Programming projects:

You are free to use any programming language but you are encouraged to use C or C++, because that will make it easier for us to help you. There are many different programs and programming languages you could use for data analysis (C/C++, gnuplot, python, shell scripts, matlab, combinations of all of those, etc.). If possible, pick something you are at least a bit familiar with.

The basic requirements of the programming projects are stated below, for higher grades consult the assessment criteria. There are two projects, one Molecular Dynamics and one Monte-Carlo. The projects are split up into smaller sections, each of which is to be presented seperately in a written report of one to two pages which includes graphs.

The reports should contain a comprehensible description of the method and analysis of the simulation. Do not just copy-paste the ingredients for your simulation from the book, but discuss what you have done and your results. Describe the steps you have gone through to verify that your program does what it is supposed to. If there is no specific instruction to plot something, find something useful to plot to demonstrate that you have done what the assignment instructed. The source code should be written from scratch and should be appended to the report and well-documented. These projects are individual.

What to do if you get stuck

- Consult the list of programming tips and common bugs on the course website. These lists are based on past performance of you fellow students. Follow the tips and try to eliminate the common bugs listed.
- Be systematic about debugging. Keep in mind that if everything did what you think it does, your program would be working, so find ways to check everything independently.
- Carefully document everything that is going wrong.
- Don't hesitate to come to the office hours to ask for help. Do not allow yourself to remain stuck for too long.

1 Molecular dynamics simulations of a classical liquid

The first project is to simulate a classical liquid consisting of Argon atoms. Your aim is to reproduce the simulation from 1964 by Rahman in Phys. Rev. 136, A405. You should calculate the total, kinetic and potential energies and their fluctuations, the heat capacity and the radial distribution function. You are encouraged to calculate other properties or try out other ensembles.

1.1 Planning

- Read the description all the way through before starting. This will save you time later on and help with the planning, because it will give you an idea of all the ingredients you will need.
- Sketch a structure of the MD simulations program and discuss with the course assistants.
- Create one or more source code files with subroutines in them that you are going to use, but leave them empty for now. You will fill these in as you go along.

1.2 Lennard-Jones

- Implement functions for the LJ potential energy and forces.
- Print them out and verify that you are calculating what you think you are.
- Implement the routine that calculates the total force acting on each particle and the total potential energy of the system.
- Remember that later you will need to include periodic boundary conditions.
- check a few configurations to make sure that it works.

1.3 Integration Algorithm

- Choose an integration algorithm and implement it in your code, for example Velocity Verlet.
- Start from an initial condition with only two atoms on a collision course. Print their position, velocity, kinetic energy, and potential energy and plot them.
- Vary the time step and check that the total energy is conserved properly.

1.4 Periodic boundary conditions

- Describe how you want to implement periodic boundary conditions and do it. You can, for instance, use the nearest integer function.
- Check that the boundary conditions work properly, by using the setup from 1.3 and verifying that the particles collide, move through the boundaries, and then collide again repeatedly.
- Plot the trajectories and make sure they are physical.
- Do this in x, y, and z directions.

1.5 Initial conditions

- Describe what initial conditions you plan to use. Explain your choice.
- Select a random number generator and write the function that generates the initial conditions.
- Print out the initial values of positions and velocities, plot them, and check that they are how you wanted them to be.

1.6 Radial distribution function

- Describe the physical meaning of the radial distrution function (RDF). Explain why it is a useful quantity to calculate.
- Implement binning of distances.
- Calculate and plot instantaneous RDFs at $t = 0$ and a number of different points in time. If your statistics are poor, try using a larger number of particles.

1.7 Equilibration

- Run your system with a large number of particles for at least 10 ps.
- Confirm equilibration by plotting some relevant quantities. Motivate your choices.

1.8 Averaging

- Run a longer simulation (at least 20 ps) and dump coordinates and other information at regular intervals.
- Implement block averaging routines for data analysis (in a seperate program). This is described in Frenkel & Smit.
- Estimate correlation times and determine a reasonable block length.

- Apply this to the kinetic energy, and check that the results are sensible. Include an analysis of the effect of the block length.

1.9 Results

- Obtain expressions for the quantities you want to calculate in terms of averages. These are discussed in Frenkel & Smit.
- Calculate averages, variances, etc. and obtain the kinetic energy, heat capacity, and RDF with error bars. Include a statistical analysis.
- Compare your results to the literature. If there is a significant discrepancy, fix your program.

1.10 LAMMPS

- Install LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator, `http://lammps.sandia.gov/`). This is one of the most common MD packages. You can find documentation on `http://lammps.sandia.gov/doc/Manual.html`, but google will usually also quickly find the right page or command for you. LAMMPS is very powerful, but it can be a bit tricky to get it to behave as you want. The documentation is sometimes hard to read and a small mistake can have bizarre consequences. Furtunately, LAMMPS gives relatively informative error messages. Pay attention to them. Since LAMMPS has a large user base, there are many people who have reported the same error messages on support email lists and such. Searching the internet can help you find a solution quickly.
- The LAMMPS documentation contains an example called "melt". If you cannot find it, contact me, and I will get you a copy of the files. Use the melt example to run a simulation with lammps. Uncomment the lines for making images/movies and have a look.
- This example simulates a melting LJ crystal. The simulation is very similar to the MD simulation you have just written from scratch. Tweak the parameters for the LJ force field in LAMMPS to make them the same. Note that LAMMPS has only a small number of sets of units that it likes you to use.
- Get LAMMPS to periodically output the kinetic energy and some other quantities you like. (hint: `thermo_style` and `thermo`). LAMMPS can even calculate the RDF for you (`compute rdf`).

1.11 Thermostatting (with LAMMPS or in your own code)

This subassignment is to implement thermostatting. You can do this either with LAMMPS or in your own code. LAMMPS makes implementation of thermostats very easy. They are just one-line commands. However, implementing a Langevin thermostat in your own code is not very complicated either.

1.11.1 Option 1: LAMMPS

- Read the documentation carefully and implement both Nosé-Hoover and Langevin thermostats. (Hint: use `fix nvt` for Nosé-Hoover and `fix langevin` for the Langevin thermostat.)
- Check that you get the correct temperature and distribution of velocities.
- Run simulations in LAMMPS using NVE and NVT with Langevin or Nosé-Hoover and use your existing analysis routines to calculate the heat capacity in all three cases with error bars. Compare to your previous results and the literature.

1.11.2 Option 2: Your own code

- Add a damping term and a random noise term to your force calculation.
- Make sure that if you have an MD algorithm that needs you to calculate the forces several times during a time step, that the results will be consistent.
- Check that you obtain the correct temperature and distribution of velocities.
- Run a simulation in the NVT ensemble using your new thermostat, and calulate the heat capacity with error bars. Compare to your NVE result and the literature.

2 Monte-Carlo simulation

2.1 Getting it running

- Sketch the structure of the MC simulation program. You can simply substitute the molecular dynamics integrator for the Monte Carlo algorithm and use the same program structure and analysis routines as in the MD project, but remember that time is not defined in Monte Carlo. (If you keep it in mind when writing the molecular dynamics program above, it should be easily modified to perform a Monte Carlo simulation instead of a molecular dynamics simulation.)

- Choose a suitable trial move or set of trial moves and write a subroutine for generating trial moves.

- Implement the acceptance step of the MC algorithm. Optimise the trial move so that you have a good acceptance ratio.

2.2 Results

- Calculate the same static quantities as you did with the MD simulation and make a comparison.

- Compare efficiency between the MD and MC simulations.

3 Bonus: MD nonequilibrium simulation

- Take a simulation box with a lot of particles and increase the temperature in one half of the box by *instantaneously* increasing the velocities of all the atoms in that half.
- Run a simulation and calculate temperature profiles in the box. In LAMMPS this can be done with `fix ave/spatial`, and in your own code you can add another binning routine similar to the RDF calculations. This would be done, for instance, to numerically obtain the heat conductivity.
- Study the effect of thermostatting on how the temperature profile develops. What does this mean for any attempt to calculate the heat conductivity?