APiE exercise – Molecular Dynamics of fluids (0.5 EC)

Write a molecular dynamics (MD) program to simulate the motion of a collection of identical particles in *two* dimensions.

To simplify the set-up and running of the simulations, we will use a very soft potential to describe the interaction. For two particles i and j, located at the 2D positions $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively, their interaction energy as a function of the distance $r_{ij} = |\mathbf{x}_i - \mathbf{x}_j|$ reads as

$$\phi(r_{ij}) = \begin{cases} \epsilon\,(\sigma - r_{ij})^2 & \text{for} \quad r_{ij} \leq \sigma, \\ \\ 0 & \text{for} \quad r_{ij} > \sigma. \end{cases} \tag{1}$$

The total potential energy of the system is obtained by a summation over all particle pairs,

$$\Phi = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \phi(r_{ij}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \phi(r_{ij}), \tag{2}$$

where the factor of one-half appears to correct for double counting, and where self-interactions ($j = i$) are excluded.

As we have seen in the first ODE exercise, the leap-frog version of the Verlet algorithm is the method of choice to integrate the Newtonian equations of motion of the particles,

$$\dot{\mathbf{x}}_i = \mathbf{v}_i \tag{3}$$

$$m\dot{\mathbf{v}}_i = \mathbf{f}_i \tag{4}$$

where $\mathbf{v}_i$ denotes the velocity of particle i. The total force $\mathbf{f}_i$ on this particle is obtain by

$$\mathbf{f}_i = -\frac{\partial \Phi}{\partial \mathbf{x}_i} = -\sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{d\phi(r_{ij})}{dr_{ij}} \frac{\partial r_{ij}}{\partial \mathbf{x}_i}, \tag{5}$$

where the chain rule was used in the last step.

The *initial configuration* of the particles can be generated by randomly placing the particles in the box (as the selected potential is soft enough to permit this option) or by placing the particles on a regular lattice (which will subsequently melt).

Since the interactions between your particles are purely repulsive, the initial configuration will gradually expand into space. To keep the particles together, in a system of constant density, we have two options:

- The simplest option is to surround your simulation box with four *walls*. Use a quadratic potential like in Eq. (1) to model the particle-wall interaction, with r_{ij} denoting penetration of the particle i into the j^{th} wall.[1]
- Using *periodic boundary conditions* (PBC), i.e. surrounding the simulation box with virtual copies of itself. Note that every particle i now interacts with the 'nearest image' of every other particle j. This will earn you a bonus.

[1] Make sure that the potential keeps increasing as the particle goes deeper into the wall, or you will see your particles escape to infinity.

We want to simulate a system of $N = 100$ particles; this will prove sufficiently large to obtain nice averages within $10^3 - 10^4$ simulation steps. Select for the radius σ and the mass m numerical values of the order 1, and for the interaction strength ϵ a value of the order 100.

Hint: As you probably have noticed, explicit `for`-loops are much slower than implicit loops (vector and matrix manipulations) in matlab. To calculate the total potential energy, Eq. (2), or the total forces on all particles, Eq. (5), requires a double loop. Since this can become *very* slow, have a look at the following three lines:

$$\begin{aligned} \texttt{r} &= \texttt{box} * (\texttt{rand}(\texttt{N}, 2) - 0.5); \\ [\texttt{i}, \texttt{j}] &= \texttt{meshgrid}(1 : \texttt{N}, 1 : \texttt{N}); \\ \texttt{rij}(:, :, 1) &= \texttt{reshape}(\texttt{r}(\texttt{i}, 1) - \texttt{r}(\texttt{j}, 1), \texttt{N}, \texttt{N}); \end{aligned}$$

Make sure that you understand what these three lines achieve – using this technique will make the simulations much quicker. We conclude that the simulation code requires *only one* explicit loop, namely

$$\texttt{for step} = 1 : \texttt{steps}$$

Likewise, the parameters and distributions mentioned in the questions can *all* be calculated with a code devoid of explicit double loops.[2]

About the *grading* of your report: as usual, answering the standard question satisfactorily will earn you an eight. A maximum of two extra points can be earned by trying your skills on the items marked with 'bonus'.

Questions

a. Derive analytical expressions for $d\phi/dr_{ij}$ and $\partial r_{ij}/\partial \mathbf{x}_i$.

b. How to efficiently calculate all forces?

- Can you calculate the strengths $f_{ij} = -\partial\phi(r_{ij})/\partial r_{ij}$ of all pair forces (ignoring the cut-off distance for the moment) without using an explicit loop?
- Can you truncate all forces f_{ij} at the truncation distance σ without using an explicit loop and without using an `if` statement?[3]
- Can you now calculate all force vectors $\mathbf{f}_{ij}$ without using an explicit loop?
- Finally, add up the pair forces $\mathbf{f}_{ij}$ to obtain the total forces $\mathbf{f}_i$ acting on the particles.

c. Add wall forces or use periodic boundary conditions (bonus point) to confine the particles to a square area measuring $L \times L$, with L a box length of the order 10. Again, there is no need for explicit loops, nor for `if`-s.

d. Combine the force calculation with the Verlet leap-frog algorithm to numerically integrate the equations of motion of the particles. The `scatter` function makes nice plots that can be used to see the particles flying around in real time. If you use PBCs, you may want to correct the particle positions after every step to keep them in the central box.

[2] Note that using an explicit loop to repeatedly call a function that itself contains an explicit loop is the same as coding an explicit double loop.

[3] Alternatively, you may switch the order and perform a distance truncation step before the pair-force calculation step.

e. Classical mechanics gives rise to conservation laws. Which conservation law(s) should be obeyed by your system? Use these to test (and debug) your algorithm, and to determine an acceptable integration time step.

In answering the below questions, please note that 'temperature', 'pressure', 'distributions' etc refer to *averages* over a number of simulation steps; the values deduced from a single 'frame' will be very noisy. Your values will improve if you ignore the first part of the simulation, where the system is still in transition from the random start configuration to an equilibrated state; the second part of the run is usually called the 'production phase.'

f. Run several simulations at varying densities. Calculate for each system the temperature and pressure, using the expressions provided on slides 37 and 38 of the handouts (Since the system is periodic, the pressure expression should be based on pairs of particles i and j). You may use $k_B = 1$ for convenience. Compare your results with the 2D ideal gas law, $P = Nk_BT/L^2$. Can you explain the similarities and/or differences?

g. Make a histogram of the velocity distribution, and compare your result against the theoretical Maxwell-Boltzmann distribution (slide 41).

h. For a bonus, calculate the radial distribution function $g(r)$ (slide 40) and/or the self-diffusion coefficient D (slide 39) of the particles in your system.