Introduction to molecular dynamics

Jane R. Allison

Preamble

In this workshop, you will work with a simple program for carrying out a molecular dynamics simulation of "Lennard-Jonesium". The interactions between the particles in the simulation are governed only by the Lennard-Jones potential:

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

The code uses 'dimensionless' or 'reduced' units, defined in terms of the parameters of the Lennard-Jones potential and the mass (Table 1). The advantages of using dimensionless units are that we can work with numerical values of the order unity, rather than the typically very small values associated with atomic-scale systems, and that the equations of motion are simplified.

As it is written, the program is only for one dimension; you will have the option later of extending it to two or three dimensions. The layout and content of the program are based on Algorithms 3 - 6 of Chapter 4 of the book "Understanding Molecular Simulation"[1]. You have been provided with a scanned pdf of (most of) this chapter as background reading and reference material - you should refer to the descriptions of the algorithms and the statistical physics underlying them to help you understand what the code is doing.

Table 1: Dimensionless units

Property		Reduced Form
Distance	$r* =$	r/σ
Time	$t* =$	$t(\epsilon/m\sigma^2)^{1/2}$
Temperature	$T* =$	k_BT/ϵ
Force	$f* =$	$f\sigma/\epsilon$
Energy	$E* =$	E/ϵ
Pressure	$P* =$	$P\sigma^3/\epsilon$
Density	$\rho* =$	$\sigma^3\rho/m$

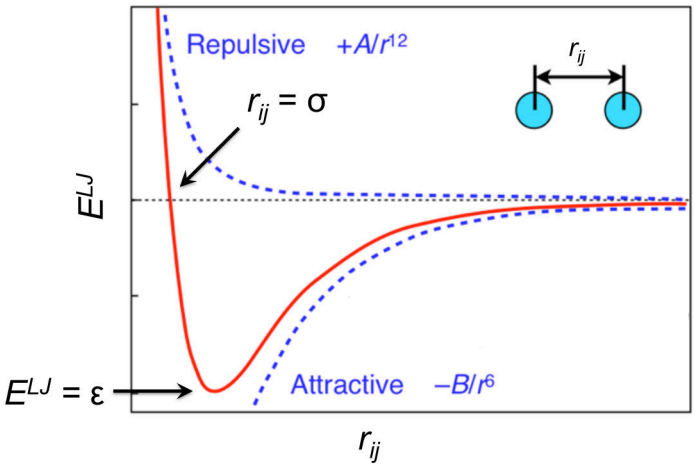

Figure 1: The Lennard-Jones potential between two particles separated by distance r_{ij}.

Instructions

Go to `https://github.com/arohl/MolSim2017`, click the green 'Clone or download' button, choose 'Download ZIP', and save the zip archive to an appropriate location on your computer. Extract its contents by typing:

```
unzip MolSim2017-master.zip
```

This should give you a directory `MolSim2017-master`, containing two subdirectories: `Workshop1` and `Workshop2`. The `Workshop1` directory contains the files for today's workshop, and `Workshop2` the files for tomorrow's workshop. Within `Workshop1`, you should have the program source code (`1dmd.py`), a pdf version of the code, and a subdirectory `example` which contains example run settings and output.

Open the file `1dmd.py` in your favourite plain text editor (e.g. Emacs, vim, gedit, TextEdit, Xcode). Make sure you have syntax highlighting turned on.

Start by reading through the program code. Do you understand what each section of the program is doing? Refer to the algorithms and background information in Chapter 4 to help you figure out what is being calculated. Discuss with your neighbour and/or one of the workshop teachers. You don't need to understand every detail, but you should have a reasonable idea what is going on. If necessary, make notes on your hard copy of the code.

To test whether the program is running correctly on your machine:

1. Check that the input parameters (specified at the beginning of the program) match those listed in the file `settings` in the `example` directory.
2. Check that the program is executable by typing: `ls -l 1dmd.py` If the left-most part of what you see does not look like this:

   `-rwxr--r--@`

   then type:

   `chmod 744 1dmd.py`

   and check that it worked by typing `ls -l 1dmd.py` again.
3. Run the program by typing: `./1dmd.py` You should see a series of messages printed to screen tracking the progress of the program. When it has finished running, there should be three new output files: `coords.xyz`, `energy.dat`, and `temperature.dat`.
4. Compare the output files to those given in the `example` directory. You may find the plotting and visualisation instructions below useful.

To plot the time-series data written to the output files `energy.dat` and `temperature.dat`, run the plotting program `plot.gnu` by typing:
`gnuplot plot.gnu`

To view a movie of the Lennard-Jonesium using VMD, either open VMD, and load the coordinate file `coords.xyz`, or type `vmd coords.xyz`. You may wish to change the style and size at which the particles are drawn.

Assuming the program is running correctly, and that you understand (roughly) what it is doing, you can now go ahead and try to answer some/all of the questions below. If you have a lot of coding experience or prior knowledge of MD, you may wish to move fairly quickly down to the "Expert" exercises.

Questions and Exercises

Basic

- Without changing the parameter values, examine how the total energy and temperature behave during the simulation. The time-series of these, followed by the average and standard deviation, are printed to the files `energy.dat` and `temperature.dat`. Is this what you expected to see?
- What happens to the energy if you increase or decrease the integration time step? Why?

- How can we know how long we should perform a simulation for? Do you think the simulation as it is currently set up runs for long enough? Try to find out.
- The behaviour of Lennard-Jonesium will depend on the temperature and density (in this case, the ratio of number of the particles to box length). What are the limits for these parameters? How does the behaviour change as you vary these parameters?
- Explain what the term "nearest image" means, and how the algorithm to compute this works.
- If you were simulating a real substance, how would you decide how many particles (atoms or molecules) to include and how large the simulation box should be? Note that your answer will differ depending on whether you are considering a simulation of e.g. a solvent or e.g. a protein in water.

Expert

- Extend the code to work for two or three dimensions. Don't forget to modify the coordinate writing function so you can watch a movie of your simulation!
- Improve the velocity selection algorithm so that the velocities are selected from a Gaussian rather than uniform distribution. Explain why this is likely to be a better option.
- Implement the Euler, Leap Frog or Velocity Verlet integration algorithm(s). Are these better or worse than the Verlet algorithm?
- Write a function to calculate the radial distribution function g(r) - see Algorithm 7. How does this vary depending on the input parameters for your simulation?
- At present, the simulation samples from an NVE ensemble. Implement a thermostat to keep the temperature constant (NVT ensemble). If that was easy, implement a barostat (pressure, NpT) too! Which ensemble do you think is most relevant to studying real life phenomena?
- Add a function to calculate Coulombic interactions between atoms. Note that you will also need a way of specifying the (partial or full) charges on atoms.

References

[1] Frenkel, D. & Smit, B. *Understanding Molecular Simulation* Academic Press, San Diego, 2nd edition, 2002.