Revision 0.1 **October 29, 2020**

QUANTUM HARMONIC OSCILLATOR ON A COMPUTER

S. Porwal

Senior Honours Project Semester 1

0.1

Abstract

1 Introduction

- Overview of background
- Brief introduction to the methods as previously used
- Objective (Include link (Sahaj (2020)))

2 Method

- Overview, Definitions and Parameters
- Monte Carlo Markov Chain
- Metropolis

2.1 OVERVIEW,DEFINITIONS,PARAMETERS

rather brief

The simulation(Sahaj (2020)) is implemented using a Monte Carlo Markov Chain(Hanada (2018)) method. The Monte Carlo method is a time integration method which uses random numbers to evaluate definite integrals. A Markov chain is a statistical tool used to model transitions of a system between its states using probabilistic rules.

In this project, a 1-dimensional lattice of coupled Quantum Harmonic Oscillators are set up, and the transitions of each of the points in this lattice are simulated using the Metropolis Algorithm(described in subsection 2.2).

To define the process, certain parameters need to be defined. These are listed in Table 1

Some Physics Quantities which need to be defined are:

The Action: The quantum action of a system is a property which measures the energy of a state of a system. In the code used for this project, the action is defined as in Westbroek et al. (2018) as:

$$S = \sum_{i=1}^{N_s} \left[\frac{1}{2}(x_{i+1} - x_i)^2 + \frac{1}{2}\omega^2 x_i^2\right] \quad (1)$$

N_s ?

Note that only the nearest neighbours have any effect on the change in action for a particular point in the lattice. So this equation was simplified and the change of action for point k on the lattice for a transition from x_i to x'_i was determined to be

what is x_{k+1} ?

k ?

$$\delta S = \sum_{i=k-1}^{k+1} \frac{1}{2}[(x'_{i+1} - x'_i)^2 - (x_{i+1} - x_i)^2 + \omega^2(x_i'^2 - x_i^2)] \quad (2)$$

Don't need this until have introduced MC method

The general flow of the simulation is:

cold start also hot start ?

1. Initialise all positions in the lattice to be at position 0.0
2. Initialise list of positions(to store all positions)
3. Initialise sum of positions(to calculate mean)
4. Initialise sum of squared positions(to calculate variance)
5. Loop over the number of time steps:
   (a) Append the current position to the list of positions
   (b) Add the current position to sum of positions
   (c) Add current position squared to the sum of squared positions
   (d) Loop over the lattice points in a randomised order.
      i. Calculate the next position for the current lattice point using the Metropolis Algorithm and the action defined above.
      ii. Update the positions if Metropolis condition is satisfied
   (e) Update the time step
6. Determine the Mean and Variance from the sum of positions and sum of squared positions.
7. Code for visualisation

(auto) Correlation fns?

Errors?

2.2 METROPOLIS METHOD

The motion of the particle was implemented using a Metropolis algorithm(as described in Metropolis et al. (1953)). The purpose of this algorithm is to produce a path for the particle using statistical methods. The process of the algorithm goes as follows:

1. For a max step size of dx, randomly select an increment for the position $\delta x \in (-dx, dx)$
2. Propose a new position $x' = x + \delta x$
3. Calculate the change in the action $\Delta S = S(x') - S(x)$ $\leftarrow$ diff here
4. Calculate the metropolis factor $M = e^{-\Delta S}$
5. Generate a uniform random number $r \in (0, 1)$
6. Conditionally update the current x value to the new proposed position from Step 2. The condition for this update is $r < M$. If the condition is not met then the next position is the same as ~~the~~ current position
7. This process is repeated for each successive step for each lattice point.

In the python code(Sahaj (2020)) this algorithm was implemented using a separate Metropolis class, which contains two components: a constructor to set values for the parameters needed, and a 'step' function to implement a single step in the Metropolis algorithm.

Parameter	Description

Table 1: Parameters for the simulation

3 Results & Discussion

$n_s = 1000$?

The method described above was run for the following parameters:

- Number of time steps: 100,000
- Number of sites on the lattice: 5 $= n_s$
- Max step-size: 0.5
- ω: 1.0

Why so small ? Finite size effects ?

The following graphs were produced from the data resulting from these parameters:

- Representative path for motion of one lattice point (Figure 1)
- Histogram of positions comparison with expected Gaussian distribution of the positions (Figure 2)
- Autocorrelation function for the positions x vs the time difference between positions (Figures 3, 4 and 5)

Some values of measurements from the simulation are

- The acceptance rate for these parameters was determined to be approx. 83%.
- The mean position(averaged over the 5 sites) was approx. 0.01
- The mean squared position(averaged over sites) was approx. 0.45

The theoretical values for the mean values are:

- Mean position = 0.0
- Mean squared position(given by Westbroek et al. (2018), Eq 42) = approx. 0.46 ← proof

The calculated values show a high agreement with the theoretical values. This agreement increases as the number of sites or the number of lattice points is increased.

The average motion of lattice points (1) shows an oscillation about the mean position(approx. 0.0) and a maximum deviation of approx. 1.0 for the last 500 steps. This is consistent with expectations. which are ?

? The positions histogram shows a Gaussian distribution of positions, however the distribution does not exactly match the expected Gaussian. This needs to be investigated by altering the parameters. ?

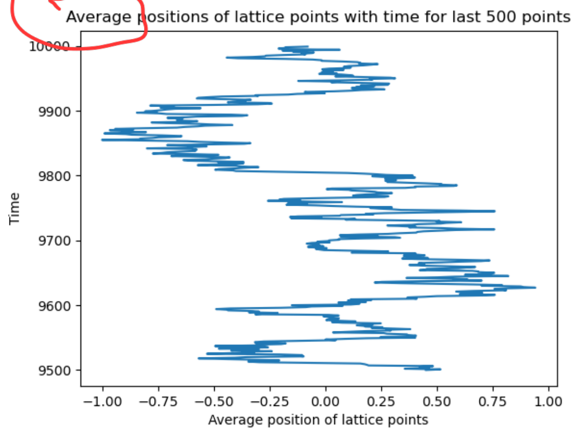
The auto-correlation function for all time differences shows a rapid decay to zero after which point it shows a random oscillation about zero(this is the noise), as is expected. The function for the first 100 steps shows how fast the auto-correlation function decays(as an exponential). The log auto-correlation function for the first number of sites 50 time differences, has a negative slope of approx. 0.95, which is very close to the value of dimensionless parameter $(\omega = 1.0)$, as is expected as it is a representative measure of the Energy(Hamiltonian) eigenvalue. It can be seen that the calculated log auto-correlation matches the theoretical value very closely.

→ Surely this is from the correlation fn not autocorrelation fn ?

3.1 RESULTS AND FIGURES

3.2 DISCUSSION

?



10,000 sweeps not 100,000?

Figure 1: Representative Average Path of Particles for last 500 steps

?

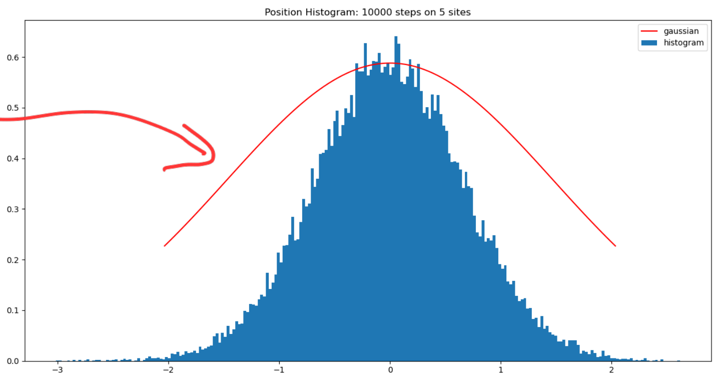


Figure 2: Position Histogram

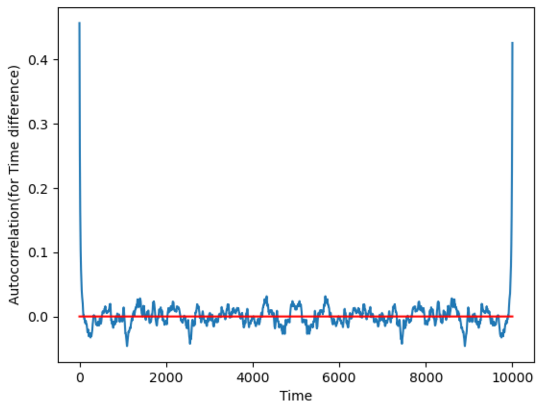


Figure 3: Auto-correlation of x for all time differences

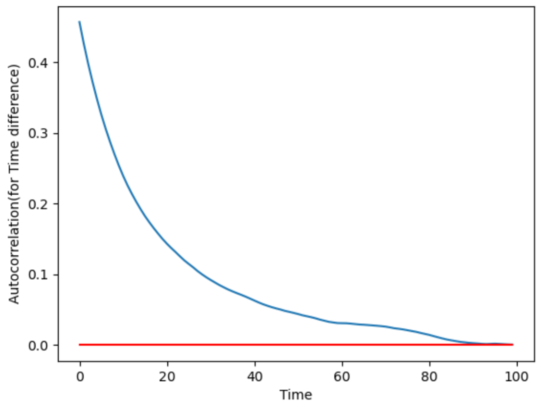


Figure 4: Auto-correlation of x for First 100 steps

Why does the theoretical have
• fit the calculated does not

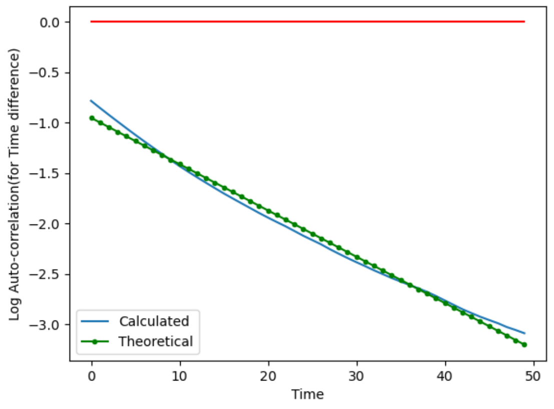


Figure 5: Log auto-correlation of x vs Time difference for first nsites = 5 timesteps

What does this mean?

4 Conclusions

Discussion computation
of errors

References

M. Hanada. Markov Chain Monte Carlo for Dummies. *arXiv e-prints*, art. arXiv:1808.08490, Aug. 2018.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. , 21(6):1087–1092, June 1953. doi: 10.1063/1.1699114.

P. Sahaj. Quantum harmonic oscillator. `https://github.com/sporwal98/SHP_QuantumHarmonicOscillator`, 2020.

M. J. E. Westbroek, P. R. King, D. D. Vvedensky, and S. Dürr. User's guide to Monte Carlo methods for evaluating path integrals. *American Journal of Physics*, 86(4):293–304, Apr. 2018. doi: 10.1119/1.5024926.