Homework 5

Due 2019 November 8

Report instructions

Please upload *one* PDF file to Gradescope (Entry code: ME62YG).

For this homework you will implement code (on PrairieLearn) for a Monte Carlo simulation code to calculate the transition temperature of argon (as you did in homework 3, using molecular dynamics). You will be able to re-use many pieces of your code from earlier homeworks, so *if you are concerned that they do not fully work, please make sure you get those working (by posting on Canvas and/or by attending office hours)*!

When reusing the old MD code, set box length to L=4.0.

Introduction

Potential Energy

We will start by writing a Metropolis Monte Carlo code. Remember, in classical Metropolis Monte Carlo you will be sampling configurations with probability $\exp(-\beta V)$, where V is the potential energy of the configuration and $\beta = 1/kT$ is the inverse temperature. These probabilities should be familiar to you as the Boltzmann distribution for a canonical ensemble at thermodynamic equilibrium.

First, we should verify that we can evaluate the potential energy on a single particle in your system. You should already have code that helps you do this from your molecular dynamics simulation. Adapt it for this homework.

Moves

Metropolis Monte Carlo works by proposing particle moves at random, and probabilistically accepting or rejecting these moves according to the Metropolis criterion. We need to implement a function that updates the position of a single particle. You will update a particle position x by choosing the new position $x' = x + \eta$ where η is chosen from a Gaussian distribution of mean 0 and variance σ^2, i.e., the probability of proposing a move to position x' such that $\eta = x' - x$ is

$$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\eta^2/(2\sigma^2)\right).$$

To study $T(x \to x')$ as the probability of drawing the displacement $(x' - x)$ from a Gaussian distribution centered on x with variance σ^2, we write:

$$T(x \to x') = \frac{1}{\sqrt{2\sigma}} e^{-(x'-x)^2/2\sigma^2}.$$

Note that $T(x \to x') = T(x' \to x)$ since $(x' - x)^2 = (x - x')^2$. For many more sophisticated moves, this is not the case.

Metropolis Acceptance Criterion

Now, let's calculate the acceptance probability. Remember, in Metropolis Monte Carlo, the probability of accepting a move from position x to x' is

$$A(x \to x') = \min\left(1, \frac{e^{-\beta V(x')} T(x' \to x)}{e^{-\beta V(x)} T(x \to x')}\right)$$

where $V(x)$ is the potential energy of the system in configuration x, and $T(x \to x')$ is the probability of proposing a move from x to x'.

Metropolis Monte Carlo Algorithm

Remember the steps of a basic Metropolis Monte Carlo:

1. Choose a particle to move (this *can* be done by selecting particles at random, but it is also acceptable and easier to implement a loop over all particles that moves each particle in succession).
2. Evaluate the energy on this particle.
3. Move this particle with a function `UpdatePosition()`.
4. Evaluate the energy again, getting the difference ΔE from the old energy (before the move).
5. Accept the move with probability $A(x \to x')$; otherwise reject (remember if you are rejecting to return the particle to its old position).

Monitoring Acceptance and Setting up Runs

Before we set up some runs to generate data, we want to add functionality to our code that calculates acceptance ratios (i.e., measure the total number of accepted moves divided by the total number of attempted moves). Knowing the acceptance ratio of your MC simulation is particularly important to judge whether you are sampling configuration space efficiently. You can tune your simulation by adjusting the value of σ, which sets the length scale for your attempted moves and thus determines the acceptance ratio of these moves.

For all runs here, unless explicitly noted otherwise, use the following simulation parameters:

```
L = 4.0 # box length
rho = 1.0 # number density
N = 64 # particle number
M = 48.0 # mass
T = 2.0 # Temperature; beta = 0.5; for canonical ensemble.
```

Standard Metropolis Monte Carlo Acceptance Criterion

Take the statement of detailed balance,

$$P(x)T(x \to x')A(x \to x') = P(x')T(x' \to x)A(x' \to x),$$

which says that the flux of particles from point x to x' is equal to the flux in the reverse direction, i.e., the net flux is zero. Here, $P(x)$ is the probability of the system being in configuration x, $T(x \to x')$

is the probability of proposing a move from x to x', and $A(x \to x')$ is the probability of accepting the proposed move.

Force-Bias Type Move (Smart MC)

Above, we implemented a Gaussian displacement move that proposed a displacement drawn from a Gaussian distribution with mean 0 and specified variance σ^2. In this case the forward and reverse sampling probabilities canceled out, i.e.

$$\frac{T(x \to x')}{T(x' \to x)} = 1$$

which you derived. Often it is non-trivial to find the reverse sampling probability, $T(x' \to x)$, given a sampling probability $T(x \to x')$. We will study one case here.

The motivation of a "force-bias type" move is to propose a displacement that is not random, but rather is biased in the direction of the force acting on the particle to be moved. To implement this idea, we will make the same Gaussian displacement described above, but we will shift the center of our Gaussian in the direction of the force. The center of the Gaussian will be shifted by

$$x_{\text{adjust}} = \frac{F(x)}{2m}(\delta t)^2.$$

Thus the new move will choose a new position $x' = x + \eta + x_{\text{adjust}}$ where η is chosen from a Gaussian distribution with probability

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\eta^2/(2\sigma^2)}.$$

This is called "smart Monte Carlo," and has an asymmetry in the transition probability which must be reflected in the Metropolis acceptance criterion.

Your report

1. Show that the standard Metropolis Monte Carlo acceptance criterion below satisfies the detailed balance equation.

$$A(x \to x') = \min\left(1, \frac{P(x')T(x' \to x)}{P(x)T(x \to x')}\right)$$

   Note this should only take a couple lines of algebra, but it is very important to understand where this equation comes from. Note also that this expression for $A(x \to x')$ is not unique, but it is commonly used because it maximizes acceptance of proposed moves.
2. By running your code, find a value of σ that gives you an acceptance ratio between 0.2 and 0.5. An acceptance ratio in this broad range is likely to give you optimal efficiency. Quote your acceptance ratio and your choice of σ. Using this value, perform a run of 100 passes. The term "passes" means one attempted move on each particle, so 100 passes is equivalent to 6400 single-particle moves for a system of 64 particles. Produce a plot of the potential energy as a function of MC step, and a plot of the pair correlation function for this run.

3. Now, find a value of σ that gives an acceptance ratio below 0.01; state the value of σ you are using. Run your simulation again for 100 passes and produce plots of the energy and pair correlation function. Find a value of σ that gives an acceptance ratio greater than 0.95; state the value of σ you are using. Run your simulation again for 100 passes and produce plots of the energy and pair correlation function. Comparing data from your three runs, discuss what happens to your simulation if your acceptance ratio is either effectively 0 or effectively 1. State specifically how this leads to an inefficient simulation in each case.
4. Now, using your choice of σ in the optimal range of 20% to 50% acceptance, perform some longer runs. Run the above simulation conditions for a minimum of 2000 passes.Plot of the energy as a function of Monte Carlo simulation steps made. State the total number of Monte Carlo steps, the average energy, and the standard error for your simulation. Plot the pair correlation function and structure factor.
5. Now do two simulations, one at temperature T=1.0 and one at a temperature below T=0.5. Produce plots of the energy, structure factor, and pair correlation function at each temperature. Compare these results with your data from the MD simulations in HW 3.
6. For the "smart MC" move, write an expression for $T(x \rightarrow x')$. Write an expression for the reverse probability $T(x' \rightarrow x)$. Combine these terms to write the acceptance criterion $A(x \rightarrow x')$.
7. Finally, modify (but save your old code) your code to switch over to using this move. Again, measure the energy of your system using the same number of steps as you did before. Submit a trace plot, number of steps, energy and standard error of your simulation. Compare the standard error you get using this move with the standard error you get using the more naive move. Also, compare the total number of minutes it took to run. Which move is better? Why?
8. **This is optional for everyone with 3 Credit Hours:** Now we would like you to design your own move that you think would be more efficient than the conventional Gaussian displacement move we used above. The only requirement is that your move obey detailed balance, which you will demonstrate. Describe your move in detail and explain why you think it will be more efficient. For this new move, write an expression for the sampling probability $T(x \rightarrow x')$. Write the reverse probability $T(x' \rightarrow x)$. Write an expression for the acceptance probability $A(x' \rightarrow x)$.
9. **This is optional for everyone:** Finally, let's look at a situation where such a move should be particularly effective, looking at a *vacancy*. To do this, you should take the following steps
    1. Figure out how to initialize your system with a vacancy in it (i.e. create a new position array that has one less particle and copy all but one of the particles in it).
    2. Add an observable that measures where your vacancy is. One effective method to do this is to define a number of lattice sites and say your vacancy is the lattice site further from any of the particles.
    3. Alternately using both moves, measure the approximate number of steps it takes for the vacancy to move from one site to another.