Simulation Methods in Statistical Physics

(FK 8028)

Programming project report 5: Metropolis Monte Carlo

Submitted by Mikhail Ivanov

Introduction

For the last part of the programming project, Metropolis Monte Carlo code was developed, which is largely based on previously developed molecular dynamics code. For a simple system of liquid argon, only one type of Monte Carlo move is needed – a random displacement of a randomly picked Ar particle. The fact that the displaced particle is selected randomly ensures that the next configuration depends only on the previous state - that means that after reaching equilibrium, generated configurations form Markov chain. Since the particle is selected at random and there is equal possibility of negative or positive displacement, the condition of detailed balance is fulfilled (equal probability of a certain move and the corresponding reverse move) so we can expect correct sampling of the configurational space of the simulated system.

Method

Metropolis Monte Carlo algorithm used in the project can be outlined as follows:

Initialization:

1. Generate lattice of 1000 Ar atoms with desired density

2. Calculate total potential energy of the system (E_{tot})

MC move:

1. Pick a random particle

2. Calculate its energy (E_{old})

3. Displace the particle:

$$x_{new} = x_{old} + \Delta(ranf - 0.5)$$
$$y_{new} = y_{old} + \Delta(ranf - 0.5)$$
$$z_{new} = z_{old} + \Delta(ranf - 0.5)$$

where *ranf* is a random number uniformly distributed over [0, 1) interval and Δ is a small displacement.

4. Calculate energy of displaced particle (E_{new})

5. If $\Delta E < 0$: accept the move – update position and add ΔE to the E_{tot}

If $\Delta E > 0$: accept with probability $\exp(-\beta\Delta E)$

If the move is accepted - update position and add ΔE to the E_{tot}

If the move is rejected – return the particle to the initial position and keep E_{tot} the same

6. Return to the step (1) until the maximum number of steps is reached

Energy is saved every step, positions are saved every 1000 steps and then used for RDF calculation.

Simulation was performed for a system of 1000 Ar atoms, starting from a lattice with density ρ = 1374 kg/m^3 and temperature T = 95 K. For the final simulation, 0.25 Å displacement step was used, 500 thousands steps were performed for equilibration and 50 millions Monte Carlo steps were carried out for the production run.

Total potential energy and RDFs were calculated and corresponding errors were estimated using block averaging.

Heat capacity per particle in k_B was estimated using the following equation:

$$C_v = \frac{3N}{2} + \frac{var(U)}{T^2}$$

(given by Michael)

Results

1. The effect of the Acceptance Ratio

Before starting the production Monte Carlo run, several displacement steps were tried to find the optimal one. The results of the optimization are presented on Figure 1:

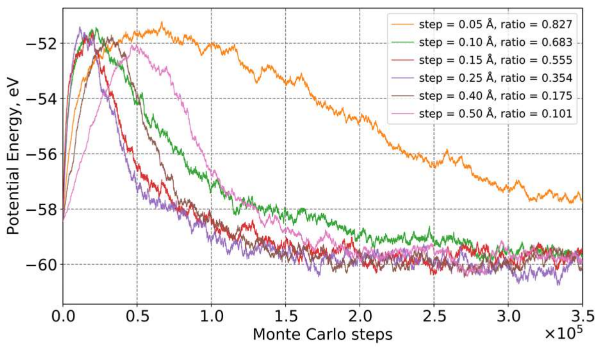

Figure 1 – Monte Carlo runs with different displacement steps

One can see that different displacement steps yield different acceptance ratios – from 82.7% for 0.05 Å displacement step to 10.1% for 0.50 Å displacement step. Every simulation starts with the lattice and before transforming into liquid, the system has to go through a certain energy barrier. It is clearly visible that some simulation go through the barrier faster than others – the simulation with the highest (82.7 and 68.3%) and lowest acceptance ratios (10.1 and 17.5%) perform much worse then simulations with 55.5 and 35.4% acceptance ratios. Probably, the latter displacement steps offer the compromise between the displacement step length and the acceptance ratio. The important conclusion from this test is that high acceptance ratio as well as large displacement step don't necessarily mean fast and efficient scan of the configurational space. For the production run displacement step of 0.25 Å was chosen. After 500k steps with the selected displacement the simulated system reaches equilibrium, as the values of potential energy are not drifting, but only fluctuating around the mean value (Figure 2):

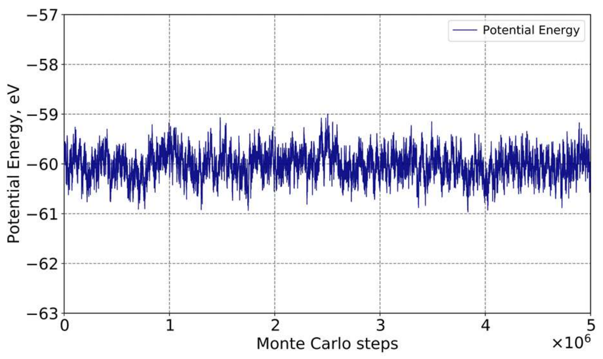

Figure 2 – Potential energy of the simulated system within 5 million MC steps after 500 thousands MC steps of equilibration period

2. Block averaging

Block averaging was performed for the potential energy to derive the 'correlation time' for the present simulation. Figure 3 shows the potential energy values and corresponding error bars with varying block length:

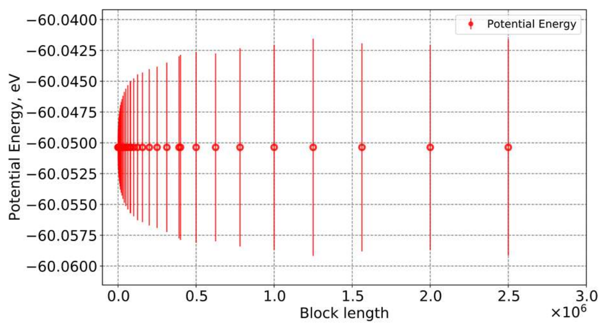

Figure 3 – Potential energy of the simulated system with varying block length

On Figure 4 and 5 standard deviations and errors with varying block lengths are presented:

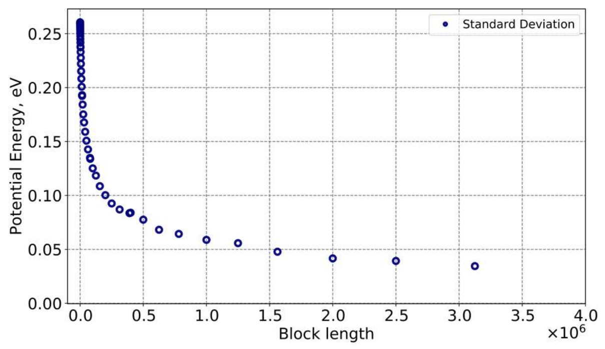

Figure 4 – Standard deviation of the potential energy of the simulated system with varying block length

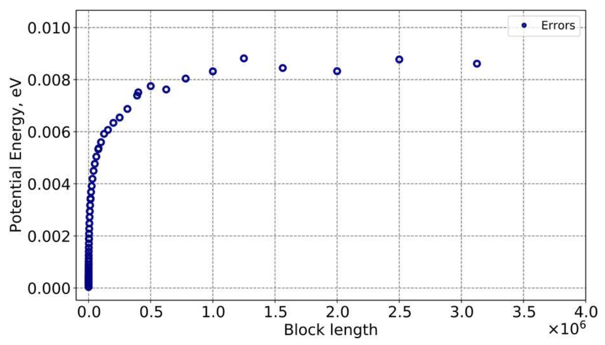

Figure 5 – Estimated errors of the potential energy of the simulated system with varying block length

From these figures it is possible to conclude that the correlation in the system decays almost completely after 500 thousands MC steps, so the 50M production run should be divided to 100 blocks with 500k steps each to ensure analysis of the independent data points.

Potential energy of the system and corresponding error was estimated using block averaging:

$$U(MC) = -60.0503 \pm 0.0077\, eV\ (T = 95\ K)$$

which is very close to the value of potential energy from MD simulations:

$$U(MD) = -59.8986 \pm 0.0084\, eV\ (T = 96.268\ K)$$

The energy is slightly higher due to higher mean temperature.

Heat capacity is also very close to the value from the MD simulation:

$C_v(MC) = 1.59\ k_B$; $C_v(MD) = 1.58\ k_B$

Radial distribution function is also very similar (Figure 6-8):

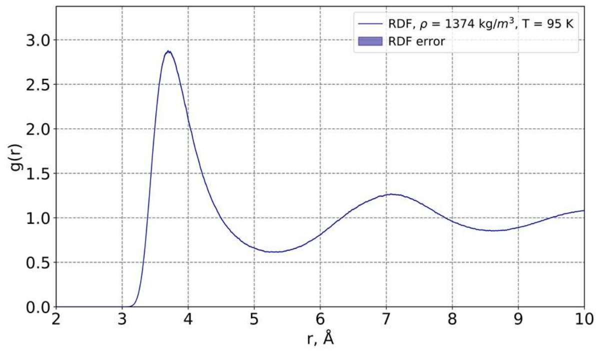

Figure 6 – Radial distribution function (global view, error bars are unnoticeable) from Monte Carlo simulation

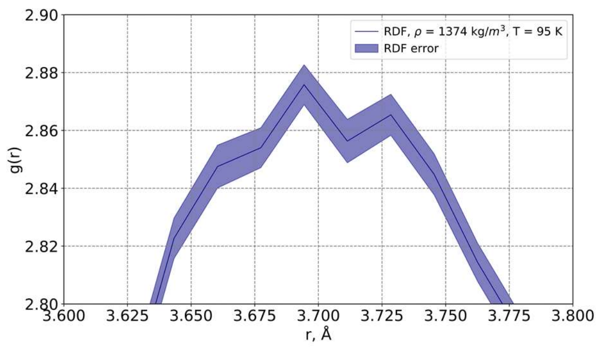

Figure 7 – Radial distribution function (first RDF peak) from Monte Carlo simulation

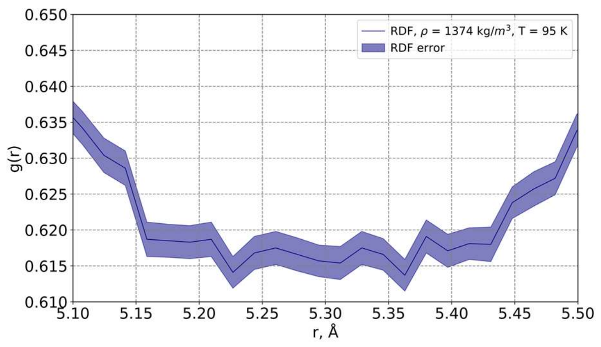

Figure 8 – Radial distribution function (first RDF minimum) from Monte Carlo simulation

Radial distribution function derived from MC simulations are generally the same as from MD simulations, though they are a bit coarser as seen from Figure 7 and 8, and maximum errors are also a bit higher (0.0075 in MC and 0.0056 in MD).

3. Comparing performance of Monte Carlo and molecular dynamics simulations

There is no time in Monte Carlo simulations, though there is one metric we can use for the comparison – how many blocks, that give an independent data point are produced per a time unit.

Same hardware was used for both simulation (hexa-core i7-8700K CPU). For the case of MD simulations, 80 ps simulation with 4 fs time step was carried out (20000 MD steps). Block length of 250 steps was chosen and with the total calculation time of 35 minutes, the MD performance for the present system is approximately 2.28 blocks per minute. For this run, all six CPU cores were utilized, as the force calculation routine was parallelized. For the case of MC simulations, parallelization has failed, so the simulations were run on one core. Production run lasted for 153 minutes, saving energies of 50 million configurations. With the block length of 500 thousands steps, 100 blocks were generated with the performance of approximately 0.65 blocks per minute. However, theoretically there is no limit for Monte Carlo parallelization, as at equilibrium it is possible to start independent chains, and after the simulation one may just merge them and analyze all the calculated averages. So it is possible to project that with correctly implemented parallelization scheme the performance will increase six fold to 3.92 block per minute, which is around 58% more effective than MD simulation. However, besides the actual performance, one should consider that the molecular dynamics algorithms are generally more flexible with respect to the system and simulation conditions, so it is quite hard to conclude what method is the most effective one.

Conclusion

Overall, the most impressive result is that different computational methods – Monte Carlo and Molecular dynamics may produce almost identical results.