this method holds an advantage.

## 2.1 Markov Chain

A Markov chain is a sequence of states $(x_0,\ x_1, \ldots,\ x_n)$ such that any state $x_{i+1}$ depends only on its immediately proceeding state $x_i$, a property called memorylessness. The probability of the transition from state $x_i$ to state $x_{i+1}$ is the transition amplitude and defining such a function allows for the modeling of a Markov chain. Every model in this paper has the property of memorylessness and can be modeled as a Markov chain.

Moreover, every model can be modeled by Markov chain Monte Carlo methods. The equilibrium of a many-state system can be difficult, if not impossible, to solve. Rather, starting with an initial state $x_0$, the system will fall into equilibrium, or reach the target distribution, at state $x_n$ for some $n$ large enough. This is true because the algorithms in this paper maintain detailed balance and ergodicity so these algorithms will, eventually, converge to the correct probability distribution. The initial state $x_0$ is arbitrary although a guess of the equilibrium distribution is helpful so as to converge quicker. Beforehand, the equilibrium distribution is unknown but it can be approximated by evolving the Markov chain until the resulting distribution satisfies within an error the conditions of the equilibrium state. As stated before, there is a transition amplitude between states $x_i$ and $x_{i+1}$. So, for some change on $x_i$, a decision must be made whether it will change to $x_{i+1}$ or not change. The method used in this paper is called the Metropolis algorithm.

## 2.2 Metropolis Algorithm

For a Markov chain, there is a probability density $f(x_j|x_i)$ of choosing state $x_j$ as the next candidate in the Markov chain given state $x_i$. The change in $x_j$ from $x_i$ is usually small. This is to allow for a large number of accepted changes but it isn't too small so as to not change the state negligibly. There is also the conditional probability $P(x_j|x_i)$ of changing to state $x_j$ given state $x_i$. If we assume the process is reversible [7], then it satisfies the condition of detailed balance

$$q(x_j|x_i)P(x_i) = q(x_i|x_j)P(x_j) \tag{4}$$

where $P(x)$ is the desired probability distribution and $q(x_j|x_i)$ is the transition amplitude from $x_i$ to $x_j$. Noting that $q(x_j|x_i) = f(x_j|x_i)P(x_j|x_i)$, this can be rewritten as

$$\frac{P(x_j|x_i)}{P(x_i|x_j)} = \frac{f(x_i|x_j)q(x_j)}{f(x_j|x_i)q(x_i)}. \tag{5}$$

We wish to find the probability $P(x_j|x_i)$ that satisfies equation 5. The Metropolis choice is

$$P(x_j|x_i) = \min\left(1,\ \frac{f(x_i|x_j)P(x_j)}{f(x_j|x_i)P(x_i)}\right). \tag{6}$$

Thus, the probability of changing from state $x_i$ to state $x_j$ is shown in equation 6. The Metropolis steps are as follows:

1. Initialize the system with some initial state $x_0$.
2. Randomly pick a state $x_j$ with probability given by $f(x_j|x_i)$ where $x_i$ is the current state.
3. Accept this new state with probability $P(x_j|x_i)$ as given in equation 6.
4. If accepted, change the state from $x_i$ to $x_j$.
5. Otherwise, keep the state $x_i$.
6. Repeat steps 2-5.

For the models in this paper, a uniform distribution is used for choosing a new state $x_j$, thus $f(x_j|x_i) = f(x_i|x_j)$. Equation 6 can be rewritten as shown in [7] as

$$q(x_j|x_i) = \min\left(1,\ \frac{P(x_j)}{P(x_i)}\right). \tag{7}$$

From the above equation, we can see that if the probability of state $x_j$ is equal to or greater than the probability of $x_i$ (i.e. the new state is more likely in the equilibrium distribution than the old state), then the change is automatically accepted. Thus acceptance of a new state occurs with certainty when $P(x_{new}) \geq P(x_{old})$. This can once more be rewritten in a piecewise fashion as

$$P(x_{i+1}|x_i) = \begin{cases} 1 & \text{if } \dfrac{P(x_{i+1})}{P(x_i)} \geq 1 \\ \dfrac{P(x_{i+1})}{P(x_i)} & \text{otherwise.} \end{cases} \tag{8}$$

The functions for $P(x_i)$ will be defined later in the paper for the specific models. The Metropolis algorithm allows for the evolution of a statistical system without necessarily the knowledge of the desired probability distribution. But Monte Carlo can also be used in the measurement of observables of the system; after all, not much use is gained by modeling a system if no measurements are made on it.

## 2.3 Monte Carlo Measurements

As shown in equation 3, Monte Carlo has its uses in the numerical integration of multi-dimensional integrals. We will see that the latter two models in this paper are derived from the Feynman path integral which is, essentially, an infinite dimensional path integral which accounts for every possible path from one state to another (since in quantum mechanics, every path will have a nonzero probability due to effects such as tunneling).

When discretized, any integral is approximated by a sum. We will see that for some expectation average $\langle f(x)\rangle$ that is weighted by some probability density function $g(x)$, it can be written as

$$\langle f(x)\rangle = \int f(x)g(x)\,\mathrm{d}x\,. \tag{9}$$

But if some $x$ has a probability of being chosen proportional to $g(x)$, then the unweighted, simple average

$$\langle f(x) \rangle \approx \overline{f(x)} = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{10}$$

can be used as an approximation by [4]. The term $\overline{f(x)}$ is the Monte Carlo estimator for $\langle f(x) \rangle$. Since $N$ is finite, the estimator will never be exact. In the limit, $\overline{f(x)} \to \langle f(x) \rangle$ as $N \to \infty$. This result allows for the approximation of any expectation value where the error is dependent on $N$.

## 3 The Ising Model

The Ising model is a good place to begin with the Metropolis algorithm because of its simplicity. On the lattice for the Ising model, the interaction is nearest-neighbor and discrete. For each lattice site $k$, there is a corresponding spin $s_k \in \{-1, 1\}$. Given this, the Hamiltonian for the Ising model is

$$H = -\sum_{\langle i,j \rangle} J_{i,j} s_i s_j + h \sum_j s_j \tag{11}$$

where $J_{i,j}$ is the interaction strength between neighbors at site $i$ and site $j$ and $h$ is an external magnetic field. If we assume no external magnetic field and a constant interaction strength, then we obtain

$$H = -J \sum_{\langle i,j \rangle} s_i s_j. \tag{12}$$

Since each site has two possibilities, for a $d$-dimensional hypercube with side length $n$, there are $2^{n^d}$ possibilities for the lattice. This is the advantage of using Monte Carlo techniques. Many measurements of the lattice can be made to simulate the model and derive an accurate evolution on which observables can be measured. In the case for the Ising model, and subsequent models, the Metropolis algorithm is used.

### 3.1 Application of the Metropolis Algorithm

For the evolution of the Ising model, one must decide whether a change in the lattice should occur or not. The Metropolis algorithm gives the probability $P(\mathbf{x}_{\ell+1} \mid \mathbf{x}_\ell)$ for a transition from state $x_\ell$ to $x_{\ell+1}$. If the change in energy $\Delta H$ is less than zero, then the change at the site will occur since the system tends towards a lowest energy state. Otherwise, the transition probability follows the Boltzmann distribution $e^{-\beta \Delta H}$ where $\beta = 1/k_B T$ [1]. Thus, the Metropolis transition probability is

$$P(\mathbf{x}_{\ell+1} \mid \mathbf{x}_\ell) = \begin{cases} e^{-\beta \Delta H}, & \text{if } \Delta H > 0 \\ 1, & \text{otherwise.} \end{cases} \tag{13}$$

To obtain the change in energy $\Delta H$, a new state must be chosen for the lattice site. In the case of the Ising model, there are only two states: $\pm 1$. Therefore, the new energy is just the opposite sign of the old energy according to equation 12. Thus, the change in energy at lattice site $j$ is double the original energy at that site, or

$$\Delta H_j = -2 J s_j \sum_i s_i. \tag{14}$$

So the change in energy can be calculated this way since the new spin being chosen for site $j$ is implicit in the equation. For later simulations, this is not the case and so a slightly different state is chosen randomly to calculate the change induced by the new state. Now to apply this, the steps for the evolution of the Ising model are as follows:

1. Choose a lattice site $j$.
2. Calculate the change in energy $\Delta H_j$ at the site $j$.
3. If $\Delta H < 0$ then change the spin at site $j$.
4. Otherwise assign it a probability $\exp(-\beta \Delta H_j)$ of changing.
5. Repeat steps 1-4 for every site on the lattice.

This completes one sweep of the lattice as every site on the lattice is checked to flip. Step 4 can be completed by randomly choosing a number $a \in [0, 1]$ such that if $a < \exp(-\beta \Delta H)$ the change occurs otherwise the change is rejected.

A problem faced by virtue of a finite lattice is the boundary. In this model, the boundary lattice sites are missing neighbors due to the limitations of a computer simulation. One solution to this problem, which is used in the simulations in this paper, is periodic boundary conditions. The missing neighbors of the boundary sites are replaced by the sites on the opposite side of the lattice.

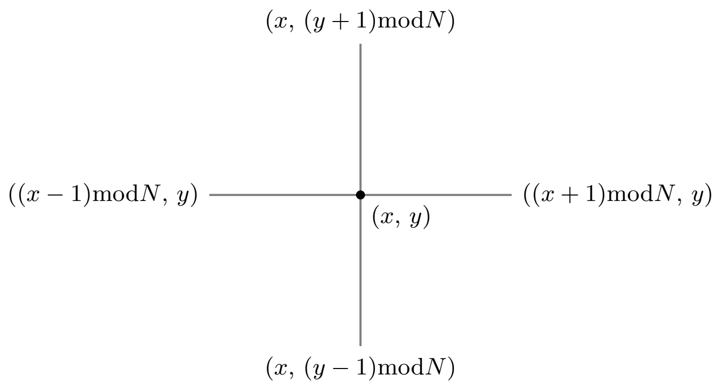
$(x,\,(y+1)\mathrm{mod}N)$
$((x-1)\mathrm{mod}N,\,y)$
$((x+1)\mathrm{mod}N,\,y)$
$(x,\,y)$
$(x,\,(y-1)\mathrm{mod}N)$

Figure 1: Neighboring sites of site $(x, y)$ with periodic boundary conditions

For example, on a square 2-dimensional lattice with sides $N$, the neighbors of any site $(x, y)$ will be as shown in figure 1 where mod represents the modulo operator. As long as the interactions are shorter than the lattice size, then the lattice is effectively infinite for each point. The interactions for the Ising model are nearest-neighbor, so periodic boundary conditions are applied for the boundary sites.

## 3.2 Measurements on the Lattice

### 3.2.1 Magnetization

Now with the ability to model the Ising Model, measurements can now be made. It is helpful to note that dimensionality has so far been arbitrary with exception to the remarks on figure 1. For

$d$ dimensions, for each lattice site $(x_1, x_2, \ldots, x_d)$, it would have $2d$ neighbors for measurement of the change in energy. One measurement is magnetization given by

$$M = \sum_{j=1}^{N^2} s_j \tag{15}$$

and the magnetization per spin is just $m = M/L^2$. What should be expected for the magnetization of this system?

From equation 13, one can see there are just two parameters: the temperature $T$ and the interaction strength $J$. But these can just be written as one parameter $T_J = J/k_B T_c$ (with the Boltzmann constant included) since they have the same effect on $\Delta H$. At some point, the system will reach an equilibrium where $m$ will oscillate about some value. Because $T_J > 0$, the interactions are ferromagnetic and so spins will tend to align in the same direction. This is seen at low $T$ since the thermal energy is not large enough to overcome the alignment. Therefore, the lattice will be highly ordered with $|m| \approx 1$ or slightly less. But with high $T$, the magnetic interactions will be dominated by the thermal energy and neighboring spins will not be correlated; so $m \approx 0$. There is some temperature at which the thermal energy is larger than the magnetic interactions of the lattice sites. This is the critical temperature $T_c$.

```
def Ising2D(lattice, Jkbt, N):
    magnetization = 0

    for p in product(range(N), range(N)):
        i, j = p
        energy = (lattice[i][(j - 1) % N] + lattice[i][(j + 1) % N] +
                  lattice[(i - 1) % N][j] + lattice[(i + 1) % N][j]) *
                  lattice[i][j]
        magnetization += lattice[i][j]

        if energy <= 0 or np.random.random() < np.exp(-2 * Jkbt * energy):
            lattice[i][j] *= -1

    return lattice, magnetization
```

Figure 2: Python code for one sweep of a $N$x$N$ lattice with $T_J$ = `Jkbt`. The change in energy, $\Delta H$, for a site is calculated and stored in `energy` where then a Metropolis update is performed. Also, the magnetization at that site is stored in `magnetization` and is used later to determine the equilibrium magnetization of the lattice. This is carried out for every site on the lattice, then returning the updated lattice and its total magnetization.

In figure 2, the function that completes one sweep of the lattice is shown. Each lattice site is checked to change before the next site is checked. The `product` function in the for loop creates tuples whose values are stored in `i` and `j` for each loop thus completing the sweep row by row. If the energy change is accepted according to the if statement, the spin value at the lattice site is flipped. It is important to note that the lattice must be thermalized before measurements can be taken, that is the lattice must first reach an equilibrium. This is done by completing enough lattice sweeps before taking measurements. The Ising model simulations discussed are done with 10000 sweeps and only the latter fourth of sweeps are measured. The application of this function is shown in its entirety in the appendix.

For the parameter $T_J$, the critical temperature is at $T_J = \frac{1}{2}\ln\left(1+\sqrt{2}\right)$ [1]. In figure 3, the behavior of different temperatures on different lattice sizes is shown. The absolute magnetization is measured instead of the signed magnetization because the magnetization at which a specific lattice settles to can be either $\pm x$ for $x \in [0,1]$ and the magnetization can rarely but suddenly switch signs due to finite size of the lattice. Ignoring the sign fixes these issues and the result is shown in figure 3; for low $T$, the lattice is more ordered and the absolute magnetization is near 1. For high $T$, the magnetization oscillates about 0 as discussed above. Since the absolute value of the magnetization is measured, the values plotted for high temperature represent the magnitude of these oscillations. For the smaller lattice size (e.g. $N = 5$), the oscillations will be larger since each site will have a greater impact on the total magnetization of the lattice.

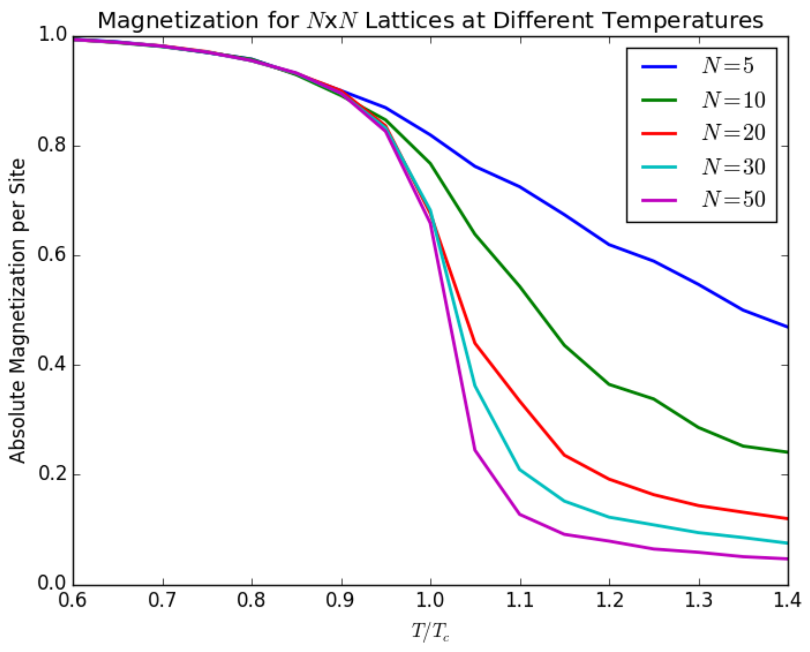

Figure 3: The normalized magnetization plotted against temperature with respect to the critical temperature for various lattice sizes. The values were computed by evolving the lattice until equilibrium, then computing the average for many measurements of the magnetization.

### 3.2.2 Susceptibility

Susceptibility can also be measured on the lattice. It is the measure of the change in magnetization due to a magnetic field. The effect of lattice size and temperature on magnetic susceptibility $\chi$ can also be measured by [3]

$$\chi = \frac{\partial\langle M\rangle}{\partial H} \tag{16}$$

$$= \frac{1}{T}(\langle M^2\rangle - \langle|M|\rangle^2). \tag{17}$$

This measurement only requires the average magnetization, which is already measured, and the average of the squared magnetization. In the simulation for figure 3, the susceptibility was also measured using equation 17 as shown in figure 4. A large susceptibility occurs where there is a large standard deviation in the magnetization of the sweeps (the susceptibility is the variance of the magnetization). At low $T$, the lattice is in equilibrium at some magnetization and at high $T$, the lattice is unordered and so the magnetization does not vary far from 0. In either case, the variance is small. At $T = T_c$, the lattice does not reach an equilibrium and so the change in magnetization is large accounting for the large peak in figure 4.

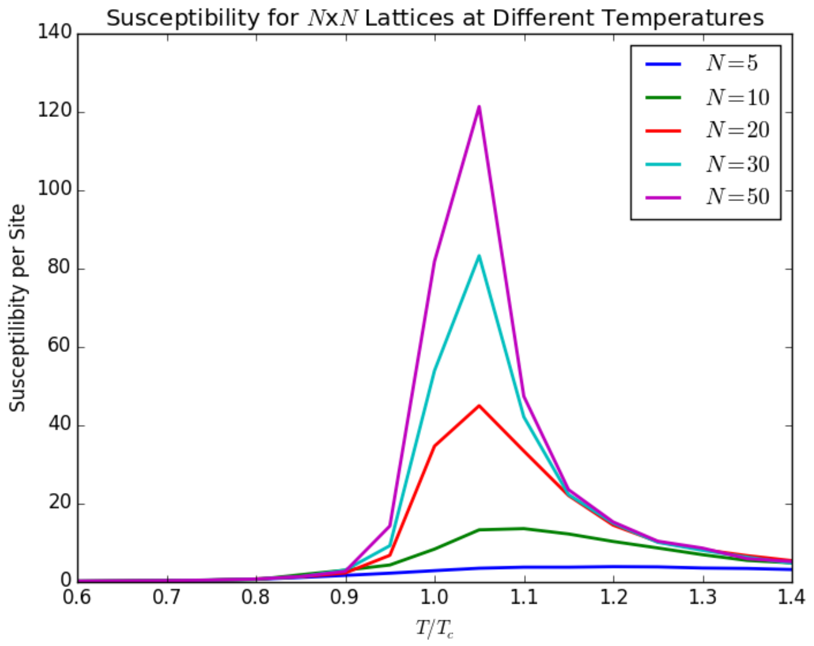

Figure 4: The normalized susceptibility for the same simulation as figure 3.

The Metropolis algorithm allowed us to model an Ising model system. With a model, measurements can be made on the system; the measurement of magnetization dependent on the parameters of the system was done above. Other measurements such as specific heat or correlation length can be made. Also, a variable interaction strength, nonzero external magnetic field or a higher dimensional model can have significant changes on the system with little change to the program. Going forward, the Monte Carlo method and the steps of the Metropolis algorithm used here will be applied for more complicated systems, specifically with respect to the discretized version of the Feynman path integral.

## 4 The Feynman Path Integral on the Harmonic Oscillator

### 4.1 Derivation

In a classical system, a path a particle will take in a potential field is determined by the principle of least action. This states that the action which can be written as a function

$$S[x] = \int_{t_i}^{t_f} \mathcal{L}(\mathbf{x}, \dot{\mathbf{x}})\, \mathrm{d}t\,, \tag{18}$$

where $\mathcal{L}(x, \dot{x})$ is the Lagrangian, gives the path of the particle when the action reaches a minimum. Classically, there is a unique path for a particle but quantum mechanically, due to its probabilistic nature, every path is possible and must be accounted for in calculations. With this in mind, how does one find the probability to go from $(x_i, t_i)$ to $(x_f, t_f)$ in some potential field when all paths are possible?

We start with the propagator in wave mechanics given by [6]

$$K(\mathbf{x}_f, t_f; \mathbf{x}_i, t_0) = \sum_a \langle \mathbf{x}_f | a \rangle \langle a | \mathbf{x}_i \rangle \exp\left[\frac{-iE_a(t_f - t_0)}{\hbar}\right] \tag{19}$$

which can be found as an integral operator for a wave function $\psi(\mathbf{x_f}, t_f)$ acting on the initial wave function:

$$\psi(\mathbf{x}_f, t_f) = \int \mathrm{d}^3x\, K(\mathbf{x}_f, t_f; \mathbf{x}_i, t_0)\psi(\mathbf{x}_i, t_i). \tag{20}$$

The propagator can be rewritten as

$$K(\mathbf{x}_f, t_f; \mathbf{x}_i, t_0) = \langle \mathbf{x}_f | \exp\left[\frac{-iH(t_f - t_0)}{\hbar}\right] | \mathbf{x}_i \rangle . \tag{21}$$

In both equations 20 and 21 and as hinted by the name, the propagator can be interpreted as a time-evolution operator on an initial state $(\mathbf{x}_i, t_i)$ to a final state $(\mathbf{x}_f, t_f)$. This can ultimately be seen by

$$\begin{aligned} K(\mathbf{x}_f, t_f; \mathbf{x}_i, t_0) &= \sum_a \langle \mathbf{x}_f | a \rangle \langle a | \mathbf{x}_i \rangle \exp\left[\frac{-iE_a(t_f - t_0)}{\hbar}\right] \\ &= \langle \mathbf{x}_f | \exp\left[\frac{-iH(t_f - t_0)}{\hbar}\right] | \mathbf{x}_i \rangle \\ &= \langle \mathbf{x}_f, t_f | \mathbf{x_i}, t_i \rangle . \end{aligned} \tag{22}$$

The Feynman path integral gives us a way to calculate the propagator. It is as follows from [4]:

$$\langle x_f, t_f | x, t_i \rangle = \int \mathcal{D}[x(t)] e^{iS[x]/\hbar} \tag{23}$$

where the position operators have been simplified to one dimension although the generalization for higher dimensions is analogous.

In equation 23, $\mathcal{D}[x(t)]$ represents all paths that the particle can take and is weighted by the $e^{iS[x]/\hbar}$ term. One concern is the divergence of the integral; it is an infinite dimensional integral with nonzero contributions from each dimension. But the integral will not diverge. Since for paths far from the classical path, the action will be large yielding a large frequency and thus a large phase shift compared to paths close to it. This will deconstructively interfere with other paths far from the classical path thus contributing little to the integral. Also at the classical $\hbar \to 0$ limit, the same behavior will be seen and the only surviving path is the one with a minimized action.

It is more convenient to work in Euclidean space so a change of variables $t \to it$ by a Wick rotation allows for simpler calculations for computers but does not effect the physics. Furthermore, using natural units such that $\hbar = c = 1$, the above equation is further simplified. This will give

$$\langle x_f, t_f | x, t_i \rangle = \int \mathcal{D}[x(t)] e^{-S[x]} \tag{24}$$

where one can see that both paths far from the classical path contribute little and the $\hbar \to 0$ limit still holds true due to the exponential dampening.

## 4.2 Discretization

The above equation is called the propagator which gives us complete information about the system. To solve this path integral numerically, the propagator must be discretized. To do so, we can split the possible paths of the particle into $N-1$ time slices such that for one path, the particle will travel along $(x_1, t_1), \ldots, (x_N, t_N)$ [6]. Thus the previously infinite dimensional integral can be approximated as

$$\int \mathcal{D}[x(t)] = A \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_{N-1 \text{ integrals}} \mathrm{d}x_1 \ldots \mathrm{d}x_{N-1} \tag{25}$$

where $A$ is a normalization constant. Using this, we can write

$$\begin{aligned}\langle x_f, t_f | x, t_i \rangle \\ &= A \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \mathrm{d}x_1 \ldots \mathrm{d}x_{N-1}\, e^{-S[x_1]} \ldots e^{-S[x_{N-1}]} \\ &= A \int_{-\infty}^{\infty} e^{-S[x_1]}\, \mathrm{d}x_1 \ldots \int_{-\infty}^{\infty} e^{-S[x_{N-1}]}\, \mathrm{d}x_{N-1}\,.\end{aligned} \tag{26}$$

This discretization also effects the action. From equation 18, writing the Lagrangian as $\frac{1}{2}m\dot{x}^2 - V(x)$ and focusing only on the $j$th time slice, we have

$$S[x_j] = \int_{t_j}^{t_{j+1}} \frac{m\dot{x}^2}{2} + V(x)\, \mathrm{d}t \tag{27}$$

$$= \Delta t \left[ \frac{m}{2} \left( \frac{x_{j+1} - x_j}{\Delta t} \right)^2 + \frac{1}{2}(V(x_{j+1}) + V(x_j)) \right] \tag{28}$$

where $\Delta t$ is the length of each time slice or $\Delta t = \frac{t_f - t_i}{N}$. The energies are summed because of the change from Minkowski to Euclidean space; a factor of $i^2$ comes from the kinetic term and the minus signs are factored out, effectively changing the Lagrangian into a Hamiltonian. Because time is discrete, the integral can be solved just by evaluating the endpoints which yields equation 28. Summing over all $N-1$ time slices, we obtain the action contribution from every path. Therefore,

$$\langle x_f, t_f | x, t_i \rangle = A \int_{-\infty}^{\infty} \mathrm{d}x_1 \ldots \int_{-\infty}^{\infty} \mathrm{d}x_{N-1}\, e^{-S[x]} \tag{29}$$

where

$$S[x] = \sum_{j=0}^{N-1} \left[ \frac{m}{2\Delta t}(x_{j+1} - x_j)^2 + \Delta t V(x_j) \right]. \tag{30}$$

In equation 29, the contribution of paths are exponentially damped so quantum mechanical paths (paths far from the classical path) contribute very little to the path integral because of their low probability. The path integral has now been discretized in terms of $N-1$ time slices where, for each slice, the action is assumed to be constant. So as $\Delta t \to 0$, equation 23 is recovered.

It is important to note that if $\Delta t$ is large and $T = t_f - t_i$, then the propagator can be rewritten as

$$\langle x | e^{-HT} | x \rangle \approx e^{-E_0 T} |\langle x | E_0 \rangle|^2 \tag{31}$$

since the ground state dominates due to the exponential. So assuming a large $\Delta t$, the propagator is effectively measured for the ground state wave function [4]. Shown below, $\Delta t$ is taken to be 1/2 and $N = 8$, so then $T = 4$.

### 4.3 Example

The integral was computed for a one-dimensional harmonic oscillator with

$$V(x) = \frac{x^2}{2}, \quad A = \left(\frac{m}{2\pi a}\right)^{N/2} \quad \text{and} \quad \Delta t = \frac{1}{2}. \tag{32}$$

The mass was set at $m = 1$ and the dimension of the integral was set at $N = 8$.

The integral was computed using Mathematica as shown in figure 5. The blue line represents an analytic approximation of the solution. As $N$ gets large, this numerical approximation will become more accurate but at the cost of computing power which is a common theme in numerical techniques.

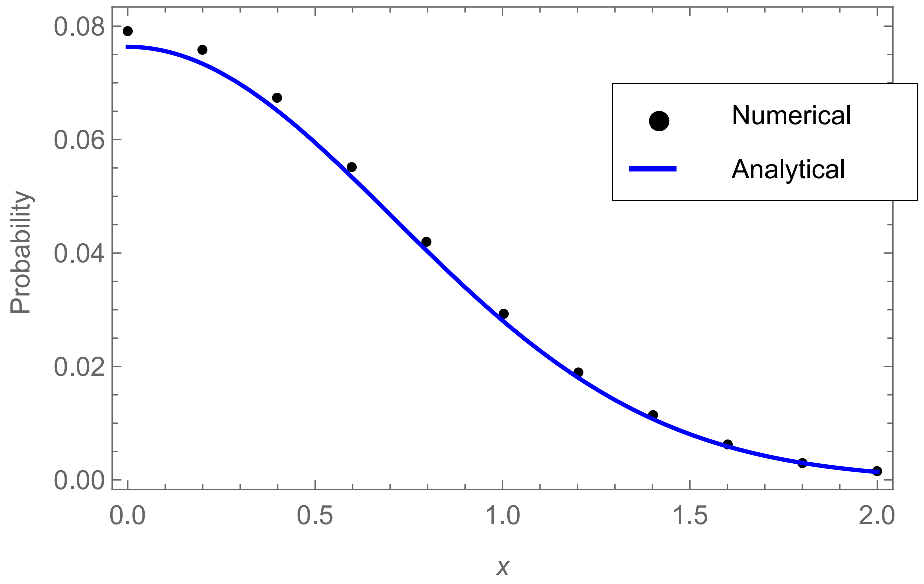

Figure 5: Plot of $\langle x|e^{-Ht}|x\rangle$ for various values of $x$ calculated using equations 29 and 30. The blue line represents the analytic solution where $\langle x|e^{-H\Delta t}|x\rangle \approx |\langle x|E_0\rangle|^2 e^{-E_0\Delta t}$ and $\langle x|E_0\rangle = \exp\left(-x^2/2\right)/\pi^{1/4}$.

### 4.4 Monte Carlo Application

#### 4.4.1 Manipulation for Monte Carlo

The above derivation is in regards to the ground state of the system. In systems such as those found in quantum field theory, the ground state is a vacuum and the measurement of observables in excited states is desired. The vacuum expectation value $\langle 0|x(t_2)x(t_1)|0\rangle$ can be written as the correlation function [5]

$$\langle x(t_1)x(t_2)\rangle = \frac{\int \mathcal{D}[x(t)]x(t_2)x(t_1)e^{-S[x]}}{\int \mathcal{D}[x(t)]e^{-S[x]}} \tag{33}$$

where $t_i < t_1 < t_2 < t_f$. Notice that equation 33 is written in the Heisenberg picture; the operators are time-dependent (in this case the position operators). If we change to the Schrödinger picture, this equation can be rewritten keeping in mind the rotation to Euclidean time as

$$\langle x(t_1)x(t_2)\rangle = \frac{\int \mathrm{d}x\ \langle 0|e^{-H(t_f-t_2)}|x\rangle\ \langle x|e^{-H(t_2-t_1)}|x\rangle\ \langle x|e^{-H(t_1-t_i)}|0\rangle}{\int \mathrm{d}x\ \langle 0|e^{-H(t_f-t_i)}|0\rangle}. \tag{34}$$

The numerator can be seen as an interruption on equation 23 with the position operator $x$ at $t_1$ and $t_2$. Simplifying, we have

$$= \frac{\int \mathrm{d}x \, \langle 0|e^{-H(t_f-t_2)} x e^{-H(t_2-t_1)} x e^{-H(t_1-t_i)}|0\rangle}{\int \mathrm{d}x \, \langle 0|e^{-H(t_f-t_i)}|0\rangle}. \tag{35}$$

A further simplification can be made by inserting the the complete set of states $\sum_i |E_i\rangle \langle E_i| = 1$ twice for $i = m, n$ next to the vacuum bra and ket. Letting $T = t_f - t_i$ and $t = t_2 - t_1$, this ultimately yields [4]

$$= \frac{\sum_n e^{-E_n T} \langle E_n|x e^{-(H-E_n)t} x|E_n\rangle}{\sum_n e^{-E_n T}}. \tag{36}$$

Making the assumption that $T$ is large and $T \gg t$, the first term will dominate due to the exponential and we end up with

$$\langle x(t_1)x(t_2)\rangle = \langle E_0|x e^{-(H-E_0)t} x|E_0\rangle \,. \tag{37}$$

For the harmonic oscillator, if a complete set of states is inserted, only the $E_1$ states survive and thus we have

$$\langle x(t_1)x(t_2)\rangle = \langle E_0|x|E_1\rangle \, \langle E_1|e^{-(H-E_0)t}|E_1\rangle \, \langle E_1|x|E_0\rangle \tag{38}$$

$$= |\langle E_0|x|E_1\rangle|^2 e^{-(E_1-E_0)t} \tag{39}$$

where we can now find the difference $E_1 - E_0$. There is the unknown transition amplitude matrix element but if we let $G(t) = \langle x(t_1)x(t_2)\rangle$, then

$$E_1 - E_0 = \frac{1}{a} \log\left[\frac{G(t)}{G(t+a)}\right]. \tag{40}$$

The quantity $G(t)$ can be considered as a weighted average from equation 33 with the weight being the exponential $\exp(-S[x])$. As stated in the first section, if the probability of the paths being drawn is proportional to this weight and the number of paths is large, then the weighted average can be approximated with an unweighted average. This means that

$$G(t) = \langle x(t_1)x(t_2)\rangle \approx \frac{1}{N} \sum_{n=1}^{N} x_n(t_1)x_n(t_2). \tag{41}$$

We now have everything needed to write a Metropolis algorithm program to generate an ensemble of paths. The process is analogous to the Ising model but with one exception: the value a site (or, in this case, a time slice) may have is not limited to only either $+1$ or $-1$ (as with the Ising model). Rather, a small random number $\alpha$ is added to the time slice to determine if such a change will be accepted by the Metropolis algorithm. Below are the steps to generate one path:

1. Choose a site $j$ in path $x$.
2. Add a random number $\alpha \in (-\epsilon, \epsilon)$ to the site $x_j$ so that $x_j \to x_j + \alpha$.
3. Calculate the change in action $\Delta S$ due to the change in $x_j$.
4. If $\Delta S < 0$, then change the spin at site $j$.

5. Otherwise assign it a probability $\exp(-\Delta S)$ of changing.
6. Repeat steps 1-5 for every site on the path.

The change in action can be written as

$$\Delta S = \frac{\alpha}{a}\left[(a^2+2)x_j + \left(\frac{\alpha^2}{2}+1\right)\alpha - x_{j+1} - x_{j-1}\right] \tag{42}$$

and, as with the Ising model, periodic boundary conditions are used, so the modulus operator is implied for $x_{j+1}$ and $x_{j-1}$. Figure 6 shows the code that applies the steps above.

```
def dS(j, alpha):
    deltaS = (alpha / a) * ((a**2 + 2)*x[j] + (a**2/2 + 1)*alpha
              - x[(j + 1) % N] - x[(j - 1) % N])
    return deltaS

def update(x):
    for j in range(N):
        alpha = np.random.uniform(-eps, eps)
        deltaS = dS(j, alpha)
        if deltaS < 0 or np.random.random() < np.exp(-deltaS):
            x[j] += alpha
```

Figure 6: Python code for one configuration of a path split into $N$ slices.

A few details must be brought up before making measurements as discussed in [4]. The first is the starting configuration. An initial configuration should be used that is close to the equilibrium. This allows the program to reach equilibrium quicker. A initial configuration of $x_j = 0$ for all $j$ is used in this program. One detail is the value of $\epsilon$. This number determines the maximum at which a site can change and there are pros and cons to having both a large or small $\epsilon$. If $\epsilon$ is small, many changes will be accepted but each successive path will be highly correlated. On the other extreme, successive paths will not be very correlated but changes will rarely be accepted. Therefore an $\epsilon$ is chosen by trial and error and $\epsilon = 1.4$ is used in this code. Correlated paths are not desirable because the evolution of the path should be independent of previous values of the path. Therefore, not every path created is kept; for this model, every $N_{cor}$th path is recorded where $N_{\text{cor}} = 20$. This number is inversely proportional to the lattice spacing $a$ and goes roughly as $N_{\text{cor}} \propto 1/a^2$. In this model, $a$ is large ($a = 1/2$), so the choice of $N_{\text{cor}}$ is appropriate. Lastly, the paths must be thermalized before measurements are taken. In other words, the system must reach an equilibrium before paths are recorded and correlation functions measured. The appropriate number for this can also be measured by trial error. Here, $5N_{\text{cor}}$ paths are thrown away before measurements are taken.

### 4.4.2 Measurements

The quantity $G(t)$ can be measured numerically by using equation 41 since the measurement is a simple product but other correlation functions could be calculated and used by changing that which is used in the function `computeG` in figure 7. The potential (in this case, that of the harmonic oscillator) can also be changed in the function `dS` in figure 6.

The correlation function requires the correlation between each pair of sites on the path to be measured for each path. This is shown in figure 7. For this code, 10000 paths are created and,

```
def computeG(x, n):
    g = 0
    for j in range(N):
        g += x[j] * x[(j + n) % N]
    return g / N

def MCPaths(G):
    for i in range(N_cf):
        for j in range(N_cor):
            update(x)

        for k in range(N):
            G[i][k] = computeG(x, k)
```

Figure 7: Python code for calculating the correlation function $G(t)$ assuming it as an unweighted average. The measurement made on the paths is made only after every `N_cor` paths and the `N` slices of each path in stored in the $i$th element of the array `G`.

to save computing time, binned into 100 different bins. Binning the paths does not change the statistics of the system, but makes calculations much quicker. Since $G(t)$ measures the correlation of two sites on the path and is the sum of a decreasing and an increasing exponential, we expect to see a curve similar to a hyperbolic cosine due to the periodic boundary conditions. This is seen in figure 8; path sites with a further distance between them are less correlated.

Using this data, the excitation energy from the ground state can be found by using equation 40. This can be rewritten as [4]

$$E_1 - E_0 = \frac{1}{a}\log\left[\frac{G_n}{G_{n+1}}\right] \tag{43}$$

where

$$G_n = \frac{1}{N}\sum_{j=1}^{N} x_{(j+n)\%N}x_j. \tag{44}$$

Using equation 43, $N = 20$ data points are averaged over 10000 paths. The result is shown in figure 9.

The error bars are calculated using the bootstrap method. An ensemble of $N_{cf}$ paths are generated and binned into 100 bins which create 100 new paths that contain the same distribution as the original $N_{cf}$ paths. From the new paths, 100 random paths are selected with replacement which we call the bootstrap paths. The result can contain the same path multiple times as a result. Measurements of $G_n$ and $E_1 - E_0$ are completed for the bootstrap paths. The bootstrap method can be calculated many times and distribution of these bootstrap measurements are used to estimate the statistical error of the Monte Carlo measurement of $G_n$ and $E_1 - E_0$.

Note that the energy of the quantum harmonic oscillator is $E_n = (n + 1/2)\hbar\omega$. So we are expected to see $E_1 - E_0 = 1$ which is what is modeled closely by the data. While there are $N = 20$ data points, only 6 are plotted for two reasons. Because of periodic boundary conditions, the data is mirrored for the latter half of the data points. Furthermore, the error grows, which can be seen in figure 9, as $t$ gets larger. This is due to randomness of the paths; the plot would asymptotically reach a value of 1 as the number of paths goes to infinity.

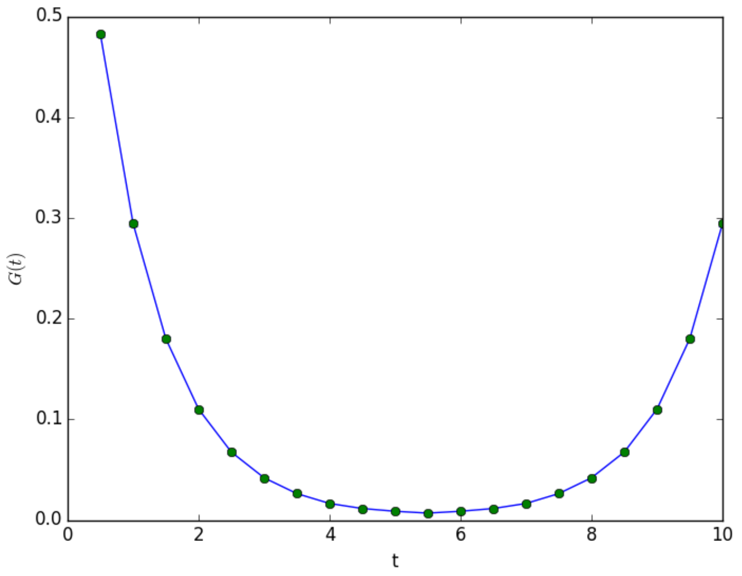

Figure 8: Measurement of the correlation function for $N = 20$ number of sites. The minimum for $G(t)$ is at the maximum distance (or maximum $t$) with periodic boundary conditions (therefore it is found at $N/2$).

The harmonic oscillator was used here because of its relatively simple potential $V(x) = x^2/2$ but any potential could be easily applied by using equation 30 and summing the terms of the sum that include the $j$th term. Then $\Delta S$ is the difference in action if the $j$th term has some small $\alpha$ added to it. The equation in figure 6 is the simplified equation of $\Delta S$ for the harmonic oscillator. As will be seen in the next section, different measurements can be made relatively easily once the system has been modeled correctly. Here, it was the path integral on a one-dimensional quantum mechanical system. In the next section, it will be the path integral in spacetime for gluons.

## 5 Lattice Quantum Chromodynamics

We now shift our focus to quantum chromodynamics (QCD). In some ways it is analogous in that instead of space coordinates $x(t)$, we now deal with the fields $A_\mu(x)$ or $U_\mu(x)$ where $x = (\mathbf{x}, t)$ is a spacetime point. The lattice is now four dimensional and thus size of the lattice $N$ will have a much greater effect on the computational time due to both the extra dimensions and the more involved calculations done per site on the lattice. So although the lattice size will be much smaller than previous models, computational time is much longer.

### 5.1 Derivation and Discretization

The value $A_\mu(x)$ represents the gauge field at the sites on the lattice but we cannot formulate a discretized version of QCD that is both gauge invariant and in terms of $A_\mu(x)$'s. Instead the variables on the links of the lattice representing the gluon fields, or the link variables, will be used.

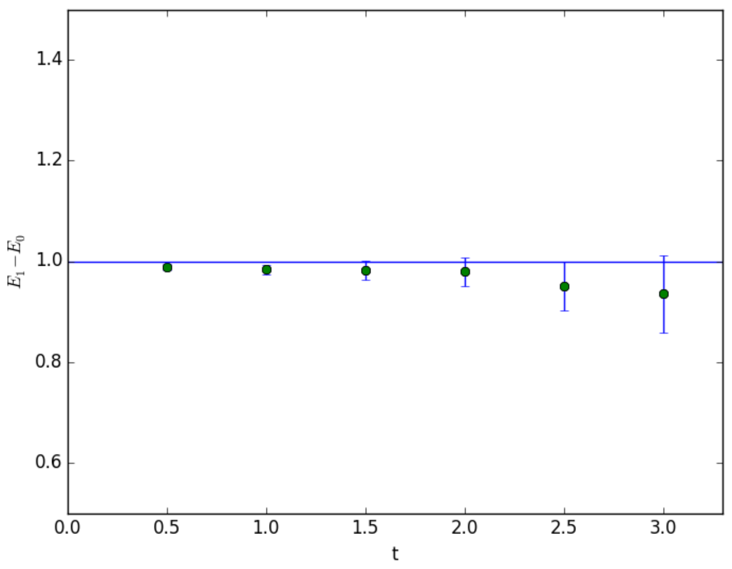

Figure 9: Excitation energy of the quantum harmonic oscillator from the ground state. The expected value $E_1 - E_0 = 1$ is represented by the line.

The link variable from some point $x$ to its neighboring point in the $\hat{\mu}$ direction is the line integral

$$U_\mu(x) = \mathcal{P}\exp\left(-i\int_x^{x+a\hat{\mu}} gA_\mu \,\mathrm{d}y\right) \tag{45}$$

where $\mathcal{P}$ path-orders the integral. These link variables are $SU(3)$ matrices and so gauge invariance is possible due to their gauge transformation:

$$U_\mu(x) \to V(x)U_\mu(x)V^\dagger(x + a\hat{\mu}). \tag{46}$$

for $V(x) \in SU(3)$.

Visually, two examples of the link variable can be seen in figure 10. For the links in the negative direction, the inverse is used. But since the links are $SU(3)$ matrices, the conjugate transpose is equivalent. It is important to note that the $U_\mu(x)$'s paths move away from $x$ while the $U_\mu^\dagger(x)$'s paths move towards $x$. Thus any path on the path can be created by a product of the link variables.

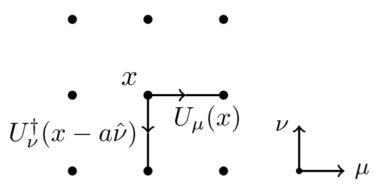

Figure 10: Two link variables on the $\mu\nu$ plane. Reverse directions are represented by the inverse of the link variable.

The simplest path that can be created is the plaquette as shown in figure 11. Mathematically, this can be written as

$$P_{\mu\nu} = \frac{1}{3} \operatorname{Re} \operatorname{Tr}(U_\mu(x) U\nu(x + a\hat{\mu}) U_\mu^\dagger(x + a\hat{\nu}) U_\nu^\dagger(x)). \tag{47}$$

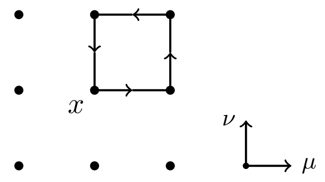

Figure 11: The plaquette which is the simplest guage invariant object that can be created from the link variables.

With this, the Wilson gauge action is

$$S = \beta \sum_x \sum_{\mu > \nu} [1 - P_{\mu\nu}(x)] \tag{48}$$

where $\beta = 6/g^2$ is the lattice coupling. The Wilson action is base on Wilson loops where the plaquette is an example of the simplest Wilson loop. It is the normalized ordered product of links in a path. It's used in equation 48 and will be used in the improved action derived shortly. For small $a$, if we expand the plaquette at some $x_0$ as a polynomial in $a$, due to $A_\mu$ being slowly varying because $a$ is small, we find

$$P_{\mu\nu} = 1 - \frac{a^4}{6} \operatorname{Tr}(g F_\mu\nu(x_0))^2 + \mathcal{O}(a^6) \tag{49}$$

where $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu + ig[A_\mu, A_\nu]$ is the field strength tensor. Taking the continuum limit $a \to 0$, what remains is the continuum action up to second-order shown in equation 50.

$$S = \int \mathrm{d}^4 x \, \frac{1}{2} \sum_{\mu,\nu} \operatorname{Tr} F_{\mu\nu}^2(x) \tag{50}$$

These higher-order terms complicate the action as such:

$$S = \int \mathrm{d}^4 x \sum_{\mu,\nu} \left[ \frac{1}{2} \operatorname{Tr} F_{\mu\nu}^2(x) + \frac{a^2}{24} \operatorname{Tr} F_{\mu\nu}(D_\mu^2 + D_\nu^2) F_{\mu\nu} + \dots \right] \tag{51}$$

The terms $D_\mu$ and $D_\nu$ are gauge covariant derivatives for their respective directions.

To cancel out the higher-order term, we can introduce the rectangle operator. Similar to the plaquette in equation 47, it is the normalized real trace of the path in figure 12. The rectangle

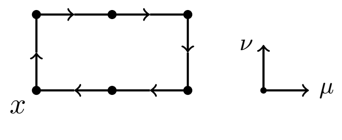

Figure 12: The rectangle operator at point $x$.

operator has as an expansion

$$R_{\mu\nu} = 1 - \frac{4a^4}{6} \operatorname{Tr}(g F_{\mu\nu})^2 - \frac{4a^6}{72} \operatorname{Tr}(g F_{\mu\nu}(4 D_\mu^2 + D_\nu^2) g F_{\mu\nu}) + \mathcal{O}(a^8). \tag{52}$$

With the different coefficients for the $a^4$ and $a^6$ terms, an improved action can be calculated which is accurate up to third-order:

$$S_{imp} = -\beta \sum_x \sum_{\mu>\nu} \left[ \frac{5}{3} P_{\mu\nu} - \frac{1}{12}(R_{\mu\nu} + R_{\nu\mu}) \right]. \tag{53}$$

One last modification to the above action is necessary for an accurate quantum action, that is an action that can accurately describe this system. This is the tadpole improvement, a renormalization $\mu_0$ that accounts for the extra terms in the action. Then equation 53 becomes

$$S_{imp} = -\beta \sum_x \sum_{\mu>\nu} \left[ \frac{5}{3\mu_0^4} P_{\mu\nu} - \frac{1}{12\mu_0^6}(R_{\mu\nu} + R_{\nu\mu}) \right] \tag{54}$$

where the $\mu_0$'s cancel the tadpole contributions. The $\mu_0$'s depend only on the lattice spacing and can be found numerically; for example, for a lattice spacing of $a = 0.4$ fm, then $\mu_0 \approx 3/4$. In this case, the contribution by the rectangle operators is increased by about a factor of 2 compared to the action without the tadpole improvement. With the action in equation 54, Monte Carlo steps can be used to run this model.

## 5.2 Monte Carlo Application

Again, the steps of Monte Carlo in this context is analogous to before. There are added complications that must be sorted out and, in doing so, increase the computing time appreciably. As stated before, the lattice is four-dimensional to account for three spacial dimensions and one temporal dimension. Furthermore, the calculations for the model exist at the links of which each lattice site has four.

We begin by looking at an $N$x$N$x$N$x$N$ lattice with coordinates $(x,\ y,\ z,\ t)$ at the site $x$ and the link in the $\hat{x}$ direction[1]. There are six plaquettes and twelve rectangle operators that affect the link $U_x(x)$ but for now we will focus only on the plaquettes. Figure 13 shows these plaquettes. The plaquettes are equivalent for each plane that $U_x(x)$ is found on and the $yz$, $yt$ and $zt$ planes are not included (even though they are part of the sum in equation 54) because they do not contain plaquettes of $U_x(x)$ and therefore do not contribute to the action for this link. So for any link, three of the planes will not contribute. It is important to note that each plane has a contribution of $U_\mu(x)$ and of $U_\mu^\dagger(x)$ for some $\mu$-direction.

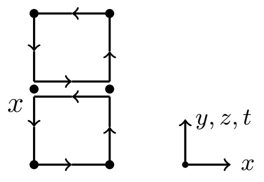

Figure 13: Two plaquettes that effect the link $U_x(x)$ with one being the inverse path or $U_x^\dagger(x)$. Since the lattice is four dimensional, the three planes of $x$ are analogous for calculation of $U_x(x)$.

The picture for $U_t(x)$ is slightly different. Because $n_t > n_i$ for all $i \in \{x, y, z\}$, the plaquette at site $x$ contains the inverse link of $U_t(x)$. Thus the plaquette at site $x - a\hat{\mu}$ must be used as the plaquette containing $U_t(x)$ as shown in figure 14. For links in the $t$-direction, this is always true while for links in the $x$-direction, this is never true. The application of this can be seen in the code in figure 16 which will be discribed shortly.

[1]In the code, the variable $n \in \{0, 1, 2, 3\}$ is used to determine the specific link. This follows more easily with the sum in the action that is over $\mu > \nu$, thus $n_x = 0 \Leftrightarrow \hat{x}$, $n_y = 1 \Leftrightarrow \hat{y}$, etc.

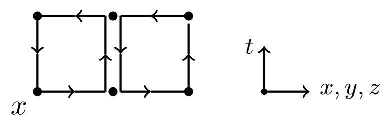

Figure 14: Two plaquettes that effect the link $U_t(x)$. This picture holds for the planes where $\mu > \nu$.

The rectangle operators are analogous but there is one important difference. Since these operators are $a \times 2a$ and not $a \times a$ as are the plaquettes, there are more contributions to the operator. This can be seen in figure 15. As with the plaquette, the paths in this figure are for the $x$-direction but these paths will not be so for the $t$-direction. Rather, there is a rotation as before and for the same reason. Also, from equation 54, there are terms $R_{\mu\nu}$ and $R_{\nu\mu}$. Thus there are six rectangle operators, double the number of plaquettes. For $U_x(x)$, there are only two staples for $R_{\nu\mu}$ and four for $R_{\mu\nu}$ and vis versa for $U_t(x)$. The staple being defined as the path operator without the path link in question. To sum up, if $n_\mu > n_\nu$, then $R_{\mu\nu}$ will be a sum of four staples and $R_{\nu\mu}$ will be the sum of two.

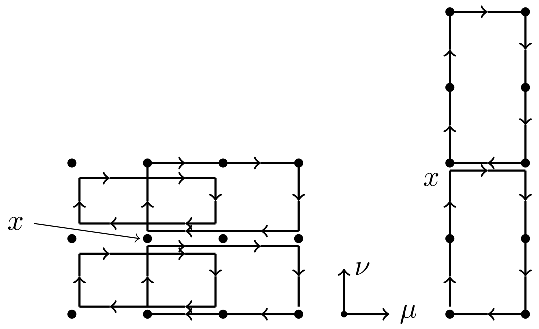

Figure 15: The possible configurations of the rectangle operator. Depending on the direction, the rectangles may be rotated as with the plaquettes. For $U_t(x)$, the rectangles are rotated 90° relative to $U_x(x)$.

In the code in 16, the function `nPlaq` calculates the staple and conjugate staple for the link at lattice site $(x,\ y,\ z,\ t,\ n)$ where $n$ specifies the link. The values `staples` and `stapleC` are the sums of the staples and conjugate staples, respectively. They are used to calculate $\Delta S$ (along with the rectangle staples for the improved action).

In the previous model, a small random change $\epsilon$ was added to a time slice for a path $x$ and the resulting change in the action was calculated. The same is done again except the path links are $SU(3)$ matrices, not numbers, so a 'small' random $SU(3)$ matrix is added to the path link, that is a matrix close to the identity. Thus we take $U \to MU$ if $U$ is a path link and $M \in SU(3)$ is close to the identity. The small change in $U$ is given by the product $MU$ which is also an $SU(3)$ matrix since $SU(3)$ is a group and thus is closed under this operation. This resulting change is used to calculate the change in the action, $\Delta S$.

Let $\mathcal{S}_{\mu\nu}(C)$ represent the staple(s)[2] for $U_\mu(x)$ in the $\mu\nu$ plane for path $C$ and $\mathcal{S}^*_{\mu\nu}(C)$ be the

[2]Here I say "staple(s)" because there can be more than one staple coming from the $\mu\nu$ plane like with the rectangle operator (where there can be two). It is implied these are being summed first in the equation which is more useful as a template for visualizing $\Delta S$ than a rigorous formula.

```
def nPlaq(x, y, z, t, n):
    a = [0] * 4
    a[n] = 1
    staples = 0
    stapleC = 0

    for i in range(3 - n):
        staples += mm(L[m(x+a[0])][m(y+a[1])][m(z+a[2])][t][n+i+1],
                        ct(L[x][m(y+a[-i])][m(z+a[1-i])][m(t+a[2-i])][n]),
                        ct(L[x][y][z][t][n+i+1]))
        stapleC += mm(ct(L[x][m(y-a[-i])][m(z-a[1-i])][m(t-a[2-i])][n+i+1]),
                        L[x][m(y-a[-i])][m(z-a[1-i])][m(t-a[2-i])][n],
                        L[m(x+a[0])][m(y+a[1]-a[-i])][m(z+a[2]-a[1-i])][t-a[2-i]][n+i+1])
    for i in range(n):
        staples += mm(ct(L[m(x-a[i+n])][m(y-a[i+n-1]+a[1])][m(z-a[i+n-2]+a[2])][m(t+a[3])][i]),
                        ct(L[m(x-a[i+n])][m(y-a[i+n-1])][m(z-a[i+n-2])][t][n]),
                        L[m(x-a[i+n])][m(y-a[i+n-1])][m(z-a[i+n-2])][t][i])
        stapleC += mm(L[x][y][z][t][i],
                        L[m(x+a[i+n])][m(y+a[i+n-1])][m(z+a[i+n-2])][t][n],
                        ct(L[x][m(y+a[1])][m(z+a[2])][m(t+a[3])][i]))
    return staples, stapleC
```

Figure 16: Python code for calculating the regular and conjugate transpose staples for $U_\mu(x)$, a link in direction $n_\mu$. The two for loops are to account for the change in the calculation of the staples for each link direction.

staple(s) for $U^\dagger_\mu(x)$. Then the change in the unimproved action from equation 48 for site $x$ is

$$\Delta S(x) = \frac{\beta}{3}\,\mathrm{Re}\,\mathrm{Tr}\left[(I-M)U_\mu(x)\sum_{\nu\neq\mu}\mathcal{S}_{\mu\nu}(P_{\mu\nu}) + U^\dagger_\mu(x)(I-M^\dagger)\sum_{\nu\neq\mu}\mathcal{S}^*_{\mu\nu}(P_{\mu\nu})\right] \tag{55}$$

and the improved action is

$$\begin{aligned}\Delta S_{imp}(x) = \frac{\beta}{3}\,\mathrm{Re}\,\mathrm{Tr}\Bigg[(I-M)U_\mu(x)\left(c_1\sum_{\nu\neq\mu}\mathcal{S}_{\mu\nu}(P_{\mu\nu}) - c_2\sum_{\nu\neq\mu}[\mathcal{S}_{\mu\nu}(R_{\mu\nu})+\mathcal{S}_{\mu\nu}(R_{\nu\mu})]\right)\\ +U^\dagger_\mu(x)(I-M^\dagger)\left(c_1\sum_{\nu\neq\mu}\mathcal{S}^*_{\mu\nu}(P_{\mu\nu}) - c^2\sum_{\nu\neq\mu}[\mathcal{S}^*_{\mu\nu}(R_{\mu\nu})+\mathcal{S}^*_{\mu\nu}(R_{\nu\mu})\right)\Bigg]\end{aligned} \tag{56}$$

where $M$ is a random $SU(3)$ matrix close to the identity, $c_1 = 5/3\mu_0^4$ and $c_2 = 1/12\mu_0^6$. With this in mind, one sweep of the lattice can be completed with the following steps:

1. Choose a site $x$ and a link $n$.
2. Choose a random $SU(3)$ matrix $M$ such that $U_\mu(x) \to MU_\mu(x)$.
3. Calculate the change in action $\Delta S$ with equation 56.
4. If $\Delta S < 0$, then keep the change $MU_\mu(x)$.

5. Otherwise assign it a probability $\exp(-\Delta S)$ of changing.

6. Repeat steps 1-5 for every site on the path.

```
def dS(staples, stapleC, rects, rectC, x, y, z, t, n):
    M = SU3list[np.random.randint(0, 2 * numMatrices)]
    U = L[x][y][z][t][n]
    deltaS = (beta / 3) * np.real(np.trace(np.dot(U - mm(M, U), c1 * staples - c2 * rects) + \
                                   np.dot(ct(U) - ct(mm(M, U)), c1 * stapleC - c2 * rectC)))
    return deltaS, M

def sweep():
    for p in product(range(N), range(N), range(N), range(N), range(4)):
        x, y, z, t, n = p
        staples, stapleC = nPlaq(x, y, z, t, n)
        rects, rectC = nRect(x,y,z,t,n)
        for _ in range(N_ma):
            deltaS, M = dS(staples, stapleC, rects, rectC, x, y, z, t, n)
            if deltaS < 0 or np.random.random() < np.exp(-deltaS):
                L[x][y][z][t][n] = np.dot(M, L[x][y][z][t][n])
```

Figure 17: The code that evolves the lattice. The array `L` is the lattice with $N^4 * 4$ sites given $N$ as the side length of the lattice and an $SU(3)$ matrix at each site with the total length of `L` being $36 * N^4$. The function `sweep` completes a sweep of the lattice and `dS` calculates the change in the action. At each site ($x$, $y$, $z$, $t$, $n$), the Metropolis algorithm is done `N_ma` times so as to let the link variable reach an equilibrium with its neighbors before moving on to the next link.

These steps are very similar to those from before, the main changes coming in the equation for the action which is much more costly in this model; the lattice is higher dimensional and the Monte Carlo steps require more computations and are computed on matrices. In figure 17, equation 56 is implemented under `dS` where the matrices staples and stapleC are calculated as shown in figure 16 and `rects` and `rectC` are calculated in a similar but longer function `nRect`. The $SU(3)$ matrices used as the small Monte Carlo variable are generated beforehand and stored in `SU3list` along with their inverses.

## 5.3 Measurements

With the derivation of the path integral in mind, we precede with the gluonic path integral similarly with $N = 8$ as a lattice size[3]. We can measure the effect of the Wilson loop operator on the vacuum state and, with the choice of Wilson loop, the measurement will change. First, the lattice is thermalized, as defined earlier, by completing 100 sweeps and then measurements are taken every $N_{cor} = 50$ sweeps. The trace of the Wilson loops for each path link are summed and averaged. In figure 18, the convergence of the Wilson loops are shown. After 100 sweeps, the lattice has reach equilibrium and 50 sweeps is sufficiently long enough to prevent correlation in the measurements [4].

[3]Even though this lattice size is much smaller than the previous, this simulation requires hours, compared to minutes, to complete 25 measurements with 50 sweeps between measurements. All code was run in Python 3.x on an Intel i7 processor on Windows 10.

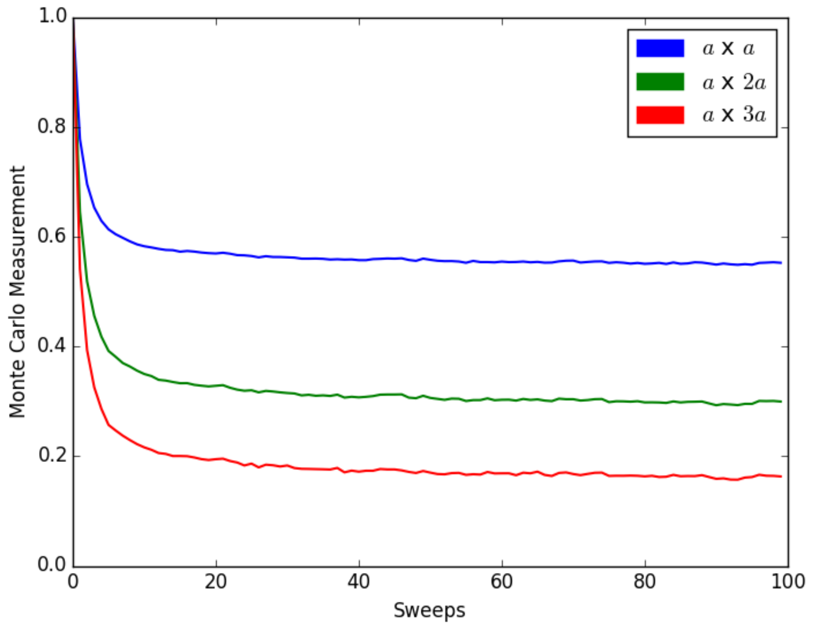

Figure 18: A visualization of the convergence rates of the Monte Carlo measurements. The first 100 sweeps, which are plotted, are used for reaching the equilibrium of the lattice and so measurements are not completed during these sweeps.

The Monte Carlo averages of $a \times a$, $a \times 2a$ and $a \times 3a$ Wilson loops are calculated as shown in the code in figure 19. In general, measurement of an arbitrary Wilson loop $W$ requires only a change in the product of the link variables.

The averages are also calculated for both the unimproved and improved actions:

$$S(x) = \frac{\beta}{\mu_0^4} \sum_{\mu > \nu} P_{\mu\nu}(x) \qquad S_{imp}(x) = -\beta \sum_{\mu > \nu} \left[ \frac{5}{3\mu_0^4} P_{\mu\nu} - \frac{1}{12\mu_0^6} (R_{\mu\nu} + R_{\nu\mu}) \right]. \tag{57}$$

This allows us to visual the difference in the second- and third-order accuracy of the discretized action, i.e. the contribution of the rectangle operators. For the unimproved and improved actions, the averages, denoted by $\mathcal{W}$ and $\mathcal{W}_{imp}$, respectively, are

$$\begin{aligned} \mathcal{W}(a \times a) &= 0.497 \pm 0.003 & \mathcal{W}_{imp}(a \times a) &= 0.546 \pm 0.003, \\ \mathcal{W}(a \times 2a) &= 0.260 \pm 0.004 & \mathcal{W}_{imp}(a \times 2a) &= 0.290 \pm 0.004, \\ \mathcal{W}(a \times 3a) &= 0.137 \pm 0.004 & \mathcal{W}_{imp}(a \times 3a) &= 0.156 \pm 0.003. \end{aligned} \tag{58}$$

These are the unweighted means for the respective Wilson loops as shown in figure 19. These results suggest that the larger the area of the Wilson loop, the smaller $\mathcal{W}$ will be.

With the system modeled based on the assumptions of the system, the measurements are relatively trivial in both computational time and in the complexity of the code. Further measurements can be made on this lattice or more realistic properties can be assumed for the lattice, although the latter may increase the computing time significantly due to its added complexity. Nevertheless, as with the previous models, the potential for added measurements exists.

```
def measure(loopLength):
    LAvg = 0
    for p in product(range(N), range(N), range(N), range(N)):
        x, y, z, t = p
        link = measureLink(x, y, z, t, loopLength)
        LAvg += np.real(np.trace(
        mm(L[x][y][z][t][0], L[m(x+1)][y][z][t][1], link[0], ct(L[x][y][z][t][1])) +
        mm(L[x][y][z][t][0], L[m(x+1)][y][z][t][2], link[1], ct(L[x][y][z][t][2])) +
        mm(L[x][y][z][t][0], L[m(x+1)][y][z][t][3], link[2], ct(L[x][y][z][t][3])) +
        mm(L[x][y][z][t][1], L[x][m(y+1)][z][t][2], link[3], ct(L[x][y][z][t][2])) +
        mm(L[x][y][z][t][1], L[x][m(y+1)][z][t][3], link[4], ct(L[x][y][z][t][3])) +
        mm(L[x][y][z][t][2], L[x][y][m(z+1)][t][3], link[5], ct(L[x][y][z][t][3]))))
    return LAvg / N**4 / 18
```

Figure 19: Python code that measures an $a \times ca$ Wilson loop for $c \in \{1, 2, 3\}$.

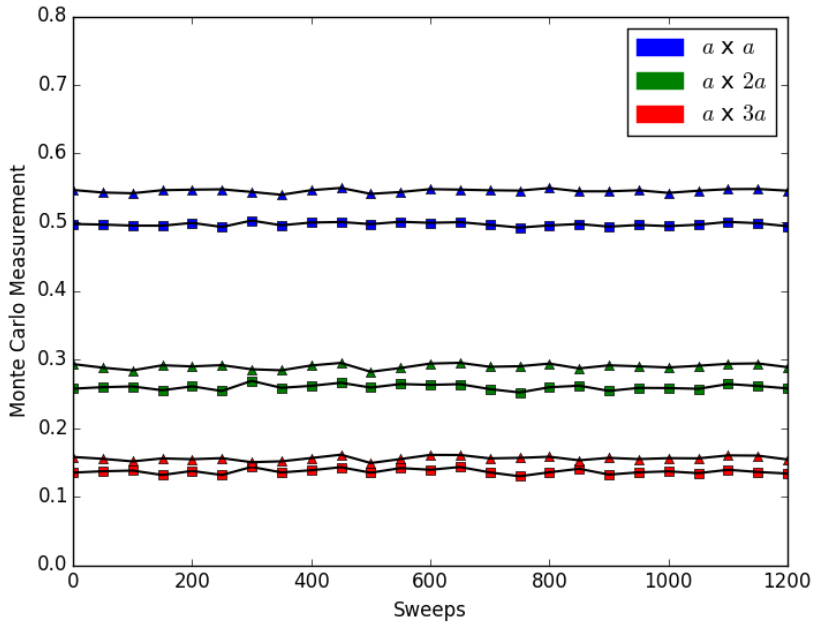

Figure 20: Monte Carlo measurements for $a \times a$, $a \times 2a$ and $a \times 3a$ Wilson loops. The triangles represent the improved action from equation 54 while the squares represent the unimproved action from equation 48. Between each measurement, $N_{cor} = 50$ sweeps were omitted along with 100 sweeps after the lattice was initialized for it to reach equilibrium.

## 6 Conclusion and Remarks

In this paper, we explored the use of Monte Carlo techniques on physical models. I used the Python language because of my previous experiences with it and its convenience especially with mathematical operations, multidimensional arrays and useful libraries such as `numpy` and `matplotlib`. Going forward, implementing these models in a language such as C would be an advantageous endeavor since the flaw of Python is its inefficiency especially compared to C and C-like languages.

Also, the use of a Message Passing Interface (MPI) and/or multithreading would also help with the time inconveniences faced with lattice QCD models. And with the lack of time, more significant measurements and more accurate lattice QCD models were not able to be explored as was hoped. As such, this may be a future project to expand on what I have learned thus far in lattice QCD.

That being said, we have seen the flexibility and power that some Monte Carlo methods have. The use of the Metropolis algorithm, assuming ergodicity and detailed balance, is extremely powerful because of its ability to converge on a desired equilibrium. And, due to taking a large sample size, possibly complex expectation values and measurements can be accurately approximated if simplified to a mean. As is shown in [4], for some functional $\Gamma[x]$ such that

$$\langle \Gamma[x] \rangle = \frac{\int \mathcal{D}[x(t)] \Gamma[x] e^{-S[x]}}{\int \mathcal{D}[x(t)] e^{-S[x]}}, \tag{59}$$

we can instead approximate it by

$$\langle \Gamma[x] \rangle \approx \overline{\Gamma} \equiv \frac{1}{N} \sum_{i=1}^{N} \Gamma[x^{(\alpha)}] \tag{60}$$

if the probability for some path $x^{(\alpha)}$ is equal to the weight of the expectation value (in this case $e^{-S[x]}$. The above can be very easily computed in a program and the error comes just from the sample size $N$.

Also, this paper has yielded some satisfying results. The expectation of the magnetization and susceptibility in the Ising model was modeled successfully, as with the excitation energy $E_1 - E_0$ of the harmonic oscillator. The latter can be solved by the Schrödinger equation but it was satisfying to see the same result through the use of Feynman path integrals and the Metropolis algorithm, a completely different approach. Lastly, three different Wilson loops, $a \times a$, $a \times 2a$ and $a \times 3a$, were calculated using an improved Wilson action. Nevertheless, such a measurement is just the beginning; measurements such as a static quark, anti-quark potential and many others could next be calculated.

Important portions of the code I used in this paper have been are shown but the entirety of every program is too lengthy and cumbersome to add to this paper. So, all of the code used in this paper can be found on GitHub at:

`http://github.com/crumpstrr33/Monte-Carlo-Applications-and-Lattice-QCD`.

## References

[1] D. W. Heermann and K. Binder. *Monte Carlo Simulation in Statistical Physics*. Springer-Verlag Berlin Heidelberg, 2010.

[2] A. Hellander. Stochastic simulation and monte carlo methods. *Available in: http://www. it. uu. se/edu/course/homepage/bervet2/MCkompendium/mc. pdf*, 2009.

[3] D. P. Landau and K. Binder. *A guide to Monte Carlo simulations in statistical physics*. Cambridge university press, 2014.

[4] G. P. Lepage. Lattice QCD for novices. In *Strong interactions at low and intermediate energies. Proceedings*, pages 49–90, 1998.

[5] C. Morningstar. The monte carlo method in quantum field theory. *arXiv preprint hep-lat/0702020*, 2007.

[6] J. J. Sakurai and J. Napolitano. *Modern quantum mechanics*. Addison-Wesley, 2011.

[7] J.-C. Walter and G. Barkema. An introduction to monte carlo methods. *Physica A: Statistical Mechanics and its Applications*, 418(C):78–87, 2015.