ME45211-23 Particle-based Modeling of Fluids
Assignment 1: Monte Carlo

April 29, 2024

Introduction

Monte Carlo (MC) simulations allow for computing thermodynamic properties by sampling random configurations of the system. During lectures, you learned about the theory behind MC simulations. In this assignment, you will create your own MC code (Part 1), where simulations in the Canonical (NVT) ensemble can be performed, and compute properties of methane, CH_4 (Parts 2 and 3).

Instructions for your report

You should send to the course instructors the following items:

- A short and precise report (in PDF format) containing your results, i.e, answers to questions, plots or figures. Make sure that your answers clearly correspond to a specific question. If a piece of the code is necessary in the answer then please refer to it using its line number in the python file.
- Your final MC code in one python file with sufficient comments to make the code easily understandable. The functions should be named as instructed in the assignment below. The code is expected to be readily executable without any additional modification needed. If any input parameters are required, please explain them clearly in the report.

Make sure that your report contains your name and student id.

For answering the questions you can use both the lecture notes and the appropriate chapters in the proposed textbooks. Keep an eye on Brightspace for announcements concerning the assignment.

You can use any programming language of your choice, however, the use of Python 3.x is recommended. The use of PyCharm[1] as an editor which works on Windows, Mac and Linux is also recommended. Some useful functions will also be provided in Python 3.x. Before you start, make sure that basic Python packages such as numpy, matplotlib, scipy and math are installed.

Assignment Part 1: Development of the MC code

Question 1.1

Write a function (named "totalEnergy") that returns the total energy of the system. In this assignment only non-bonded van der Waals interactions between particles are considered. These interactions are computed using the Lennard-Jones (LJ) potential

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] \tag{1}$$

where $U(r_{ij})$ is the energy of interaction of particle i with j, r_{ij} is the distance between the interaction sites i and j, and σ and ϵ are the LJ parameters. The total potential energy of the system, $U_{potential}$, is the sum of all particle-particle interactions

[1] `https://www.jetbrains.com/pycharm/`

$$U_{potential}(\mathbf{r}^N) = \sum_i \sum_{j>i} U(\mathbf{r}_i, \mathbf{r}_j) \tag{2}$$

where $\mathbf{r}$ is the position vector, and N is the number of particles in the system.

The potential should be truncated such that only interactions within a cutoff distance (*Rcut*) are included. Additionally, analytic tail corrections should be added. Periodic boundary condition (PBC) with minimum image convention (MIC) should be applied. The function should have the LJ parameters (σ and ϵ), total number of molecules (*Npart*), and an array of positions of all particles (r) as an input. The function should return the total energy (*Etot*) of the system. You can use the pseudocode Algorithm 1 as a starting point.

Run your code using the provided configuration file "box.xyz" which contains the coordinates of a liquid CH_4 system using United-Atom TraPPE force field[2]. The molar mass of methane is 16.04 g mol^{-1}. The number of particles is, $N = 362$ and the temperature you need to assume is $T = 150$ K. The box is a cubic box with side lengths of 30 Å. You can visually inspect this file using VMD (or any other molecular visualization software).

Algorithm 1 LJ: Function to compute energy of a system of Npart particles

```
for i from 1 to Npart-1 do
  for j from i+1 to Npart do
    # Calculate distance between particle i and j
    d = ...
    # Correct distance for PBC/MIC
    d <=> Lbox
    # Check distance for the cutoff
    if d < Rcut then
      # Update the energy
      Etot = ...
      # Apply energy correction for the truncation
      Etot = ...
    end if
  end for
end for
```

Question 1.2

Write a function (named "singleParticleEnergy") that returns the energy of one particle in a box. Use the function that is used to compute the total energy as a starting point and modify it appropriately. TIP: Essentially you need to remove the outer **for loop** and avoid self interaction with an additional **if loop**.

Question 1.3

Modify the previous functions to return also the virial pressure. The virial pressure has an ideal gas contribution and a contribution due to the interactions of the particles in the system,

$$P = \rho k_B T - \frac{1}{3V} \langle \sum_{i<j} r_{ij} \left(\frac{dU(r_{ij})}{dr} \right) \rangle \tag{3}$$

where P is the virial pressure, k_B is the Boltzmann constant, ρ is the number density of the system, $V = L_{box}^3$ is the volume of the simulation box and $U(r_{ij})$ is the interaction potential. For particles interacting via the Lennard-Jones potential, find the expression for the virial pressure and add the expression to functions where energy is computed.

[2] `http://chem-siepmann.oit.umn.edu/siepmann/trappe/index.html`

Question 1.4

Write a function (named "translate") that performs translational trial moves. Use the following algorithm:

1. Select randomly a particle in your simulation box.
2. Give the particle a random displacement
   $x_n = x_o + \Delta$
   $y_n = y_o + \Delta$
   $z_n = z_o + \Delta$
   where x_n is the new x coordinate (the same for y and z), x_o is the current x coordinate (the same for y and z). Δ is a uniformly distibuted random number from $[-\Delta x, +\Delta x]$ (and $[-\Delta y, +\Delta y]$, $[-\Delta z, +\Delta z]$ for the respective direction).
3. Check if the new coordinates of the particle are outside of the box and apply periodic boundary conditions accordingly.
4. Compute the energy, U_o, of the particle at the old configuration r_o and the energy, U_n, of the same particle at the new configuration r_n.
5. Compute the difference ΔU between U_o and U_n.
6. Accept the trial move and update the position of the selected particle according to:
   acc (o $\rightarrow$ n) = min (1,exp$[-\beta \Delta U]$), where $\beta = 1/(k_B T)$.
7. Update the positions to r_{old} or r_{new} according to the acceptance rule.

Note that the values for the displacements in each direction are independent from each other. The value of Δ directly affects the number of accepted and rejected moves in MC simulations [3]. If the maximum displacement is too small the move will almost always be accepted, but it will be that $U_o \approx U_n$. This means that the system will not change much. On the other hand, if the maximum displacement is very large it will be that $\beta \times [U_o - U_n] \gg 1$. Thus, the acceptance ratio will be very low and again the system will not change much. The rule of thumb is that a maximum displacement distance is considered good if the rate of acceptance of tranlational moves is 20 to 50 %. At low densities, usually a large fraction moves is accepted, so the maximum displacement should be limited to half the boxsize.

TIP 1: You can start by restricting Δ to be a random number between -0.5 and 0.5 Å. This means that the maximum allowed displacement in any direction is 0.5 Å. By trial and error you can identify the optimum value for Δ.

TIP 2: You can use the random number generator "random" from the package numpy in python to get random numbers when needed.

Question 1.5

Write a function ("startConf") that provides a random starting configuration for your simulation. The function should assign random positions to a number, N_{part}, of particles inside a cubic box with side length L_{box}. Provide a configuration without any particles overlapping. This is achieved by performing MC cycles in which particles are displaced randomly. To achieve a proper initial configuration, it is recommended to perform $N_{init} = 50 \times N_{part}$ translational moves. After assigning random positions (stored in the array r) to all particles, the translational moves implemented in question 1.4 can be utilized to remove overlaps from your system.

To test your function try to recreate the original box given ("box.xyz") by using the following parameters:

1. $N_{part} = 362$
2. $L_{box} = 30$Å

The density of the final box should be $\rho = 358.4$ kg m^{-3}.

[3] see chapter *Monte Carlo Methods* in Allen & Tildesley, Computer Simulation of Liquids.

Question 1.6

Write a function (named "averages") to sample averages. At each MC cycle this function will be called (using the energy and virial pressure of the current configuration of the system) to return the updated average total energy and average total virial pressure. Remember that all moves should be take into account when performing the ensemble average, even the rejected ones.

Question 1.7

Write the full MC program (named "MC-NVT") using the functions you have already created before. The basic skeleton is given in Algorithm 2 below.

Algorithm 2 MC: Main part of MC program

```
# Set or read input parameters
# Call the initialization function
Call initial
# Compute initial total energy and total virial pressure
Call Etot
# set averages to zero
# MC loop
for i from 1 to Ncycle do
  Call TrialMove
  Call Sample
end for
# Write out properties, averages and rate of acceptance of transnational moves
```

Assignment Part 2: Application of the MC-NVT code

Use the TRAPPE force field[4] for CH_4: LJ parameters: $\sigma = 3.73$Å and $\epsilon/k_B = 148$K. In all Questions use a truncated potential with analytic tail corrections.

Question 2.1

Assume a temperature $T = 150$ K and a density $\rho = 358.4$ kg m^{-3}. Assume a cubic box with side lengths $L = 30$ Å and cutoff distance of 14 Å. Run the MC simulation for 500,000 cycles and provide a plot of the rate of acceptance as a function of the different maximum displacements tested. What value would you choose as a maximum displacement for this simulation based on the results?

Question 2.2

Previously you found the maximum displacement for a liquid system. How will the value of the maximum displacement differ when studying gases? Try with methane at $T = 400$ K and $\rho = 9.68$ kg m^{-3}. Assume a cubic box with side lengths $L = 75$ Å and cutoff distance of 30 Å. Provide a plot of the rate of acceptance as a function of the different maximum displacements tested.

Question 2.3

For the liquid CH_4 system studied in Question 2.1, run the MC simulation for 500,000 cycles (using the optimum maximum displacement) and plot the energy of the system as a function of the cycles. How can you establish based on that plot that the system reached equilibrium? How many cycles does it take for the system to reach equilibrium? Furthermore, report the average energy and the error of the energy of the system.

[4] `http://chem-siepmann.oit.umn.edu/siepmann/trappe/index.html`

Assignment Part 3: Compute the P-T diagram of CH_4 using NVT MC

Question 3.1

Compute and report the isotherm (plot of pressure as a function of temperature) of CH_4 at the following densities and temperatures:

- $\rho = 358.4$ kg m^{-3} at $T = 200$, 300 and 400 K
- $\rho = 1.6$ kg m^{-3} at $T = 200$, 300 and 400 K

To create the systems use the function "startConf" (Question 1.5). Note that you have already created the box containing the liquid system earlier in this assignment. For the case of the lower density (1.6 kg m^{-3}) use also $N = 362$ and set the cutoff distance at 50 Å.

The obtained results should be in agreement with the experimentally measured quantities. To check whether your simulation is close to reality, please visit the website of https://webbook.nist.gov/chemistry/fluid/ and choose the right settings to check your result. On this website you will find data for methane derived by Equations of state, which are based on experimental results. Plot your results (P-T curves along with the NIST data). Remember that the quality of the force field (here TraPPE) is very important. You may observe deviations from the NIST data even if your MC code is correct.
TIP 1: Make sure that your system is equilibrated (see Question 2.3) and then start sampling averages.
TIP 2: The time for each calculation depends on the computational power of your PC, the efficiency of the MC implementation and the system size. On a normal laptop one simulation can take up to approximately 1 hour.