A Comparison Between Molecular Dynamics and Monte Carlo Simulations of a Lennard-Jones Fluid

CBE 641: Nanoscale Transport
Project Report

Kaushik Shankar

May 1, 2019

Contents

List of Tables

List of Figures

Abstract

Molecular dynamics and Metropolis Monte Carlo simulations are used to study the behavior of a Lennard-Jones fluid. The relative sampling efficiencies of the two methods are compared. Both the methods give identical equilibrium properties, but one method is better than the other for different system parameters. Molecular dynamics simulations, in addition to predicting thermodynamic properties accurately, also provide reliable estimates of dynamic properties like diffusivity. Monte Carlo is also able to predict dynamics, but only when move sizes are small.

1 Introduction

Molecular dynamics (MD) and Monte Carlo (MC) methods are widely used in computer simulations to study many-particle systems. The MC method samples the configuration space and calculates the properties of the system by randomly sampling microstates of the system. On the other hand, MD uses Newton's equations of motion to dynamically predict the motion of particles and generate configurations. Any computer simulation requires a model for interaction between the particles. The Lennard-Jones model serves as a great starting point to understand the physical properties of fluids.

In this project, the focus is on the comparison between MD and MC simulations of a Lennard-Jones fluid. Comparisons are made with respect to the prediction of dynamic and equilibrium properties, and the efficiency of the two methods.

The rest of this report is organized as follows: in the next section, the MC and MD techniques are discussed briefly. This is followed by a brief presentation of computational details of the simulations. The remaining sections comprise of a discussion on the simulation results.

2 Methodology

2.1 Molecular Dynamics Simulations

MD simulations essentially solve Newton's equations of motion numerically. The dynamics of each atom i is given by

$$m_i \frac{d^2 \vec{r_i}}{dt^2} = \vec{f_i} \tag{1}$$

where m_i and $\vec{r_i}$ represent the masses and positions of the particles, and $\vec{f_i}$ is the force that acts on each particle. This force is usually obtained as a gradient of a potential

energy function U given by

$$\vec{f_i} = -\vec{\nabla}_i U \tag{2}$$

The potential energy U depends on the set of positions of the particles. In the simplest scenario, the potential can be represented as a sum of pairwise interactions between the particles.

$$U = \sum_{i=1}^{N} \sum_{j>i}^{N} u(\vec{r_{ij}}) \tag{3}$$

where $\vec{r_{ij}} = \vec{r_i} - \vec{r_j}$, and N is the total number of particles. The forces acting on each particle in such a case is then given by

$$\vec{f_i} = \sum_{j \neq i}^{N} \vec{f_{ij}}, \quad \vec{f_{ij}} = -\frac{du(r_{ij})}{dr_{ij}} \frac{\vec{r_{ij}}}{r_{ij}} \tag{4}$$

2.2 The Lennard-Jones Potential

The Lennard-Jones potential is a pair potential given by

$$u_{LJ}(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] \tag{5}$$

The parameter ϵ determines the strength of the interaction and the parameter σ defines a length scale. The attractive r^{-6} term dominates at large distances and models the van der Waals interaction potentials. The repulsive r^{-12} term dominates at short distances and accounts for the repulsive forces arising from overlap of atomic orbitals. The functional form of the repulsive term is arbitrary.

The inter-particle forces arising from the Lennard-Jones potential have the form

$$\vec{f_{ij}} = \frac{24\epsilon}{r_{ij}^2} \left[2 \left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] \vec{r_{ij}} \tag{6}$$

2.3 Metropolis Monte Carlo

In a Metropolis Monte Carlo simulation, the system is simulated by sampling its transition across its accessible microstates such that the microstates are sampled according to their equilibrium distribution. In the canonical (NVT) ensemble, this is the Boltzmann distribution. At each MC step a new configuration is generated. This consists of making a random change to the coordinates of a randomly chosen atom.

After the move the energy of the new configuration, U_{new} is calculated and compared to the old energy, U_{old}. If $U_{new} < U_{old}$, then the move is accepted. If $U_{new} > U_{old}$, the move is accepted with a probability given by

$$P_{accept} = \exp\left[\frac{-(U_{new} - U_{old})}{k_B T}\right] \tag{7}$$

2.4 Constant Temperature MD

Molecular dynamics methods involve solving Newton's equations of motion. When subject to conservative forces that depend only on position, they yield constant energy dynamics. In order to compare the results on a correct basis, the simulations of MD need to be performed in the canonical ensemble. In order for a system to transition from a constant energy dynamics to constant temperature dynamics, it has to exchange energy with the surroundings. This is achieved by the introduction of a thermostat. In this project, the Nose-Hoover thermostat chain is used, as it accurately reproduces the canonical ensemble dynamics accurately and is ideal for estimating dynamic properties like diffusion coefficients [1].

2.4.1 Nose-Hoover Chains

The Nose-Hoover dynamics is generated from an extended Hamiltonian where one considers the total energy of the system and augments it with the energy of a reservoir bath (or thermostat). This adds an additional degree of freedom to the system, ξ_1. This degree of freedom has an associated *mass*, Q_1, which effectively determines the strength of the thermostat. The equations of motion obeyed by this additional degree of freedom guarantee that the original degrees of freedom sample the canonical ensemble. This degree of freedom is the terminus of a chain of similar degrees of freedom, each with their own mass. The modified equations of motion for the system with a chain of M thermostats can be represented as

$$\dot{\vec{v_i}} = -\frac{1}{m_i}\vec{\nabla}_i U - \xi_1 \vec{v_i} \tag{8}$$

$$\dot{\xi_1} = \frac{1}{Q_1}\left[\sum_{i=1}^{N} \frac{p_i^2}{m_i} - N_f k_B T\right] - \xi_1 \xi_2 \tag{9}$$

$$\dot{\xi_j} = \frac{1}{Q_j}[Q_{j-1}\xi_{j-1}^2 - k_B T] - \xi_j \xi_{j+1} \tag{10}$$

$$\dot{\xi_M} = \frac{1}{Q_M}[Q_{M-1}\xi_{M-1}^2 - k_B T] \tag{11}$$

where ξ_i are the thermostat momenta, Q_i are the thermostat masses, and N_f is the number of degrees of freedom.

2.5 The Ergodic Hypothesis

The ergodic hypothesis states that the time averages and ensemble averages of properties of a system are equivalent at sufficiently large times. This assures the equivalence of time averages (typical of MD simulations) and ensemble averages (typical of MC simulations), and both the methods are therefore expected to give identical estimates of equilibrium properties of a system.

2.6 Estimation of Thermodynamic and Physical Properties

2.6.1 Potential and Kinetic Energy

The potential energy $\langle U \rangle$ is computed by taking either the time or ensemble average of U as computed from Equation 3. The constraint of constant temperature T sets the kinetic energy from the equipartition theorem

$$\langle K \rangle = \frac{dNk_BT}{2} \tag{12}$$

where d is the dimensionality of the system.

2.6.2 Pressure

Pressure can be estimated using the virial formula. For a pair potential, the expression is

$$PV = Nk_BT + \frac{1}{3}\left\langle \sum_{i<j}^{N} \vec{r_{ij}} \cdot \vec{f_{ij}} \right\rangle \tag{13}$$

where P is the pressure and V is the volume.

2.6.3 Radial Distribution Function $g(r)$

The radial distribution function gives the mean organization measured around an atom, taking the ideal gas as reference. It is useful in analyzing the structural properties displayed by the system. Essentially, it is the probability of finding a particle at a distance between r and $r + dr$, given that a particle is present at the origin. The radial distribution function is determined by calculating the distance between all particle pairs and binning them into a histogram. The histogram is then normalized with respect to an ideal gas, where particle histograms are completely uncorrelated.

The $g(r)$ obtained from the above procedure is for one configuration. In order to get the mean distribution of particles, an average over different time steps or different moves is taken.

2.6.4 Heat Capacity

It can be shown that the energy fluctuations in the canonical ensemble is related to the specific heat capacity per molecule C_v as

$$C_v = \frac{\langle U^2 \rangle - \langle U \rangle^2}{N k_B T^2} = \frac{var(U)}{N k_B T^2} \tag{14}$$

2.6.5 Self-diffusion Coefficient

The self-diffusion coefficient is related to the mean squared displacement (MSD) of a particle through Einstein's relation. The MSD becomes proportional to time t in the limit of large times. The proportionality constant that relates MSD to t is the self-diffusion coefficient D, and is given by

$$D = \lim_{t \to \infty} \frac{\langle |\vec{r}(t) - \vec{r}(0)|^2 \rangle}{2dt} \tag{15}$$

where d is the dimensionality of the system.

3 Computational Details

3.1 Units

For systems with the interaction among the atoms given by the Lennard-Jones potential, it is common to work with reduced units, where all physical properties are dimensionless. Basic definitions of physical quantities of interest are listed in Table 3.1.

3.2 Domain

All the simulations were carried out in the canonical (NVT) ensemble. The reduced temperatures T, reduced number density ρ and the number of particles N. The volume is then set by $V = N/\rho$. The domain is a cubic box. Although only 3D simulations are carried out, the code has been written so that switching to other dimensions can be achieved easily.

Table 3.1: System of reduced units used in simulation of Lennard-Jones particles

Physical Quantity	Unit
Length	σ
Energy	ϵ
Mass	m
Time	$\sigma\sqrt{m/\epsilon}$
Velocity	$\sqrt{\epsilon/m}$
Force	ϵ/σ
Pressure	ϵ/σ^3
Temperature	ϵ/k_B
Density	$1/\sigma^3$

3.2.1 Periodic Boundary Conditions

The behavior of finite systems is very different from that of infinite systems. Since infinite number of particles cannot be simulated, finite number of particles are used and periodic boundary conditions are imposed to minimize finite size effects. Using periodic boundary conditions implies that particles are enclosed in a box, which is replicated to infinity by rigid translation in all the three directions, completely filling the space. When a particle enters or leaves the simulation region, an image particle leaves or enters this region, such that the number of particles from the simulation region is always conserved. The surface effects are thus virtually eliminated and the position of the box boundaries plays no role.

3.2.2 Minimum Image Convention

As a result of using periodic boundary conditions, each particle in the simulation box appears to be interacting not only with the other particles in the box, but also with their images. Apparently, the number of interacting pairs increases enormously. This is overcome by using the minimum image convention. The minimum image convention considers only the closest image of the particle and neglects all the rest while computing interactions.

3.3 Initial Condition

For both the MC and MD simulations, the initial condition was to place the particles in a simple cubic lattice that spanned the entire domain. Adjacent particles are equidistant from each other. A snapshot of the initial condition is shown in Figure

3.1. The simulation domain shown here consists of 216 particles, placed on a uniform simple cubic lattice spanning the domain.

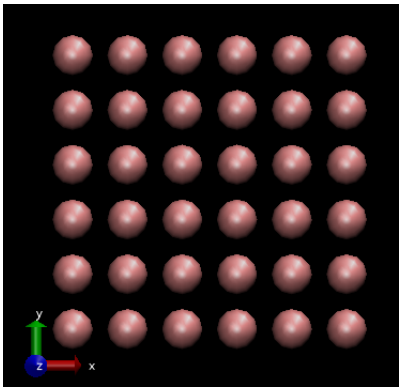


Figure 3.1: A snapshot of the initial condition

3.4 Time Integration in MD simulations

The equations of motion in MD require numerical integration. Several classes of MD integrators exist, but only the Velocity Verlet algorithm is considered here. This choice was made due to the ease of implementation and reasonable numerical stability. The integration algorithm is given by

$$\vec{v}(t + \Delta t/2) = \vec{v}(t) + \frac{\Delta t}{2m}\vec{f}(\vec{r}(t)) \tag{16}$$

$$\vec{r}(t + \Delta t) = \vec{r}(t) + \Delta t\ \vec{v}(t + \Delta t/2) \tag{17}$$

$$\vec{v}(t + \Delta t) = \vec{v}(t + \Delta t/2) + \frac{\Delta t}{2m}\vec{f}(\vec{r}(t + \Delta t)) \tag{18}$$

For all the MD simulations that have been carried out in this project, a time step of $\Delta t = 0.005$ in reduced units is used. This value has been chosen because it was numerically stable and the numerical error was insignificant.

3.5 MC Sweeps

For the MC calculations, there is no objective definition of time. The notion of time is replaced with *sweeps*. One MC sweep corresponds to N attempts of displacement of particle positions, where the Metropolis algorithm is employed to decide the acceptance of an atomic displacement. In each attempt, one randomly selected atom is

displaced in each Cartesian direction ϕ given by

$$r'_{i,\phi} = r_{i,\phi} + 2\delta(rand - 0.5) \tag{19}$$

where δ is a parameter that controls the maximum size of displacements and *rand* is a uniformly distributed random number between 0 and 1. The value of δ is a key ingredient of the MC simulation: if it is too large, there follows a high probability for the resulting configuration to have a high energy and thus the trial move has a large probability of being rejected. On the other hand, if δ is too small, the change in potential energy is also small and most trials will be accepted, but the configuration space would be sampled poorly. For all simulations (unless stated otherwise), the value of δ is adjusted after each sweep so as to get an acceptance ratio (number of accepted attempts over total number of attempts) of 0.5.

3.6 Implementation

All of the above algorithms were implemented using the commercial computing environment MATLAB. The routines that were used have been attached to the appendix as supplementary information.

4 Results and Discussion

4.1 Sampling Efficiency

In order to obtain reliable averages of thermodynamic quantities at equilibrium, good sampling has to be ensured. The energy autocorrelation function can be used to identify the number of successive time steps or sweeps after which one can get uncorrelated samples. A good sample must consider successive configurations separated by an interval sufficient to decorrelate them. The autocorrelation of the configurational potential energy is defined as

$$R_U(\tau) = \langle U(t)\ U(t+\tau) \rangle_t \tag{20}$$

where $U(t)$ is the potential energy at time t, and $U(t+\tau)$ is the potential energy at time $t+\tau$. In order to get a meaningful comparison of the sampling efficiency of MD and MC methods, t has to have units of machine time.

Simulations were run long enough to generate equilibrium configurations (MC simulations always reached equilibrium faster). The number of equilibrium steps required for a simulation were set based on the smoothness of the $g(r)$ plot that was obtained.

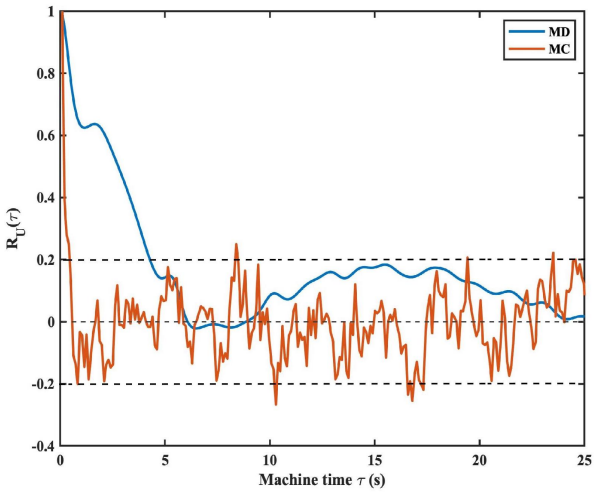


Figure 4.1: Energy autocorrelation for an ideal gas simulation. The parameters for the simulation are $T = 1.2, \rho = 0.001, N = 216$.

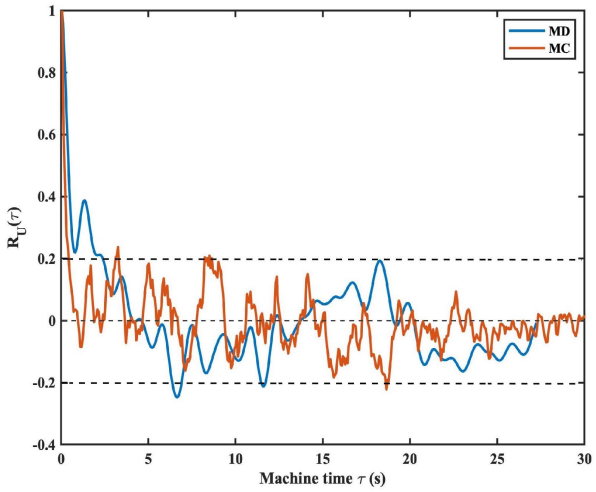


Figure 4.2: Energy autocorrelation for an ideal gas simulation. The parameters for the simulation are $T = 1.2, \rho = 0.05, N = 216$.

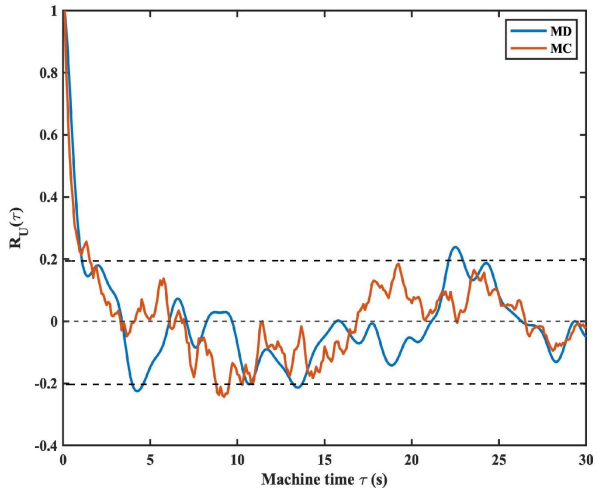


Figure 4.3: Energy autocorrelation for a liquid simulation. The parameters for the simulation are $T = 1.2, \rho = 0.8, N = 216$.

MD and MC methods are now compared based on the energy autocorrelation function. A full description of the system would involve the systematic study of its properties for all possible combinations of the fluids parameters. In order to simplify the analysis, the number of particles N is fixed at 216 and the reduced temperature T is fixed at 1.2. The reduced density ρ is chosen as the variable parameter.

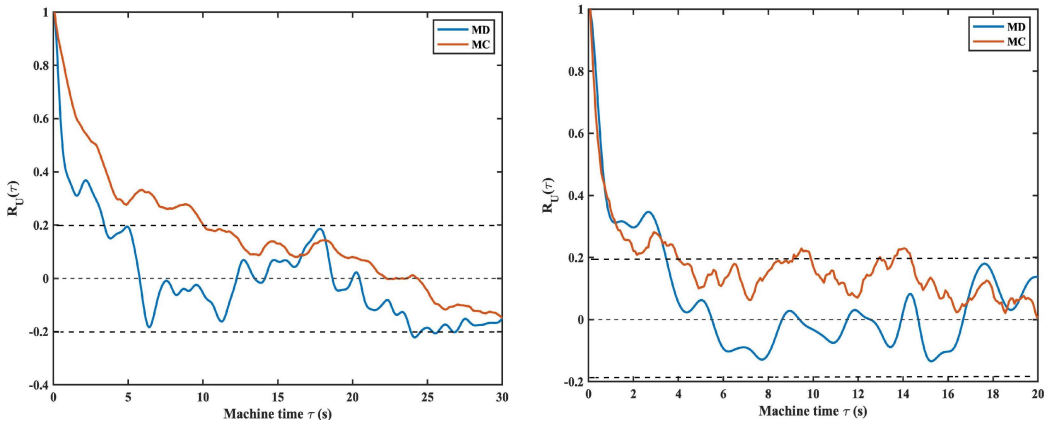
The Lennard-Jones phase diagram was used to locate densities at which the system is in gas phase and liquid phases [2]. At $\rho = 0.001$, the system is a very low density gas (an ideal gas). In this case, the MC simulation decorrelates successive configurations much faster than the MD simulation (by a factor of $\sim 10 - 20$), as in Figure 4.1. At $\rho = 0.05$, the system is still a gas, but does not behave ideally. The autocorrelation function drops faster to the uncorrelated limit for the MC simulation in this case as well, but by a smaller order of magnitude difference in decorrelation times. Figure 4.2 illustrates the same. In the high density liquid phase, however, the MD decorrelation time becomes comparable to the MC decorrelation time. A representative plot is shown in Figure 4.3, for the case where $\rho = 0.8$.

A change in the number of particles N, or the reduced temperature T, gave identical results in terms of the relative efficiency of the two methods (i.e., MC samples much better in low density systems, but MD becomes comparable or marginally better at higher densities in general).

The analysis here shows that for the simulation of low density systems in general, MC simulations are more advantageous, if evaluating equilibrium properties is the primary objective. The ability of MC simulation in a low density phase to make random and large unphysical moves enable it to sample microstates better. However,

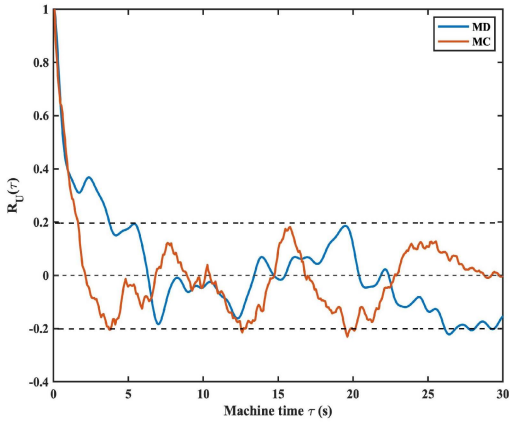
for a higher density system, there is a greater probability of overlap between two or more particles from a random MC move, and therefore MD becomes comparable to MC. There is a higher number of rejected moves in MC, or the necessity for a smaller MC move size, leading to decreased efficiency in sampling. Furthermore, the ability of MD to handle collective motions makes it comparable to MC in the liquid phase.

4.1.1 Effect of MC Move Size



(a)

(b)



(c)

Figure 4.4: Energy autocorrelation for various MC move sizes (a) Very large moves (b) Very small moves (c) Moves satisfying optimal acceptance ratio

In all the MC simulations above, the MC move size was adjusted after each sweep based on the move acceptance ratio. The MC code was adjusted in order to study this effect of the maximum move size on the decorrelation times.

The base case is where the move size is adjusted to get the optimum acceptance ratio of 0.5. Two other cases, one where the maximum move size is $0.9L$ and one where the maximum move size is $0.002L$ are studied. Here, L is the size of the simulation domain. The simulation parameters are set as $N = 216, \rho = 0.2, T = 1.4$. The results are illustrated in Figure 4.4. For very small move sizes, the acceptance ratio is very high $(0.9 - 1)$, but successive configurations are highly correlated, and therefore the decorrelation time is very high. For extremely large move sizes, most moves end up being rejected and therefore the acceptance ratio is very small $(0.1 - 0.2)$, which also increases decorrelation time. These decorrelation times are at least an order of magnitude higher than the case where the optimal acceptance ratio is maintained. Move size therefore has a significant effect on the efficiency of sampling. This is also a confirmation that an acceptance ratio around 0.5 is optimal for achieving good sampling.

Furthermore, it is observed that for very small or very large MC moves, the decorrelation times become high enough that they are comparable to MD or worse than MD, even for a low density simulation, as in this example.

4.2 Equilibrium Property Prediction

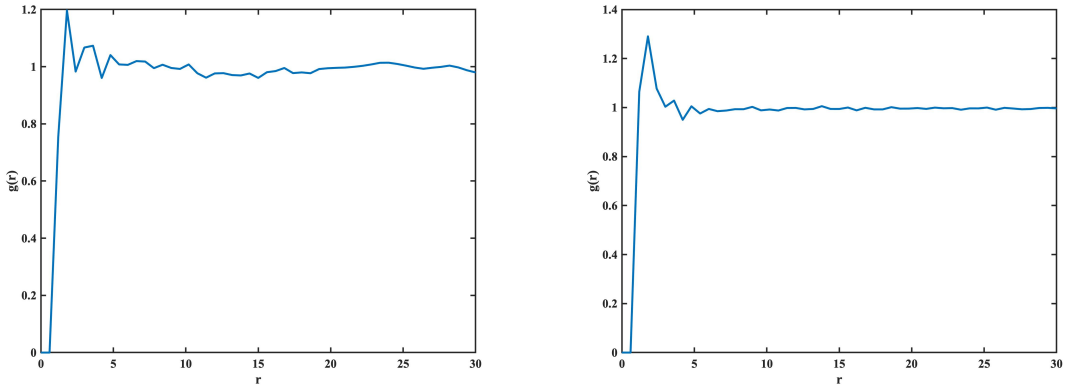
The thermodynamic properties that are studied here are the pressure P and the radial distribution function $g(r)$. In order to compute these properties, the number of particles in the simulations was set to $N = 216$, and the reduced temperature was set to $T = 1.2$. The reduced density ρ was varied.

4.2.1 Radial Distribution Function

The radial distribution function $g(r)$ was estimated for the ideal gas ($\rho = 0.001$), the non-ideal gas ($\rho = 0.05$) and the liquid ($\rho = 0.8$) phases. These are the same cases considered in the previous section to compare the sampling efficiency of the two methods.

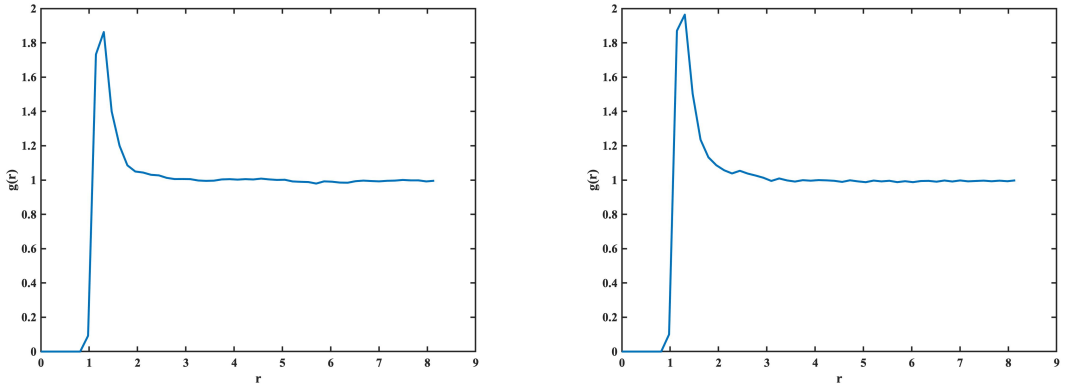
Both MD and MC give identical results for $g(r)$ in all three cases, and are also consistent with theoretical results. For an ideal gas, $g(r)$ is unity for $r > 1$, as in Figure 4.5.

For a non-ideal gas, the range of correlations is just the range of the intermolecular pair potential. Therefore, there is a peak at $r = 1$, followed by a uniform decay to unity, as in Figure 4.6.



(a) (b)

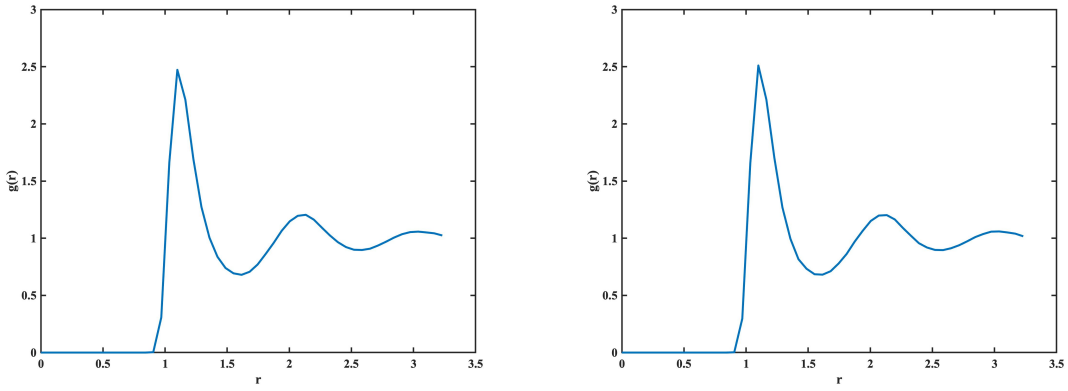
Figure 4.5: $g(r)$ predictions for an ideal gas (a) MD (b) MC



(a) (b)

Figure 4.6: $g(r)$ predictions for a non-ideal gas (a) MD (b) MC

In the liquid phase, the atoms are drawn close together because $\rho \sim 1$. Since the fluid is dense, there is a high probability of finding a nearest neighbor at $r = 1$. This is the first coordination shell, and there is a peak in $g(r)$ here. The presence of the first coordination shell tends to lower the probability of finding a neighbor at $r = 1.5$. Therefore, there is a trough in the $g(r)$ curve near $r = 1.5$. This then gives rise to a second coordination shell near $r = 2$. This trend results in an oscillatory pattern for $g(r)$ in the liquid phase, as in Figure 4.7.



(a) (b)

Figure 4.7: $g(r)$ predictions for a liquid (a) MD (b) MC

4.2.2 Pressure

The equilibrium pressure P is computed using both MD and MC simulations. In order to make realistic comparisons, the reduced pressure that is obtained from the simulation is converted to actual pressure using the Lennard-Jones parameters for argon ($\epsilon = 1.65 \times 10^{-21} J$, $\sigma = 3.4 \times 10^{-10} m$). This is then compared with the experimentally observed pressure for argon [3].

Both MD and MC methods result in predictions that agree well with literature, as in Table 4.1.

Table 4.1: Comparison of pressure obtained from simulation with experimentally observed pressure for argon at T = 144 K

ρ	P from MD (MPa)	P from MC (MPa)	P observed (MPa) [3]
0.001	0.5044 ± 0.0032	0.5027 ± 0.3316	0.5327
0.1	2.176 ± 0.0104	2.063 ± 0.1060	2.060
0.7	26.70 ± 1.338	37.10 ± 1.840	30.00
0.8	93.36 ± 1.437	91.45 ± 1.095	85.00

4.3 Vapor-Liquid Phase Coexistence

For most simulations in this section, only the MC method was used, since it is better, or only marginally worse than an MD simulation in most cases for sampling

equilibrium properties. The number of particles in all these simulations was set to $N = 216$.

4.3.1 Critical Point

The determination of the critical point, or in general, the phase coexistence curve is not straightforward using NVT ensemble MC. As a workaround, the critical temperature and density values of $T_c = 1.312$ and $\rho_c = 0.316$ available in literature are used to detect the critical point through the NVT simulation code that has been written.

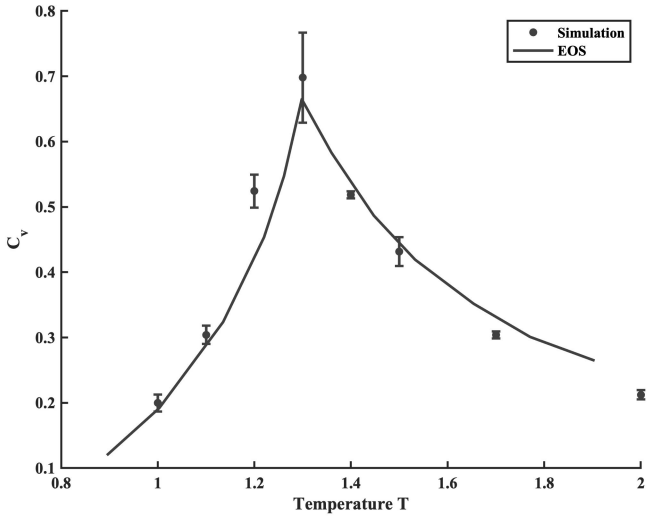


Figure 4.8: Isochoric heat capacities calculated along the saturation and the critical isochore curves for the Lennard-Jones fluid

The reduced temperature is varied from $T = 1$ to $T = 2$. The reduced density ρ was set to $\rho = \rho_c = 0.316$ for T greater than the observed T_c, and the phase coexistence curve [2] was used to set ρ to the saturated vapor density for $T < T_c$. The specific heat capacity per particle C_v is calculated at these temperature values. It is observed that the specific heat diverges at $T = T_c \simeq 1.3$. The results are consistent with the results from the Lennard-Jones equation of state [4]. This is also shown in Figure 4.8.

4.3.2 Coexistence Simulation

Once the consistency of critical point detection is verified with the theoretical value, a sample phase coexistence simulation is carried out using MC. The reduced density

was set at $\rho = 0.3$. Two simulations, one at $T = 0.7$, and the other at $T = 1.5$, are carried out. A histogram of densities is obtained by dividing the simulation domain into smaller volumes and estimating the density at each volume. The histogram plot of the densities shows phase coexistence at $T = 0.7$, as represented by the presence of two peaks in the density histogram (Figure 4.9), at $\rho = 0.16$ and $\rho = 0.72$. Although these peaks do not correspond to the exact phase coexistence curve, it still gives a qualitative confirmation that two phases are coexistent. On the other hand, the histogram of densities do not show multiple peaks at $T = 1.5 > T_c$. There is just a single peak at $\rho = 0.3$ (Figure 4.10).

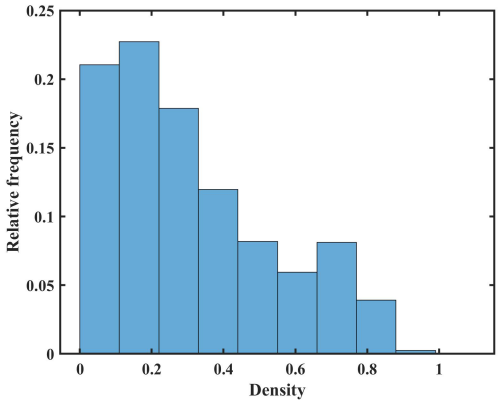


Figure 4.9: Histogram plot showing phase coexistence below critical point

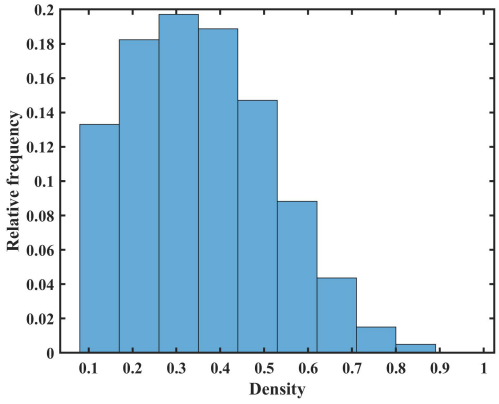


Figure 4.10: Histogram plot showing no phase coexistence above critical point

The coexistence simulation was also carried out using MD at $T = 0.7$ and $\rho = 0.3$. The final configuration (generated from an animation using VMD) is shown in Figure

4.11. A low density vapor phase can be observed roughly toward the middle of the simulation box, and a high density liquid phase can be seen near the corners of the box. This, once again, is a qualitative confirmation of phase coexistence. The animation of the MD simulation shall be made available upon request.

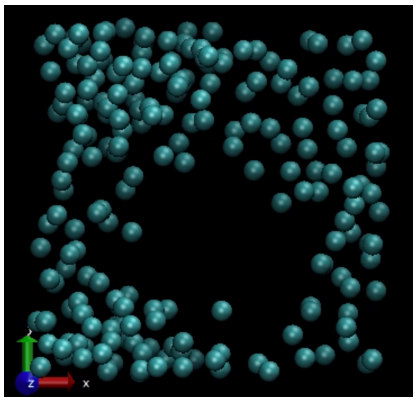


Figure 4.11: Final configuration of the system following MD simulation of vapor-liquid coexistence at $T = 0.7$, $\rho = 0.3$.

4.4 Dynamic Property Prediction

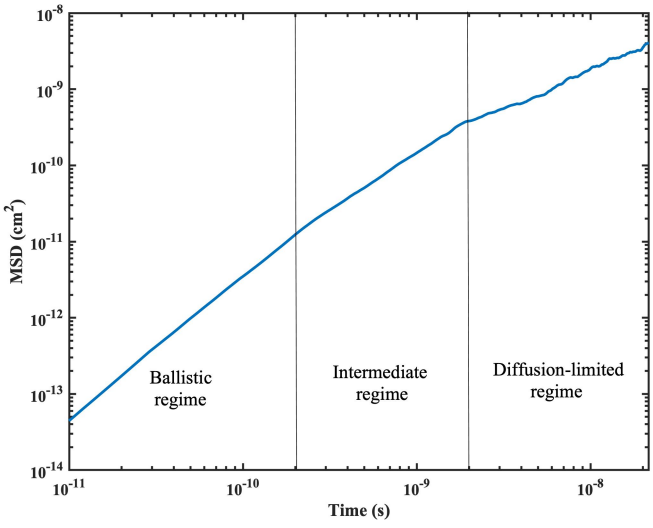


Figure 4.12: MSD versus time for argon particles at $T = 195K$, $P = 1atm$

In this section, the self diffusion coefficient for argon is calculated using MD simulations at different temperatures. The number of particles was fixed at $N = 100$. Since diffusion coefficient data are more readily available at atmospheric pressure, the density input to the simulation was fixed by using experimentally measured values of density at the given temperature and atmospheric pressure [3]. Figure 4.12 shows the plot of the mean squared displacement (MSD) as function of time t at a reduced temperature of $T = 1.622$ $(195K)$ and atmospheric pressure. The plot accurately captures the ballistic regime where the slope of the curve on a log scale has a slope of 2 at short times. This is followed by an intermediate regime, followed by which the slope becomes unity at large times in the diffusion-limited regime. It is this diffusion-limited regime that is used for estimating diffusivity.

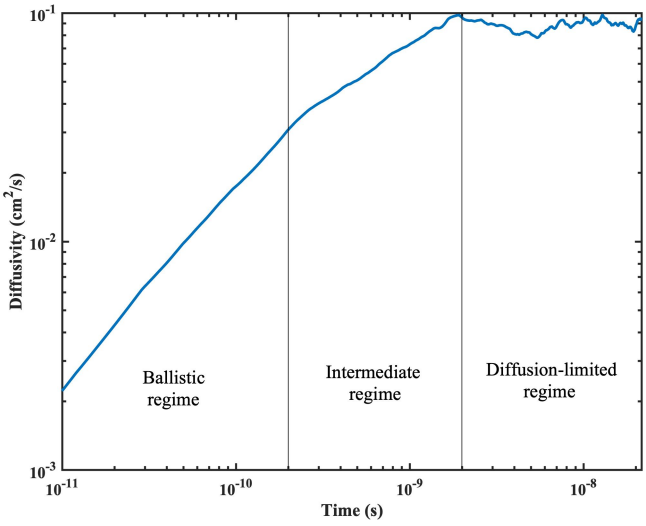


Figure 4.13: Self-diffusion of argon versus time at $T = 195K$, $P = 1atm$

A plot of the diffusion coefficient computed using the Einstein relation for the same conditions as above is shown in Figure 4.13. In the diffusion-limited regime, this predicted value becomes a constant. Uncorrelated samples of the diffusion-coefficient values are then averaged to compute the average and obtain an error estimate.

This process of estimating the diffusion coefficient for Argon is repeated for different temperatures and the resulting values are compared with those available in the literature [5]. There is a good agreement between results, as shown in Figure 4.14.

Although MC is in general a better choice for equilibrium property estimation, prediction of dynamic properties of a system like the diffusion coefficient is largely possible only using MD simulations, as illustrated in this section. This is because Monte Carlo methods lack a definition of time, and instead deal with moves/sweeps.

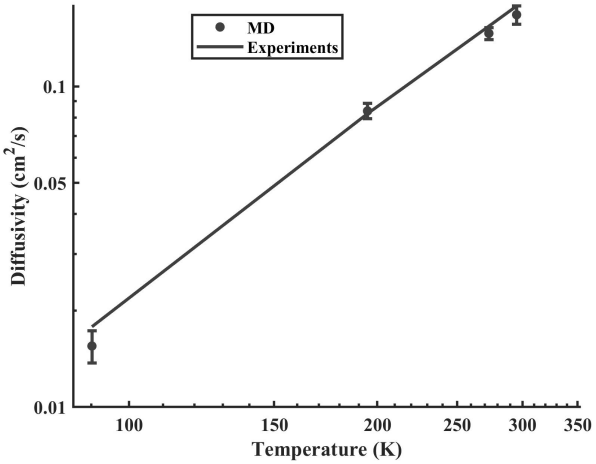


Figure 4.14: Temperature dependence of self-diffusion coefficient of argon at $1atm$

4.5 MC as a Predictor of Overdamped Brownian Dynamics

Although it was stated in the previous section that MD can predict dynamics of a system whereas MC in general can not, there is a special case where Metropolis MC can predict dynamics. In the limit of small move sizes, it has been shown that MC results can be connected to those from dynamic simulations. One such example is a study by Scarlett et al on the dynamics of colloidal crystallization using MC simulations [6].

It is stated here without proof that MC simulations and dynamic simulations equivalent when the move size is constrained in such a way that

$$K \equiv \frac{3}{8}\frac{\partial U}{\partial x}\frac{\delta}{k_B T} << 1 \tag{21}$$

is always satisfied.

MC simulations in 1D were carried out to demonstrate this equivalence. As a test case, a system of $10,000$ non-interacting particles were simulated in the absence of a potential energy function. This case is then essentially the free-diffusion process. Since there are no potential energy terms, there is no limit on the maximum MC move size in this case. The initial condition of the system was a pulse at $x = 0$ ($\sim \delta(x)$), and the particle positions after a *time* of 100 sweeps and 400 sweeps were recorded, and are shown in Figure 4.15.

The particle concentrations versus position fit a Gaussian curve, with the peak height after 100 sweeps being twice the peak height after 400 sweeps. The standard deviation of the curve after 400 sweeps is twice the standard deviation of the curve

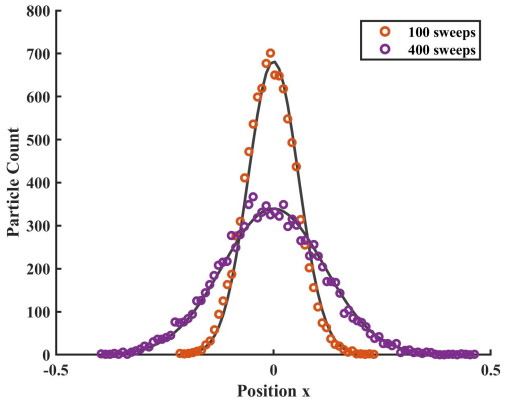


Figure 4.15: Free diffusion using MC simulations

after 100 sweeps. This is consistent with the scaling observed for the standard deviation with respect to time. ($\sigma \sim \sqrt{t}$). All one now needs to do is map these results with the available theoretical expressions in order to get a definition for an MC sweep in terms of actual time units.

Now, since the test case gave satisfactory results, a harmonic potential was added. This is given by $U = -\frac{1}{2}kx^2$. For this case, the MC move size was constrained according to Equation 21. The parameters chosen for the study are $k = 1$ and $k_BT = 1$ in reduced units. A plot of MSD versus number of sweeps from the MC simulation was obtained, and the resulting plot was mapped with Brownian Dynamics (BD) and MD simulations to get an equivalence between MC sweeps and the dynamic time step.

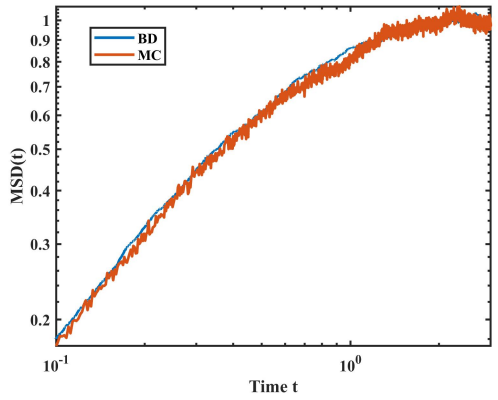


Figure 4.16: MSD versus time for an overdamped Brownian harmonic oscillator using MC and BD

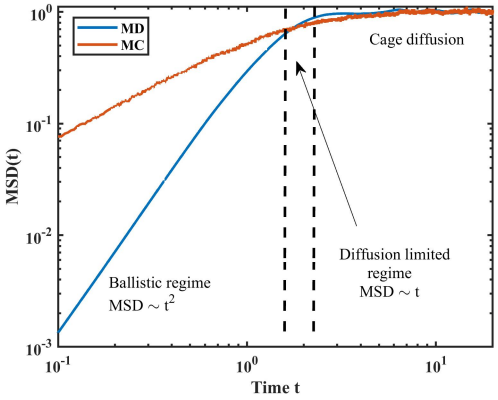


Figure 4.17: MSD versus time for an overdamped Brownian harmonic oscillator using MC and MD

From Figure 4.16, it can be seen that the MC simulations are consistent with the result predicted by the overdamped (diffusive) Brownian dynamics simulation. However, neither the MC nor BD simulations are able to capture the ballistic regime at short times. An MD simulation is still a more accurate representation of the dynamics in that it can capture the ballistic regime at short times. This has been illustrated in Figure 4.17. All the three methods give rise to the same estimate for the cage length, and match the theoretical value.

$$Cage\ length = \sqrt{\frac{k_B T}{k}} = \lim_{t \to \infty} \sqrt{MSD(t)} = 1 \tag{22}$$

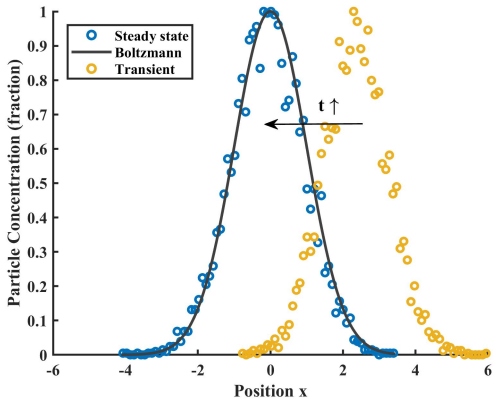


Figure 4.18: Evolution of concentration profile using MC dynamics

At steady state (long times), the particle concentration profile is consistent with the steady state solution to the Fokker Planck equation (Boltzmann relation), as in Figure 4.18. The initial condition is a pulse of 10,000 particles at $x = 5$ ($\sim \delta(x-5)$).

5 Conclusions

MD and MC simulations were applied to simulate a Lennard-Jones fluid. Both these simulation methods gave identical predictions of thermodynamic properties of the system. Differences arise in the sampling efficiency of these methods. While MC simulations provide a better sampling at low density, MD simulations become comparable or marginally better than MC simulations at higher densities. This difference arises because of the ability of MC to make large unphysical moves at low densities, while the maximum allowable move size becomes smaller at higher densities, leading to poorer sampling. The move size in MC simulations is a very significant parameter in determining the sampling efficiency of MC. A very small or very large move size are both not ideal choices. A move acceptance ratio around $0.4 - 0.6$ offers optimum sampling. Although MC is generally better for estimating equilibrium quantities, the determination of dynamic quantities like diffusivity is largely possible only using an MD simulation. As an illustration, the self-diffusion coefficient of argon was computed under various conditions to give results consistent with the literature. However, making small constrained moves in an MC simulation results in an accurate prediction of dynamics. As an example, an MC simulation of the 1D overdamped Brownian harmonic oscillator was carried out to show the equivalence between BD, MD and MC simulations.

References

1. Martyna, G. J., Klein, M. L. & Tuckerman, M. Nosé–Hoover chains: The canonical ensemble via continuous dynamics. *The Journal of Chemical Physics* **97,** 2635–2643 (1992).

2. Smit, B. Phase diagrams of Lennard-Jones fluids. *The Journal of Chemical Physics* **96,** 8639–8640 (1992).

3. Lemmon, E. Thermophysical properties of fluid systems. *NIST Chemistry WebBook* (1998).

4. Boda, D., Lukács, T., Liszi, J. & Szalai, I. The isochoric-, isobaric-and saturation-heat capacities of the Lennard-Jones fluid from equations of state and Monte Carlo simulations. *Fluid phase equilibria* **119,** 1–16 (1996).

5. Winn, E. B. The temperature dependence of the self-diffusion coefficients of argon, neon, nitrogen, oxygen, carbon dioxide, and methane. *Physical Review* **80,** 1024 (1950).

6. Scarlett, R. T., Crocker, J. C. & Sinno, T. Computational analysis of binary segregation during colloidal crystallization with DNA-mediated interactions. *The Journal of chemical physics* **132,** 234705 (2010).

A *Cleaner* autocorrelation plot

The autocorrelation plots present in the main report are for small particle numbers. Simulations with large particle numbers or large data sets have not been carried out in this project due to time constraints in obtaining results. However, in order to show that the sampling efficiencies based on autocorrelation functions using $N = 216$ particles provide a reasonable estimate, a simulation was run where a larger number of particles and more number of data points were used to give rise to a *cleaner* energy autocorrelation function. Figure A.1 shows the same, for $N = 400, T = 1.2, \rho = 0.8$. It is observed that the range of fluctuations about the zero correlation line is smaller in this case. These fluctuations are expected become lower and lower as the number of particles in the system is increased. But this could not be verified as it requires really long simulation runs.

As in the main report for high density systems, the MD and MC decorrelation times are comparable in this case. This is confirmation that using fewer data points results in a representation that is accurate, although it is *noisier* representation of the relative efficiency of MD and MC methods.

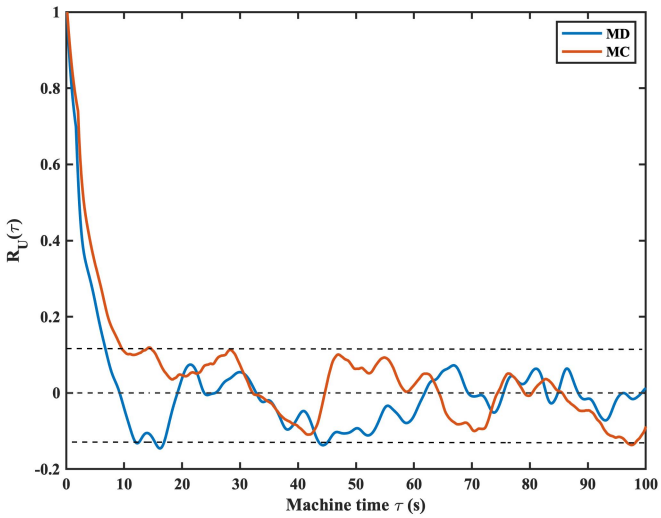


Figure A.1: A *cleaner* energy autocorrelation plot

B MATLAB Code

Programming experience:

– Introductory courses in computer programming, data structures and algorithms in freshman year of undergrad.

$\sim$ 40-50 pages of code in C/C++

$\sim$ 20 pages of code in MATLAB this semester (Numerical methods course)

B.1 Molecular Dynamics

```
function [dx,r,v,Tk,g,rg,u_out,P,t_md] = md(n,L,dt,T,dim,nStep)
tol = 10^-6;
% LJ simulation always done in 3d for this project
if dim > 1
    [r,v] = init3d(n,L,T,tol);
    dx = zeros(nStep,n);

    % 1d sim, BHO
elseif dim == 1
    r = zeros(n,1);
    v = rand(n,1);
    v = v - mean(v);
end
r0 = r;
r_ini = r;
f = force(r,L,dim);

% Nose Hoover thermostat variables
p = 5;
Q = zeros(p,1);
tau = 25*dt;
Q(1) = n*T*tau^2;
for j = 2:p
    Q(j) = 3*T;
end

xi = zeros(p,1);
%Data analysis part
t = 0;
count = 0;
g = 0;
rg = 0;
iter = 0;
u_out = zeros(nStep,1);
Tk = sum(dot(v,v))/(dim*n);

% Used interchangeably for pressure (2-3d) and MSD (1d)
P = zeros(nStep,1);

tic;

% MSD computation for 1d case
if dim == 1
    for i=1:nStep
```

```
45          dx = r - r_ini;
46          P(i) = dot(dx,dx);
47          f = force(r,L,dim);
48
49          % MD
50          r = r + dt*v;
51          v = v + dt*f - dt*v + sqrt(2*dt)*normrnd(0,1,[n 1]);
52
53          % BD
54  %           r = r + dt*f + sqrt(2*dt)*normrnd(0,1,[n 1]);
55      end
56      P = P/n;
57      t_md = toc;
58      return;
59  end
60
61  % Use only if you want to write config to file
62  % fileid = fopen('md.xyz','a+');
63
64  % Use only if you want to write U to file
65  % fileid = fopen('potential.out','a+');
66
67  for i=1:nStep
68      r = r + dt*v + 0.5*dt*dt*f;
69
70      % Diffusivity calc for PBC in 2d-3d case
71      r = mod(r,L);
72
73      %%% Data generation for diff. calc.
74      delta = r(:,1) - r0(:,1);
75      for j=1:n
76          if delta(j) > 0.5*L
77              delta(j) = delta(j) - L;
78          elseif delta(j) < -0.5*L
79              delta(j) = delta(j) + L;
80          end
81          if i==1
82              dx(i,j) = delta(j);
83          else
84              dx(i,j) = dx(i-1,j) + delta(j);
85          end
86      end
87      %%%
88
89
90      %%% Radial dist function
91      if mod(iter,10)==0 && iter > 0.5*nStep
92          count = count+1;
93          [g,rg] = gr(r,n,L,dim);
94          if count==1
95              g_old = g;
96              continue;
97          end
98          g = ((count-1)*g_old + g)/count;
99          g_old = g;
100     end
101     %%%
102
103     r0 = r;
104
```

```
105       v = v + 0.5*dt*f - 0.5*xi(1)*dt*v;
106       f = force(r,L,dim);
107       v = v + 0.5*dt*f - 0.5*xi(1)*dt*v;
108
109       Tk = sum(dot(v,v))/(dim*n);
110       xi_dot = nh(T,Q,xi,p,Tk,n,dim);
111       xi = xi + dt*xi_dot;
112
113      % disp(Tk);
114
115       t = t + dt;
116       iter = iter + 1;
117  %      P(iter) = pres(n,L,Tk,r);
118       u = lj(r,L,dim);
119       u_out(iter) = u;
120  %      fprintf(fileid,'%f \n',u);
121
122      % Use only if you want to write config to file
123  %      sig = 3.4; % In Angstrom
124  %      fprintf(fileid, '%d \n \n',n);
125  %      for j=1:n
126  %          fprintf(fileid,'Ar %f %f %f \n',r(j,1)*sig,r(j,2)*sig,r(j,3)*sig);
127  %      end
128  end
129  % fclose(fileid);
130  t_md = toc;
```

B.2 Nose Hoover Chain Thermostat

```
1  function [xi,xi_dot] = nh(T,Q,xi,p,Tk,n,dim)
2  xi_dot = zeros(p,1);
3  xi_dot(1) = -xi(1)*xi(2);
4  xi_dot(1) = xi_dot(1) + (dim*n*Tk - dim*n*T)/Q(1);
5  for j=2:p-1
6       xi_dot(j) = (Q(j-1)*xi(j-1)*xi(j-1) - T)/Q(j) - xi(j)*xi(j+1);
7  end
8  xi_dot(p) = (Q(p-1)*xi(p-1)*xi(p-1) - T)/Q(p);
9  end
```

B.3 Metropolis Monte Carlo

```
1  function [r,g,rg,u_out,P,t_mc] = mmc(n,L,T,dim,nStep)
2  u_out = zeros(nStep,1);
3  P = zeros(nStep,1); % Used interchangeably for pressure (2-3d) and MSD (1d)
4  beta = 1/T;
5  tol = 100;
6
7  % LJ simulation always done in 3d for this project
8  % 2d or 3d sim. of LJ fluid
9  if dim > 1
10       [r0,~] = init3d(n,L,T,tol);
11       r = r0;
12       delta = 0.1*L;
13       u = lj(r,L,dim);
14 end
15
16 % MC as a generator of overdamped Lang. dynamics (always 1d)
17 % 1d sim, free diffusion, BHO
```

```
18 if dim == 1
19     r0 = zeros(n,1);
20 %     r0 = 5*ones(n,1);
21     r = r0;
22     r_ini = r0;
23     % Free diffusion
24     % u = 0;
25     % BHO
26     u = spring(n,r);
27     delta = 0.01;
28 end
29
30 count = 0;
31 tic;
32 for i=1:nStep
33     %"Sweep"
34     p_acc = 0;
35     for j=1:n
36         q = rand;
37         ind = ceil(q*n);
38         dr = 2*delta*(rand(1,dim)-0.5);
39         r(ind,:) = r0(ind,:) + dr;
40
41         if dim == 1
42             % Free diffusion
43             % du = 0;
44
45             % Overdamped BHO
46             du = 0.5*(r(ind)*r(ind) - r0(ind)*r0(ind)); %Spring k = 1
47         end
48         % Periodic BC
49         if dim > 1
50             r(ind,:) = mod(r(ind,:),L);
51             du = delta_u(r,r0,ind,L,dim);
52         end
53
54         u = u + du;
55         accept = min(1,exp(-beta*du));
56         s = rand;
57         if s > accept
58             r(ind,:) = r0(ind,:);
59             p_acc = p_acc-1;
60             u = u - du;
61         end
62
63         % MSD computation for 1d case
64         if dim == 1
65             dx = r(ind,1) - r_ini(ind,1);
66             dx2 = dx*dx;
67             P(i) = P(i) + dx2;
68         end
69         r0 = r;
70         p_acc = p_acc + 1;
71     end
72
73     g = 0;
74     rg = g;
75
76     if dim > 1
77         %%% Modify move size if acceptance neq 0.5
```

```
78          p_acc = p_acc/n;
79          % disp(p_acc);
80          if p_acc > 0.5
81              delta = delta*1.1;
82              if delta > L
83                  delta = L/1.1;
84              end
85          elseif p_acc < 0.5
86              delta = delta*0.9;
87          end
88          %%%
89
90          %%% Radial dist function
91          if mod(i,10)==0 && i > 0.5*nStep
92              count = count+1;
93              [g,rg] = gr(r,n,L,dim);
94              if count==1
95                  g_old = g;
96                  continue;
97              end
98              g = ((count-1)*g_old + g)/count;
99              g_old = g;
100             div = 3;
101             dens_hist(r,n,L,div);
102         end
103         %%%
104     end
105     if dim == 1 && i == nStep/20
106         r_mid = r;
107     end
108     u_out(i) = u;
109     %     P(i) = pres(n,L,T,r);
110 end
111 if dim == 1
112     P = P/n;
113     r = [r_mid, r];
114 end
115 t_mc = toc;
```

B.4 Lennard-Jones Potential

```
1  function [u] = lj(r,L,dim)
2  u = 0;
3  n = length(r);
4  for i=1:n
5      for j=i+1:n
6          rij = r(i,:) - r(j,:);
7              for k=1:dim
8                  if rij(k) > 0.5*L
9                      rij(k) = rij(k)-L;
10                 elseif rij(k) < -0.5*L
11                     rij(k) = rij(k)+L;
12                 end
13             end
14         mag_rij = sqrt(sum(rij.^2));
15         u = u + 4*((1/mag_rij)^12 - (1/mag_rij)^6);
16     end
17 end
```

B.5 Force Computation for MD

```
function [f] = force(r,L,dim)
n = length(r);
f = zeros(n,dim);
% 1d sim, BHO
if dim == 1
    f = -r; % Spring k = 1
    return;
end
% Continue with a 2d or 3d LJ simulation if dim neq 1
for i=1:n
    for j=1:n
        if i~=j
            rij = r(i,:) - r(j,:);
            for k=1:dim
                if rij(k) > 0.5*L
                    rij(k) = rij(k)-L;
                elseif rij(k) < -0.5*L
                    rij(k) = rij(k)+L;
                end
            end
            mag_rij = sqrt(sum(rij.^2));
            fij = 24*(2*mag_rij^(-14) - mag_rij^(-8))*rij;
            f(i,:) = f(i,:) + fij;
        end
    end
end
end
```

B.6 Radial Distribution Function

```
function [g,r0] = gr(r,n,L,dim)
dens = n/L^dim;
dr = L/100;
r0 = (0:dr:(0.5*L))';
g = zeros(length(r0)-1,1);
m = n*(n-1);
rij = zeros(m,1);
m = 1;
for i=1:n
    for j=1:n
        if i~=j
        delta = r(i,:) - r(j,:);
         for k=1:dim
            if delta(k) > 0.5*L
                delta(k) = delta(k)-L;
            elseif delta(k) < -0.5*L
                delta(k) = delta(k)+L;
            end
         end
        %delta = mod(delta,L);
        rij(m) = sqrt(dot(delta,delta));
        m = m + 1;
        end
    end
end
m = m-1;
```

```
for i=2:length(r0)
    count = 0;
    for j=1:m
        if rij(j)>=r0(i-1) && rij(j) < r0(i)
            count = count + 1;
        end
    end
    if dim == 1
        dens_act = count/(n*(r0(i)^dim - r0(i-1)^dim));
    elseif dim == 2
        dens_act = count/(n*pi*((r0(i))^dim - (r0(i-1))^dim));
    elseif dim == 3
        dens_act = 3*count/(4*n*pi*((r0(i))^dim - (r0(i-1))^dim));
    end
    g(i) = dens_act/dens;
end
```

B.7 Pressure

```
function [P] = pres(n,L,T,r)
V = L^3;
P = n*T;
for i=1:n
    for j=i+1:n
        rij = r(i,:) - r(j,:);
            for k=1:3
                if rij(k) > 0.5*L
                    rij(k) = rij(k)-L;
                elseif rij(k) < -0.5*L
                    rij(k) = rij(k)+L;
                end
            end
            mag_rij = sqrt(sum(rij.^2));
            fij = 24*(2/mag_rij^14 - 1/mag_rij^8)*rij;
            P = P + dot(rij,fij)/3;
    end
end
P = P/V;
```