Übung: Metropolis Monte Carlo

Assignment due date: 20.5.2019

1. Introduction

The topic is Monte Carlo simulations. The aim is to see how different parameters such as temperature or the number of particles affects properties such as energy. The system is simpler than a protein and is known as a Lennard-Jones fluid. The particles interact with each other only via a Lennard-Jones term. There are no bonds, bond angles or charges, but it can be a good model for some properties of fluids, such as liquid Argon.

2. A first Monte Carlo program

Make a directory to work in. Copy the files from */home/petersen/teaching/sus/3_metropolis_monte_carlo* to this new directory. One of these files is *mc.c*. This is the Metropolis Monte Carlo program for simulating particles in an equilibrium liquid.

Do read of the source code.

Compiling the first MC program

Make sure you are in the directory that contains the file *mc.c*. From the command line, compile the code:

```
gcc -o monte -O2 mc.c -lm
```

Note that at the end of this command, the `-O` uses a letter `O` not the number zero.

Once it is finished, you should have a new file in your directory called *monte*. This can be run with the input file `in_mc` by typing `./monte in_mc` in the command line. When you execute this program, a new file will be written into the directory called *mc.res*. Read the file and check the three columns:

1. step number
2. energy
3. pressure

The input file

The input file is called *in_mc*. It gives the starting conditions for our computer simulations and specifies the name of the output file. Look at the contents of this file to see the starting values. By editing this file, you can control the properties of the system you will study. Be careful when editing this file. Change only the numbers. Do not add spaces around the '=' and do not change the order of the arguments.

There are five lines in the input file specifying the

- density
- number of particles
- temperature
- number of Monte Carlo cycles
- name of the file where the output is written to

Important parts of the mc.c code

Note:

- the random number generation lines of the program: `drand48()` generates a pseudo-random number between 0 and 1. `srand48(10101)` sets the seed for the random number generator to 10101.
- how the random number is used to change the position of a particle.
- the acceptance criterion used to accept or reject a trial move.
- the manner in which macroscopic properties, such as the average internal energy, are calculated.

You can see how the potential energy of a configuration is calculated using a Lennard-Jones potential. The interaction of each particle in the system with all other particles is calculated by the routine `lennardJonesEnergy()`. Interactions of only one particle with all other particles are calculated in the routine called `lennardJonesEnergyOneVsAll()`. You could also try changing the random number generator seed value that is passed to the `srand48()` procedure to see how this changes the results.

Practice running some simulations and changing the starting conditions for each run. Observe how the average energy changes, for example, when you change the temperature of the system. Experiment with changing the length of the simulation and see how this affects the results you obtain.

Input values: Since we are using reduced units in this simulation, keep the input parameters in the following ranges.

Temperature	Between 0.5 and 1.5
Density	Between 0.7 and 1.1
$N_{particles}$	Less than 1 000.

3. Viewing graphs of your output

In order to have an idea of how the internal energy changes over the course of your simulation, you can graph the output file that is created as the program runs. If you like to use *gnuplot*, type `gnuplot` and then `plot "mc.res" us 1:2` which will result in a plot of the energy versus time (energy will be the *y*-axis and time will be the *x*-axis).

4. Assignment

Answer the following questions in English or German. You will need to run many different simulations with different starting conditions to answer these questions. Reports should not exceed five pages in length. Please remember one of the points from the lectures. Monte Carlo simulations are only valid when the system is at equilibrium, so the first part of a simulation is often ignored or thrown away.

When making graphs, use whatever program you like, but axes must have labels and units.

Part I

How does the average energy vary when you change the length of the simulation? What does this suggest about the length of time you need to get reliable results ?

1. Examine how the instantaneous energy changes over the course of a simulation of 10 000 steps.

Describe what happens to the energy. Are there any trends? Any unusual features? Describe, in detail, any anomalies/trends that you notice.

Part II

Run a simulation using the pre-compiled program called *monte_1*. This program is identical to the program you have been examining. The only difference is that there is no input file. All of the input variables for the simulation are hard-coded into the program.

After running the simulation, look at the output file *mc_1.res* and try to determine what the initial conditions of the simulation could be. Report what you think the initial values of the temperature, density and number of particles were. Explain how you deduced these values. You may add any comments you think are relevant to explain what is happening in the system modelled by the program *monte_1*.

For this part of the Übung you will probably have to run many simulations and observe how the output changes to determine the initial conditions of the simulation. Your mission is to find parameters for the program `monte` which give the same temperature and pressure as for the system in `monte_1`.

Part III

Look at the source code contained in the file *mc_2.c*. There is a small difference to the code contained in *mc.c*. You can find this difference by using `diff` command as in `diff mc.c mc_2.c`.

Compile *mc_2.c* and run it. Do the results seem different to results you obtain from running the executable you compiled with *mc.c* ? Ensure that you are using the same input file when running each program. That way, you can directly compare the results between the two programs.

What is the difference between the two programs? This difference causes the program *mc_2.c* to be "broken".

Why does this affect the simulation results? Describe what the difference in *mc_2.c* causes? Discuss this in as much detail as possible.