Problem Set 9

Exercises for course Fundamentals of Simulation Methods, WS 2021

Prof. Dr. Mario Flock, Prof. Dr. Friedrich Röpke
Tutors: Brooke Polak (brooke.polak@uni-heidelberg.de | ITA), Glen Hunter (glen.hunter@uni-heidelberg.de | ITA), Jan Henneco (jan.henneco@h-its.org | HITS)
Offices: HITS: Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg; ITA: Albert-Ueberle-Strasse 2, 69120 Heidelberg

Hand in until Wednesday, 12.01, 23:59
Tutorials times: 13-14.01.
Group 1: Brooke | Thursday 11:00 - 13:00
Group 2: Glen | Thursday 14:00 - 16:00
Group 3: Jan | Friday 11:00 - 13:00

1) A simple molecular dynamics code

In this exercise, we construct a simple molecular dynamics code, using first the micro-canonical ensemble in which the system is closed and its total energy stays constant. 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

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

For argon, we will use the parameters

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

where σ and ϵ characterize the potential, and m is the mass of each atom. Write a computer code that integrates the equations of motion of N argon atoms, placed into a cubical box of side-length L with periodic boundary conditions.

NOTE: To simplify the coding work, you can use the C-template provided. For this exercise, the speed advantage of C is an asset and we strongly advise doing this exercise with C. You can use Python, but be prepared for significantly long run times. The template has been made such that required knowledge of C-syntax is minimal, and you only have to fill in the physical calculations for each function. The file compile.txt contains the necessary compile and run commands. You can then use the outputted data file to load and analyze your results with python. If you don't have a compiler installed, here are some recommendations: Mac use MacPorts to install gcc, Windows use MinGW, and Linux follow the instructions given here. Learning to compile programs is a *very* important skill to have as a computational scientist, so don't give up if you run into some issues at first! It's all part of the learning process, and your tutors are here to help :)

Proceed along the following steps:

(a) In your code, 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

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

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 $N_{\rm 1d}$ 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 $L' = 5\,N_{\rm 1d}$ 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

$$\sigma = \sqrt{\frac{k_{\rm B}T}{m}}. \tag{7}$$

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 $\vec{v}' = \sigma'\tilde{\vec{v}}$. 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 $g = \sqrt{-2\log(u_1)}\cos(2\pi u_2)$.

(c) Now write a function that calculates the acceleration $\vec{a}_i'$ 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 $r_{\rm cut} = 10.0\,\sigma$, i.e. set the potential to zero for distances larger than $r_{\rm cut}$.

(d) Use the Leapfrog time integration scheme to advance the particles. To this end, 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.

(e) Now, write a driver routine that first initializes the particles, and then calculates the accelerations once at the beginning. Add a loop over N_{steps} that first kicks the particles by half a step, then drifts them by the full step, followed by a new force calculation. Finally, complete the step by again kicking the particles by half a step.

(f) In the force calculation routine, add a computation of the total potential seen by each particle due to its neighbors. Also, write a routine that computes the total kinetic energy and total potential energy of the system, as well as the instantaneous kinetic temperature. Call this function whenever a full timestep has been completed, and output the mean kinetic energy per particle, mean potential energy per particle, and kinetic temperature to a file.

(g) 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 60000 steps using $N_{1\text{d}} = 8$ (i.e. $N = 512$ atoms) and $T_{\text{init}} = 80\,\text{K}$. Confirm that the total energy is conserved well.

(h) Now we want to ensure that the system maintains a temperature equal to a prescribed temperature T, meaning that we deviate from the microcanonical ensemble and rather seek a coarse approximation of a canonical ensemble. To this end, add a function that scales the velocities such that the instantaneous kinetic energy corresponds to the imposed value. Call this scaling function every 100-th step in your timestep loop.

(i) Run your molecular dynamics simulations for the target temperature 80 K, and also for a temperature 70 K. Estimate from the results *for the last 10000 steps* (to reduce the influence of the initial transient phase) the molar heat capacity at fixed volume, C_v, at the temperature $T \simeq 75\,\text{K}$ (and the given density). Also, do a similar exercise to estimate the heat capacity at a temperature of $T \simeq 400\,\text{K}$. Compare both results with the heat capacity of $C_v = \frac{3}{2}R$ expected for a monoatomic ideal gas, where R is the gas constant. Interprete your result.

(j) Finally, carry out a MD simulation where you set the imposed temperature to $T = 30\,\text{K}$. Make a plot that shows the mean kinetic and mean potential energies per particle as a function of time, as well as the total mean energy. Interprete your result.