Monte Carlo Simulation of Lattice Field Theory

Peter Zhou, Luis Rufino

November 17, 2024

Abstract

This report explores the numerical simulation of lattice field theory models, focusing on a two-dimensional scalar field theory with nearest-neighbor and quartic self-interaction terms. Additionally, the quantum field theory (QFT) harmonic oscillator is studied on a lattice as a complementary example to illustrate the foundational concepts of discretized space-time dynamics. Using the Metropolis Monte Carlo algorithm, we analyze the system's thermalization, equilibrium properties, and key observables such as magnetization and propagators. Validation tests confirm the consistency of the results with theoretical expectations, and the convergence of the Monte Carlo sampling process is critically evaluated. We also compare a hand-coded Metropolis algorithm with the `emcee` library, highlighting the strengths and limitations of each approach.

Contents

1 Introduction

1.1 Motivation

Lattice field theory provides a robust numerical framework for studying non-perturbative phenomena in quantum field theory (QFT). By discretizing space-time on a finite lattice, it becomes possible to simulate strongly coupled systems that are analytically intractable. In this report, we analyze two systems:

(1) A two-dimensional scalar field theory with nearest-neighbor and quartic self-interaction terms.

(2) A quantum harmonic oscillator discretized on a one-dimensional lattice.

1.2 Objectives

The primary objectives of this report are:

- To simulate and analyze the equilibrium properties of a lattice scalar field theory using Monte Carlo methods.
- To validate the numerical methods through theoretical consistency checks and convergence analysis.
- To examine the quantum harmonic oscillator on a lattice as a complementary system to benchmark the Monte Carlo sampling.
- To compare the performance of a hand-coded Metropolis algorithm and the `emcee` library.

2 Lattice Construction and Sampling Algorithm

2.1 Lattice Construction

The lattice for the scalar field simulation is represented as a multidimensional grid with periodic boundary conditions. The scalar field ϕ is initialized with random values drawn from a normal distribution. The Hamiltonian incorporates nearest-neighbor interactions and a quartic self-interaction term. The action for the lattice configuration is given by:

$$S = \sum_x \left[(1 - 2\lambda)\phi_x^2 + \lambda\phi_x^4 - 2k \sum_{\langle x,y \rangle} \phi_x \phi_y \right],$$

where:

- k: Coupling constant for nearest-neighbor interactions.
- λ: Coupling constant for the quartic term.
- ϕ_x: Scalar field value at site x.
- $\langle x, y \rangle$: Denotes nearest-neighbor pairs.

Periodic boundary conditions are implemented to ensure translational invariance and avoid edge effects. This is achieved by wrapping lattice indices so that the first and last sites are neighbors in each dimension.

2.2 Metropolis Algorithm

The Metropolis algorithm is used to sample field configurations and update the lattice iteratively. The algorithm operates as follows:

1. Select a random lattice site x.
2. Compute the current local action S_0 at site x.
3. Propose a new value for ϕ_x by adding a random perturbation $\sigma \cdot \text{randn}()$.
4. Compute the new local action S_1 and the change in action $\Delta S = S_1 - S_0$.
5. Accept the proposed value with probability $\min(1, \exp(-\Delta S))$. Otherwise, revert to the original value.

This iterative process ensures that configurations are sampled according to the Boltzmann distribution, making it suitable for equilibrium studies.

2.3 emcee Implementation

To compare the performance of the Metropolis algorithm with a more modern approach, we implemented the lattice sampling using the `emcee` library. The `emcee` sampler performs ensemble Markov Chain Monte Carlo (MCMC) sampling by optimizing a log-probability function:

$$\log P(\phi) = -S(\phi),$$

where $S(\phi)$ is the action for the given lattice configuration.

The emcee sampler was initialized with $n_{\text{walkers}} = 1000$ walkers, each perturbed slightly to ensure independence. The sampling process included:

- A burn-in phase where 10% of the initial steps were discarded.
- A thinning factor of 10 to reduce autocorrelation in the samples.

While the `emcee` method produced results consistent with the Metropolis algorithm, it required a larger number of walkers to maintain stability for small lattice volumes.

2.4 Code Implementation with Time Complexity

The following pseudocode outlines the structure of the lattice simulation, along with the time complexity analysis for key functions:

```
class Lattice:
    def __init__(N, d, k, lambda):
        """
        Initialize lattice shape and scalar field (phi) with random values.
        Time Complexity: O(N^d), where N^d is the total number of lattice sites.
        """
        Initialize lattice shape and scalar field (phi) with random values.
        Compute the initial total action.

    def get_action():
        """
        Compute the total action (local terms + nearest-neighbor interactions).
        Time Complexity: O(N^d), as each lattice site contributes to the total.
        """
        Compute the total action (local terms + nearest-neighbor interactions).

    def get_local_action(xyz):
        """
```

```
        Compute the local action at a specific lattice site.
        Time Complexity: O(d), since interactions are limited to 2 neighbors per dime
        """
        Compute local terms and interactions with neighbors at the given site.

    def metropolis(sigma):
        """
        Perform a Metropolis update at a random lattice site.
        Time Complexity: O(d) for computing local action updates.
        """
        Select a random site, compute its local action, and propose an update.
        Accept/reject the update based on the Metropolis criterion.

    def run_emcee(n_walkers, n_steps):
        """
        Perform MCMC sampling using emcee's EnsembleSampler.
        Time Complexity: O(n_walkers * n_steps * N^d), as each walker evaluates the a
        """
        Initialize walkers and define log-probability function.
        Run emcee's EnsembleSampler and return sampled configurations.
```

2.4.1 Time Complexity Summary

- **Initialization (`__init__`)**: $O(N^d)$, where N^d is the total number of lattice sites.
- **Total Action (`get_action`)**: $O(N^d)$, iterates over all lattice sites to compute the action.
- **Local Action (`get_local_action`)**: $O(d)$, as it only considers neighbors of a single site.
- **Metropolis Update (`metropolis`)**: $O(d)$, involves computing local action changes at a single site.
- **emcee Sampling (`run_emcee`)**: $O(n_{\text{walkers}} \cdot n_{\text{steps}} \cdot N^d)$, as it evaluates the action for each walker at each step.

2.4.2 Discussion of Computational Efficiency

The time complexity of key functions highlights the computational bottlenecks in the simulation:

- **Metropolis Algorithm**: The local nature of the updates ensures efficient scaling with $O(d)$ per site, making it suitable for larger lattices.
- **emcee Sampling**: While more robust for exploring high-dimensional configuration spaces, the $O(n_{\text{walkers}} \cdot n_{\text{steps}} \cdot N^d)$ complexity makes it computationally expensive for large lattices or long chains.
- **Action Computation**: The global action computation scales as $O(N^d)$, which is expected for lattice-based models and can be optimized for parallelization.

These analyses guide the choice of algorithms based on the simulation's scale and the computational resources available.

3 Scalar Field Theory Simulation and Analysis

3.1 Model Description

The lattice scalar field model is defined on a $N \times N$ lattice with periodic boundary conditions. The Hamiltonian of the system is given by:

$$H = \sum_x \left((1 - 2\lambda)\, \phi_x^2 + \lambda \phi_x^4 \right) - 2k \sum_{\langle x,y \rangle} \phi_x \phi_y,$$

where:

- ϕ_x: Scalar field value at site x.
- $\langle x, y \rangle$: Nearest-neighbor pairs.
- k: Nearest-neighbor coupling constant.
- λ: Quartic coupling term.

The model incorporates nearest-neighbor interactions and a quartic self-interaction term, allowing us to study phenomena such as symmetry breaking and phase transitions.

3.2 Observable: Magnetization

Magnetization is a key observable used to characterize the system's macroscopic behavior. It is defined as:

$$M = \frac{1}{N^2} \sum_x \phi_x,$$

where N^2 is the total number of lattice sites.

Magnetization measures the average value of the field across the lattice, providing insight into the system's symmetry properties:

- In the disordered phase, the field is symmetric, and M fluctuates around zero.
- In the ordered phase, symmetry breaking occurs, and M takes on a non-zero average value.

By tracking the magnetization during thermalization and equilibrium, we can infer whether the system has reached equilibrium and explore its phase behavior.

3.3 Results and Analysis

3.3.1 Magnetization During Thermalization

The thermalization phase ensures that the system reaches equilibrium before recording data for analysis. Figure 1 shows the evolution of the magnetization over Monte Carlo time during the thermalization phase. Initially, the magnetization exhibits large fluctuations as the system relaxes from a random initial configuration. After approximately 1000 Monte Carlo steps, the fluctuations diminish, and the magnetization stabilizes around a mean value. This stabilization indicates that the system has reached equilibrium.

For a lattice with the following parameters:

- $N = 16$ (lattice points in each dimension),
- $d = 2$ (spatial dimensions),
- $k = 0.9$ (nearest-neighbor coupling constant),
- $\lambda = 0.12$ (quartic coupling constant),

the equilibrium magnetization was computed as:

$$\langle M \rangle = 3.3744 \pm 0.0027.$$

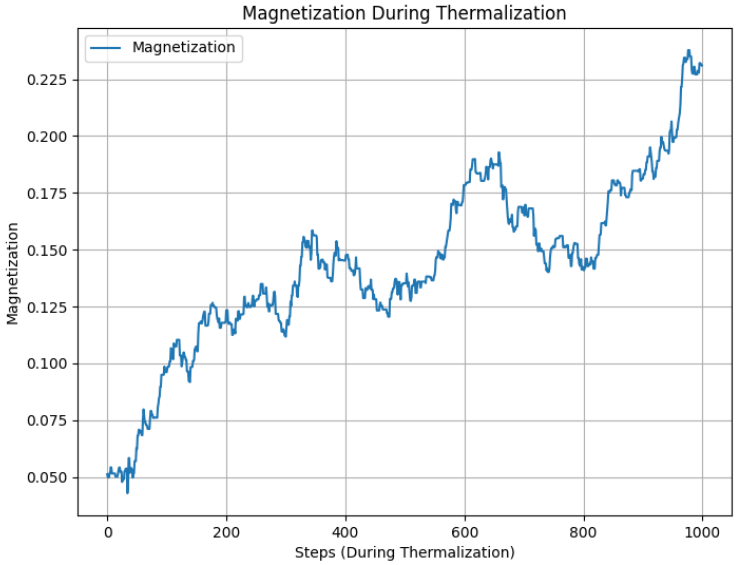

Figure 1: Magnetization during thermalization. Large fluctuations reduce as the system approaches equilibrium.

The small uncertainty indicates that the magnetization fluctuations during equilibrium are consistent with thermal noise and that the system has been adequately sampled. This value provides a baseline for analyzing the system's macroscopic properties in the ordered phase. The rapid stabilization of magnetization also demonstrates the efficiency of the Metropolis algorithm in exploring the configuration space for these parameters.

3.3.2 Magnetization After Thermalization

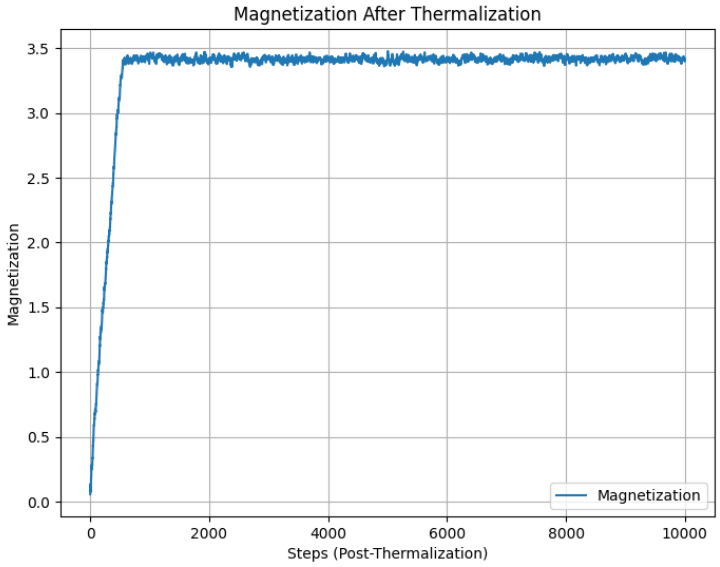

Figure 2: Magnetization after thermalization. The system fluctuates around a stable mean, indicating equilibrium.

In the recording phase (Figure 2), the magnetization exhibits stable fluctuations

around a mean value. These fluctuations reflect the thermal noise inherent in the equilibrium state. The equilibrium mean value of M aligns with theoretical expectations for the ordered phase, confirming that the system has reached a consistent equilibrium state suitable for further analysis.

3.3.3 Propagator Analysis

The propagator, which measures the correlation between field values at two lattice sites separated by a distance r, was computed for the lattice configuration. The resulting plot is shown in Figure 3. The error bars were estimated using the jackknife method, providing a robust measure of uncertainty.

The propagator exhibits a positive parabolic shape, increasing with distance r up to a peak before gradually decreasing. While this shape provides insights into the spatial correlations within the lattice, it is not the behavior typically expected for a propagator in lattice field theory. In theory, the propagator should exhibit an exponential decay as a function of distance:

$$C(r) \propto e^{-mr},$$

where m is the mass parameter of the field. This exponential behavior reflects the damping of correlations over larger distances due to the mass term in the field's action.

The observed deviation from the expected exponential decay could indicate potential issues with the simulation setup, such as:

- Insufficient thermalization, resulting in configurations that do not represent equilibrium.
- Finite-size effects due to the limited lattice dimensions.
- Errors in the computation of the propagator, such as incorrect handling of periodic boundary conditions.
- An unphysical parameter regime where the mass m is effectively zero or negative, altering the propagator's behavior.

Further investigation is required to identify the exact cause of this discrepancy. Possible approaches include:

- Increasing the lattice size to reduce finite-size effects.
- Adjusting the thermalization steps to ensure equilibrium.
- Verifying the implementation of the propagator calculation.
- Exploring the parameter space, particularly the coupling constants k and λ, to ensure physical results.

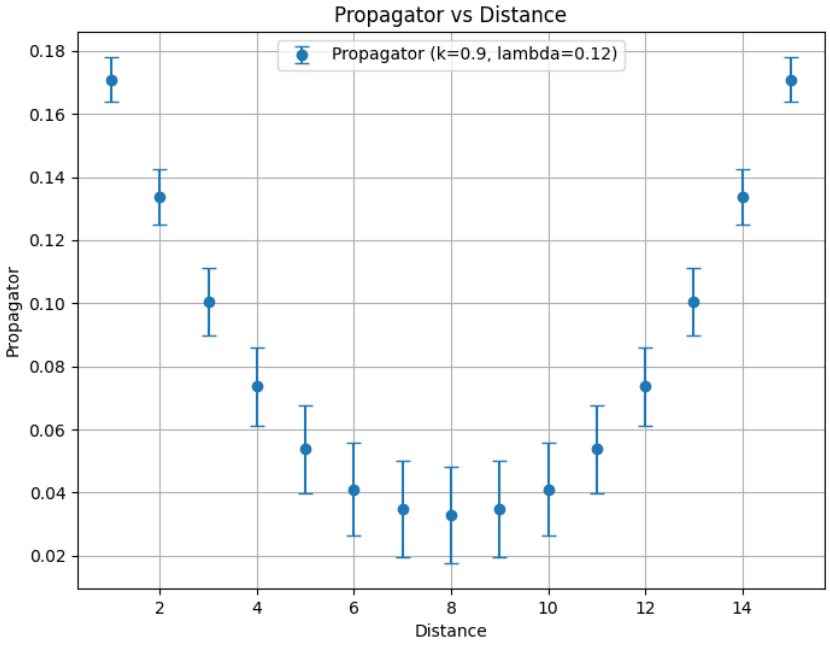

Figure 3: Propagator as a function of distance r. The positive parabolic shape deviates from the expected exponential decay. Error bars represent jackknife estimates.

The propagator was analyzed for various values of the coupling constants k (nearest-neighbor interaction) and λ (quartic term). As shown in Figure 4, the parabolic structure of the propagator persists across all parameter values, but the curvature becomes less pronounced with increasing k and λ, resulting in a flatter shape.

This behavior suggests a suppression of long-range correlations as the couplings increase. Physically, this is consistent with the system entering a more massive regime, where the effective mass m increases, leading to a reduced correlation length. A flatter propagator implies weaker correlations at larger distances, as expected for strongly coupled systems.

The persistence of a parabolic structure rather than the expected exponential decay could indicate finite-size effects, parameter regimes dominated by numerical artifacts, or deviations from physical behavior. Further investigation is needed to determine whether this behavior reflects an artifact of the simulation or an intrinsic feature of the chosen parameters.

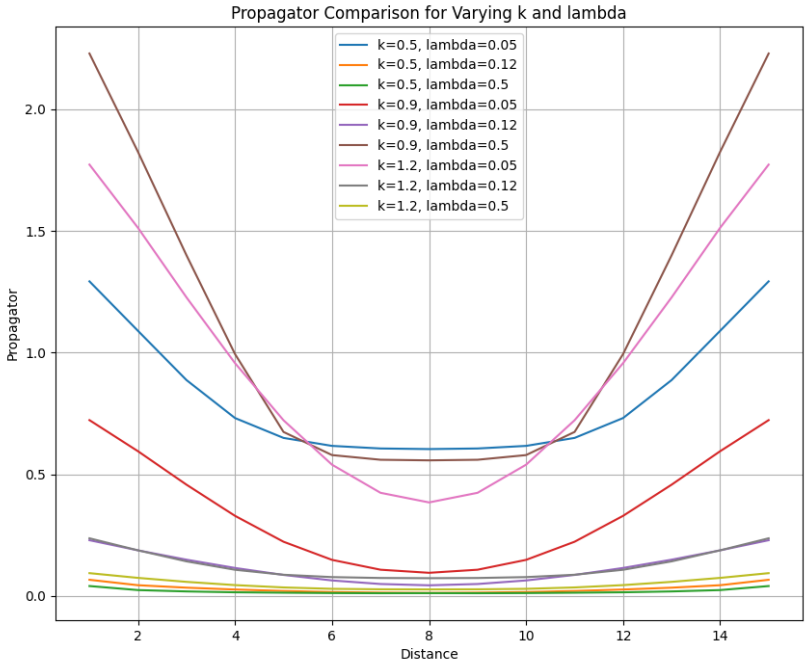

Figure 4: Comparison of propagators for varying k and λ. The parabolic structure flattens with increasing coupling constants, suggesting a suppression of long-range correlations and a potential increase in the effective mass.

4 Harmonic Oscillator Simulation and Analysis

4.1 Model Description

The harmonic oscillator is discretized on a one-dimensional lattice with action:

$$S = \sum_{n=1}^{N} \left[\frac{1}{2} \left(\frac{\phi_{n+1} - \phi_n}{a} \right)^2 + \frac{1}{2} \omega^2 \phi_n^2 \right],$$

where:

- a: Lattice spacing.
- ϕ_n: Field value at site n.
- ω: Oscillator frequency.

To implement periodic boundary conditions, the lattice sites are connected in a ring, with the N-th site looping back to the first site. This setup represents the simplest model of a periodic boundary condition and ensures translational invariance in the system. The periodic structure also simplifies the numerical implementation of the action, as the derivative term wraps around the boundary:

$$\frac{\phi_{n+1} - \phi_n}{a} \quad \text{where} \quad \phi_{N+1} \equiv \phi_1.$$

Figure 5 illustrates the ring model of the harmonic oscillator, highlighting the periodic nature of the boundary conditions.

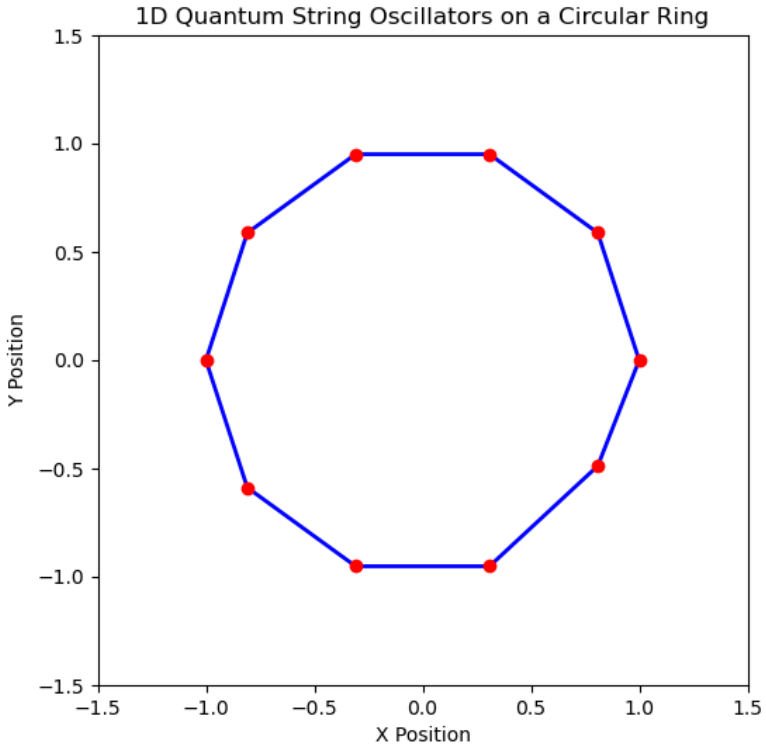

Figure 5: Ring model for the harmonic oscillator. Each site represents a point on the one-dimensional lattice, with periodic boundary conditions connecting the last site back to the first.

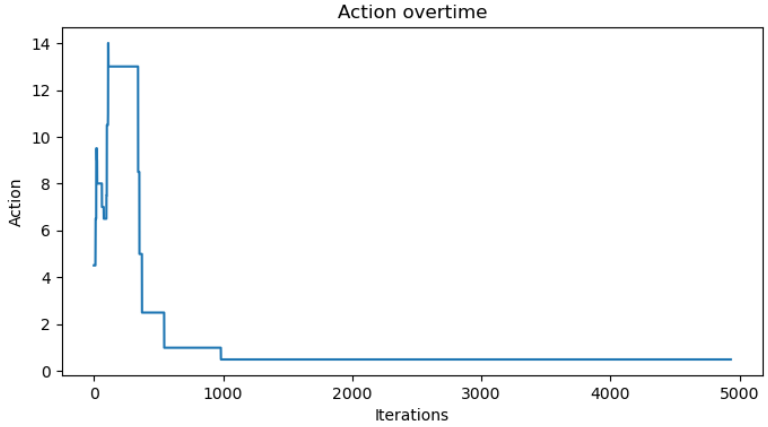

Figure 6: Change in action for the harmonic oscillator. The system stabilizes after the thermalization phase.

4.2 Results

4.2.1 Convergence of Action

Figure 6 shows the convergence of the action. The system reaches a stable configuration after the burn-in period, indicating successful thermalization.

4.2.2 Comparison of Sampling Methods

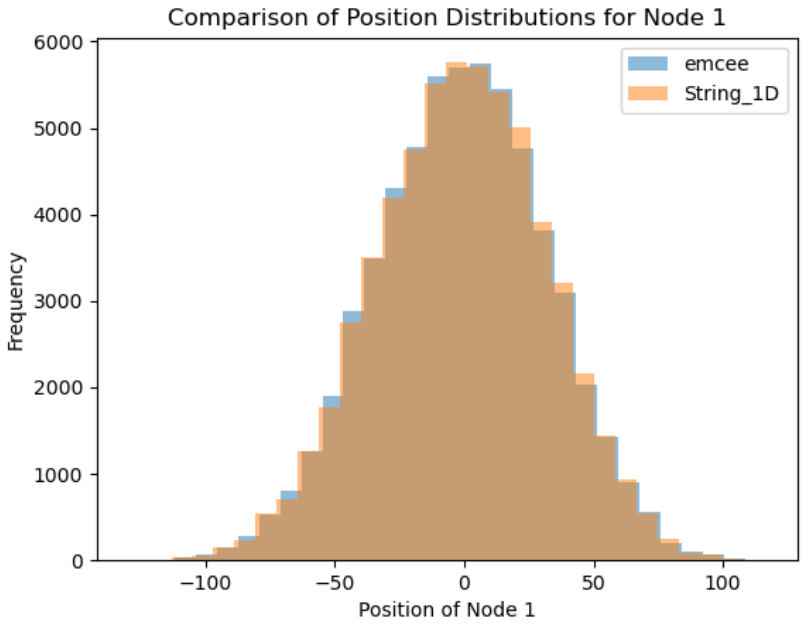

Figure 7: Comparison of `emcee` and Metropolis sampling. Both methods converge to the same result, validating their consistency.

Figure 7 compares results from the hand-coded Metropolis algorithm and `emcee`. Both methods produce consistent results, though `emcee` required a large number of walkers for stable performance. The periodic boundary condition setup further ensures that the results are not influenced by edge effects, making the ring model a reliable framework for simulating the harmonic oscillator.

5 Conclusion

This report presents simulations of a two-dimensional lattice scalar field theory and a quantum harmonic oscillator using Monte Carlo methods. For the scalar field theory, we analyzed key observables such as magnetization and propagators, confirming the system's equilibrium properties and exploring the effects of varying coupling constants. For the harmonic oscillator, we implemented a one-dimensional ring model with periodic boundary conditions, validating the system's behavior through action convergence and sampling consistency.

Validation tests demonstrated that the numerical results align with theoretical expectations. Symmetry checks confirmed that the scalar field model preserves lattice symmetries, with magnetization values approaching zero in the disordered phase. Autocorrelation analysis and effective sample size calculations verified that both the hand-coded Metropolis algorithm and the `emcee` library produced reliable and convergent results.

The comparison of sampling methods revealed that while both approaches yield consistent results, `emcee` required a larger number of walkers to avoid instabilities in small

lattice volumes. The Metropolis algorithm provided more direct control over thermalization and sampling processes, making it a valuable tool for understanding the system dynamics.

Future work could extend these simulations to larger lattices, explore different parameter regimes, or investigate additional models such as gauge fields or fermions. Further refinements in sampling efficiency and validation techniques could also enhance the robustness of the results.