Exercises for the lecture

Fundamentals of Simulation Methods

WS 2016/17

Lecturers: Frauke Gräter and Rüdiger Pakmor

Exercise sheet 7 ***(due date: Dec 8, 2016, 11:59pm)***

Molecular Dynamics: Integrators, RDF's

A simple molecular dynamics code (50 points)

(Note that for this exercise, the speed advantage of C is an asset. You may try with Python as well, but be prepared for significant wait times.)

In this exercise, we construct a simple molecular dynamics code, using first the microcanonical ensemble in which the system is closed and its total energy stays constant (also called "NVE": constant number, volume, and energy). We want to simulate a simple system of $N = (N_{1\mathrm{d}})^3$ argon atoms, interacting with a Lennard-Jones potential,

$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right], \tag{1}$$

meaning that the total potential energy is given as

$$E_{\mathrm{pot}} = \frac{1}{2} \sum_{i,j}^{N} V(|\mathbf{r}_i - \mathbf{r}_j|). \tag{2}$$

For argon, we will use the parameters

$$\sigma = 3.4 \times 10^{-10}\,\mathrm{m}, \tag{3}$$
$$\epsilon = k_{\mathrm{B}} \times 120\,\mathrm{K} = 1.65 \times 10^{-21}\,\mathrm{J}, \tag{4}$$
$$m = 6.69 \times 10^{-26}\,\mathrm{kg}, \tag{5}$$

where σ and ϵ characterize the potential, and m is the mass of each atom. Write a computer code (based on the uploaded C template) that integrates the equations of motion of N argon atoms, placed into a cubical box of side-length L with periodic boundary conditions. The program prints out the positions after every 100 time steps in `.xyz` format, which is possible to open with, e.g., VMD (available for Windows, Linux, or Macs). You can check what your program is doing by visualizing your `.xyz` trajectory (in VMD, choose the "VDW" mode).

Proceed along the following steps:

(a) In your code, you will express all length units in terms of σ, all energies in terms of ϵ, and all masses in terms of m. In other words, introduce dimensionless distances

$$\mathbf{r}' = \frac{\mathbf{r}}{\sigma}, \tag{6}$$

dimensionless energies $E' = E/\epsilon$, etc., and rewrite all relevant equations in terms of the dimensionless quantities. What is a suitable quantity to scale the velocities?

(b) Write a function that sets up N_{1d} particles per dimension on a regular grid in a periodic box of size L. We adopt a mean particle spacing $\overline{d} = 5.0\,\sigma$ (implying that $L' = 5\,N_{1d}$ in the scaled length units). For the initial velocities, assume that we prescribe a certain kinetic temperature T in Kelvin, from which we can compute a one-dimensional velocity dispersion as

$$\sigma = \sqrt{\frac{k_B T}{m}}. \tag{7}$$

Scale this velocity dispersion to internal dimensionless units, yielding σ'. Now draw three random numbers $(\tilde{v}_x, \tilde{v}_y, \tilde{v}_z)$ for every atom from a Gaussian distribution with zero mean and a dispersion of unity, and scale them with σ' to get the initial velocities $\mathbf{v}' = \sigma'\tilde{\mathbf{v}}$. This means your initial velocities will then correspond to a Maxwellian with temperature T. Note: If you only have a random number generator that produces uniform random numbers in the interval $]0, 1[$, you can produce a Gaussian distributed number g by drawing two random numbers u_1, u_2 from $]0, 1[$ and transforming them as $g = \sqrt{-2\log(u_1)}\cos(2\pi u_2)$.

(The positions are initialized in the code and the Gaussian-distributed random variables are also implemented. You will need to give the correct scaling for the velocities.)

(c) Now write a function that calculates the acceleration $\mathbf{a}(t)$ of each particle, in the dimensionless units used by your code. For simplicity, sum over all distinct other particles in the box and always consider the nearest periodic image for each pair. Use a (quite large) cut-off radius for the potential equal to $r_{cut} = 10.0\,\sigma$, i.e. set the potential to zero for distances larger than r_{cut}.

(d) Prepare a function that 'kicks' the particles with their stored accelerations for a given time interval Δt. Also, produce a function that 'drifts' the particles with constant velocity over a given time interval Δt. After the particles have been moved, map them back periodically into the principal box in case they have left it. First, implement Euler integration:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t) \cdot \Delta t \tag{8}$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \mathbf{a}(t) \cdot \Delta t \tag{9}$$

and then also the Velocity Verlet in another version of the code:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t) \cdot \Delta t + \frac{1}{2} \cdot \mathbf{a}(t) \cdot \Delta t^2 \tag{10}$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \frac{1}{2}(\mathbf{a}(t) + \mathbf{a}(t + \Delta t)) \cdot \Delta t. \tag{11}$$

(e) In the acceleration calculation routine, add a computation of the total potential seen by each particle due to its neighbors (complete the routine `calc_forces`). The template already takes care of the calculation of the energies and the output of the mean kinetic energy per particle, mean potential energy per particle, and kinetic temperature to a file.

(f) Run your code with a timestep $\Delta t' = 0.01$ in internal units (corresponding to $\Delta t = \sigma(m/\epsilon)^{1/2}\Delta t'$) for 100000 steps using $N_{1d} = 8$ (i.e. $N = 512$ atoms) and $T_{init} = 80\,\mathrm{K}$. Confirm that the total energy is conserved well. Is there any major difference between Velocity Verlet and Euler integration in this regard? Does the kinetic temperature change compared to T_{init} (does the system warm up or cool down)? Why? (Hint: this calculation can take up to an hour, so you could run fewer time steps or less particles for testing purposes, or if it is very slow, also in your final report.)

(g) Our code determines and prints the radial distribution function (RDF) $g(r)$, which corresponds to the probability density that two particles are within r of each other. We compute this on a grid between $0..L/2$ and store the frequency of all the observed distances during all the time steps (without double counting). By convention, we normalize $g(r) = 1$ for an ideal, homogeneous gas, without any volume exclusion. Another way of checking whether the normalization is correct is that $g(r) = 1$ holds for very long r. Complete the normalization function in the code (right at the end of the program), and plot $g(r)$ for the starting temperatures $T = 200K$ and $T = 1K$. Which state(s) of matter do they correspond to?