Event-driven molecular dynamics

Goal: Learn simulating particles that interact via hard-core potentials.

So far in our MD simulations, we considered particles interacting via a spatially continuous potential. In contrast, event-driven MD can be used to simulate particles interacting via hard-core potentials by simply considering binary collisions (no lasting contacts). More concretely, assuming no external forces, the particle trajectories can be analytically calculated, *i.e.* they follow a straight line between collisions[1].

The time t_{ij} between the latest and the next collisions between a pair of particles i and j can be calculated as follows

$$|\vec{r}_{ij}(t_0) + \vec{v}_{ij}(t_0)t_{ij}| = \sigma_i + \sigma_j, \tag{1}$$

where $\vec{r}_{ij}$ is the relative position vector between particles i and j, $\vec{v}_{ij}$ is the relative velocity, σ_i the radii of the hard spheres and t_0 is the time step at which the latest collision occurred. To get the "global" collision time, $t_C := \min_{ij}(t_{ij})$, *naïvely* one needs to compute the collision time between all particle pairs.

Task 1: Assuming that all hard-spheres have identical masses $m_i = m$ and radii $\sigma_i = \sigma$, implement the event-driven dynamics in 1- and 2-dimensions (assume also perfect-slip collisions between hard-disks).

Hint: Note that, only the neighbouring hard-rods interact with each other in 1-dimension. There you can set the width (radius) of the rods to $\sigma_i = 0$.

Task 2: Investigate different scenarios:

- Simulate a 1-dimensional row of N beads in a box with restitution coefficient $e = 1$, and validate your implementation conserves the total energy.
- Consider a 1-dimensional row of N beads with $e < 1$ hitting a resting wall. Compute the effective restitution coefficient $e_{\text{eff}} := \sqrt{E_f/E_i}$, where E_i and E_f are the initial and final total kinetic energies of the system, respectively. Vary N at fixed e. Determine $N = N_c$, at which e_{eff} practically vanishes, in other words, the cluster of beads ceases to re-bounce.

Task 3 (optional): Improve the efficiency of your code by storing the events for each particle in a priority queue (see lectures). While checking all particle pairs requires $\mathcal{O}(N^2)$ operations, generating events only for the colliding particles and inserting them in the queue has a complexity[2] of only $\mathcal{O}(N \log N)$.

Hint: Only the interacting particles need to be advanced, but keep in mind that the collision can invalidate the previously predicted events. You can either remove them manually from the queue in time $\mathcal{O}(N)$ or, more efficiently, keep a counter of each particle's collisions and identify an event as invalid only when resolving it.

[1] In fact, note that, the event driven algorithm is *exact* if the velocities and the collision times are calculated with *infinite* precision. However, computers truncate floating point numbers at a finite precision. This is particularly important in dimensions > 1 because there the negative curvature of the surface of the sphere leads to *chaos* in the system, *i.e.* there can be the vast differences in the final states despite having only marginal differences in the initial conditions. Therefore, the event-driven MD simulations would be valid only for a limited number of time steps despite the numbers calculated by the algorithm have millions of digits.

[2] Intuitively, the bound is even tighter in 1-dimension: methods for an insertion time of $\mathcal{O}(1)$, *i.e.* $\mathcal{O}(N)$ in total, have been proposed. See, *e.g.*, https://arxiv.org/pdf/physics/0606226.pdf.