

Übung: Monte Carlo Methods

SS 2009 Übung zu Struktur und Simulation

27-Apr-09

1. Timeplan.....	1
2. Introduction	1
3. A first Monte Carlo program.....	2
3.1. Compiling the first MC program.....	2
3.2. The input file	2
3.2.1. Important parts of the mc.cpp code	3
4. Viewing graphs of your output.....	4
5. Assignment.....	4
5.1. Part I	4
5.2. Part II.....	5
5.3. Part III.....	5

1. Timeplan

- 28 April week 1
- 5 May week 2
- 12 May report due

2. 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. It is not a bad model for an inert gas such as argon.

Originally written by Dr Emil Mittag

3. A first Monte Carlo program

Make a directory in your home directory to work in. Copy the files from */home/torda/uebung_mc* to this new directory. One of these files is *mc.cpp*. This is the simple Metropolis Monte Carlo program for simulating particles in an equilibrium liquid.

Do have a read of the source code.

3.1. Compiling the first MC program

Make sure you are in the directory that contains the file *mc.cpp*. From the command line, issue the following command to compile the code so it can be executed:

```
g++ -o monte -O3 mc.cpp
```

Note that at the end of this command, the `-O3` 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 executed by typing `monte` from the command line. Run the program and look at the screen output. Additionally, 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

3.2. The input file

The input file is called *in_mc*. It gives the starting conditions for our computer simulations. 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, however, as spacing of the values is important. For example, if you change:

```
1.0      Temperature  to      1.0Temperature
```

either the code will not run or it will not behave as expected.

There are five lines in the input file specifying the

- density
- number of particles
- temperature
- number of Monte Carlo cycles
- type of averaging of phase properties

3.2.1. Important parts of the mc.cpp code

Note :

- The random number generation lines of the program. `drand48 ()` generates a pseudo-random number between zero and one. `srand48 (10101)` sets the seed for this random number generator. The seed is 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 may also spend some time looking at how the potential energy of a configuration is calculated. The potential energy is calculated using a Lennard-Jones potential. The interaction of each particle in the system with all other particles is calculated by the routine called `calc_interact ()`. Interactions of only one particle with all other particles are calculated in the routine called `calc_i ()`. 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. Observe what happens when you change the averaging type between instantaneous (no averaging) to time averaging (the average over the whole time the simulation has been run).

Notes about input values: Since we are using reduced units in this simulation, you should 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. The program will crash if $N > 1\ 000$

4. Viewing graphs of your output

In order to have an idea of how the internal energy is changing over the course of your simulation, you can graph the output file that is created as the program runs. To do this for the *monte* program, from the command line type `gnuplot` and then type `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). You can use the plots of the output to help you with the assignment. If you wish, you may include plots generated by `gnuplot` in your report.

5. Assignment

Answer the following questions. Your answers may be in English or German. From n students, there should be n different reports. You may need to run many different simulations with different starting conditions to answer these questions correctly. 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.

5.1. 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 (ten thousand) steps. Describe what happens to the energy. Are there any trends? Any unusual features? Describe, in detail, any anomalies/trends that you notice.

2. What happens when you vary the frequency with which the sampling of phase properties is calculated ? As written, the program calculates and stores things like the internal energy every time a move is attempted. Change how often these properties are stored and explain what you observe happening to the results of a simulation. Based on what you observe, do you think it is better to calculate averages at every step of a simulation or to increase the interval that is used between calculation of such averages? Why?

5.2. 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 (i.e. they are already set inside the program and don't need to be read from the file). You will not know what these starting values are.

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. Note that it is not possible to estimate the number of particles used in the simulation by timing. The precompiled program will run at a different speed than the program for which you have the source code.

5.3. Part III

Look at the source code contained in the file *mc_2.cpp*. There is a small difference to the code contained in *mc.cpp*. You can find this difference by using the Linux command `diff`. To use this command to compare *mc.cpp* and *mc_2.cpp* type `diff mc.cpp mc_2.cpp`.

Compile *mc_2.cpp* and run it. Do the results seem different to results you obtain from running the executable you compiled with *mc.cpp*? 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.cpp* to be “broken”. How do you think this difference could affect the simulation results? Can you describe what the difference in *mc_2.cpp* causes? Discuss this in as much detail as possible.