Institute for Theoretical Physics
University of Cologne

Prof. Dr. S. Trebst
Jan Attig

Computational Many-Body Physics

Exercise Sheet 6

Summer Term 2018

Due date: Monday, **9th July** 2018, 2 pm

Website: www.thp.uni-koeln.de/trebst/Lectures/2018-CompManyBody.shtml

Exercise 16: Molecular Dynamics – from molecules to galaxies

In the last exercises you have deepened your numerical understanding of many-fermion problems and also learned about Monte Carlo simulations — an approach which, however, *replaces* the physical dynamics of a system by an ad-hoc artificial dynamics. In contrast, *molecular dynamics simulations*, which we introduce in this assignment, mimic the true dynamics of the system more closely allowing for a variety of insights into the dynamical behavior of a many-body system.

Molecular dynamics simulations are suited to investigate the collective behavior of a large number of interacting particles in continuous space. To every constituent particle (or molecule) a position and a velocity are assigned. The basic principle then is to determine the net force acting on each particle and — using Newton's second law — to update the velocity of the particle. Subsequently, all particles are moved according to their current velocity and a chosen time discretization. This procedure is repeated in a long sequence of consecutive time steps. The dynamical behavior of the system can then be observed both qualitatively by visualizing the positions the particles in an animation and quantitatively by measuring observables such as temperature, average velocity or pair-correlation functions.

A naive approach of determining the effective force onto a particular particle is to sum up the forces exerted by the totality of all other particles. However, doing so would strongly restrict the number of particles N, because the number of mutual interacting forces to be considered would scale as $(N-1)+(N-2)+(N-3)+\ldots \sim O(N^2)$. Therefore, advanced strategies have to be applied to cope with the large number of mutual forces. In the two examples on this exercise sheet, we cover the appropriate algorithm for short ranged interactions, namely the **cell method**, as well as the preferred algorithm for long ranged interactions, namely a **tree code method**.

Molecules

In the first part of this exercise, your task will be to implement a molecular dynamics simulation for the argon gas in two dimensions (at a given temperature of $T = 150K$). Its interactions are described by the Lennard-Jones potential

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

with parameters $\epsilon = 1.65 \times 10^{-21} J$ and $\sigma = 0.335$nm. The atomic mass of argon is $m = 40$u, and $k_B = 1.38 \times 10^{-23}$ J/K. Use the following units for the simulation: [Length] = 1nm, [Time] = 1ps, [Mass] = 1u and [Temperature] = 1K.

a) In a first step, we want you to implement a working molecular dynamics simulation based on the naive calculation of all forces. To do so, proceed as follows:

(i) Set up a system with 100 atoms initially arranged on a grid. Draw the initial velocities from a Maxwell-Boltzmann distribution, which is a Gaussian distribution with $\sigma^2 = \frac{k_B T}{m}$. To do so, you can use the julia package Distributions.jl.

(ii) Implement a function that determines the interacting force between two atoms using the Lennard-Jones potential and a function for the update of positions and velocites using the Verlet method. Complete your code so that you obtain a working MD simulation with a temporal discretization $\Delta t = 0.01ps$. (Do not try to perform measurements at this stage!)

(iii) *(Optional)* Show your results as an animation and visually check if your simulation performs as expected (do particles attract each other or repell each other? Can you observe discontinuities?)

With a naive implementation done, you can improve the simulation by adding the cell method. The idea of the cell method is as follows: The system is divided into equally sized cells, so that each particle is contained in exactly one particular cell. For every particle, only interactions with those of the other particles are considered which reside in the same cell or one of the neighboring cells. Hence, for a two-dimensional system only 9 cells need to be considered. Technically, the method can be implemented using two lists **A** and **B**. The indices of **A** enumerate the cells and the indices of **B** enumerate all particles. **A** has one entry for every cell, namely the particle index of the first particle in this cell. (Which of the particles in the cell is *first* is arbitrary.) If a cell is empty, the corresponding list entry for that cell should by set to -1. **B** has one entry for every particle, namely the particle index of the *next* particle in the same cell. (Again, the ordering of the particles within the cell is arbitrary.) For the last particle, the list entry should be set to -1 in order to indicate that there is no next particle. Update the two lists only if a particle has moved to another cell. Only three list entries need to be changed in that case.

To illustrate the method in a little example, consider 3 cells and 6 particles. Particles 2 and 4 are in cell 1, cell 2 is empty and the rest of the particles are in cell 3. The two lists could look like this: $A[1] = 2,\ A[2] = -1,\ A[3] = 5$ and $B[1] = -1,\ B[2] = 4,\ B[3] = 1,\ B[4] = -1,\ B[5] = 6,\ B[6] = 3$.

b) Apply and implement the cell method with 6×6 cells in order to reduce the number of interactions.

c) Improve your code by adding a a velocity rescaling after every 10 MD steps in order to keep the temperature constant at T=150K.

With the code set up, it is now time to measure observables. In particular, we want you do simulate your system at different densities $\rho = 0.0007/\text{nm}^2$ and $\rho = 0.001/\text{nm}^2$ and compare both scenarios.

d) Measure the velocity distribution and the radial pair-correlation function $g(r)$ averaged over time. Interpret your result for $g(r)$ in comparison with the animation. Reminder: The radial pair-correlation function for a 2D system is defined as

$$g(r) = \frac{1}{2\pi} \int g(\vec{r}) d\theta \,, \tag{2}$$

where

$$g(\vec{r}) = \frac{1}{\rho(N-1)} \left\langle \sum_{i \neq j} \delta(\vec{r} - \vec{r}_{ij}) \right\rangle \,. \tag{3}$$

Use the discretized formulation of $g(r_k)$ for $r_k = k \cdot \Delta r$ which needs discrete histograms of the particle separations: $h(r_k)$ is the number of particle *pairs* with $r_{ij} \in [(k - \frac{1}{2})\Delta r, (k + \frac{1}{2})\Delta r]$. We then have

$$g(r_k) = \frac{2}{\rho(N-1)} \frac{\langle h(r_k) \rangle}{2\pi r_k \Delta r} \,. \tag{4}$$

Galaxies

In the second part of this exercise, we want to switch applications to the galactic scale, more precisely, we want you to simulate the **collision of two galaxies**. In principle, this calculation still relies on the same many-body molecular dynamics simulation that you already set up in the first part of this exercise. The biggest difference between the galactic scenario and the molecular scenario is the difference in interactions. Whereas interactions between molecules are mostly due to Van-der Waals interactions and Pauli exclusion, the interactions between galactic bodies are dominated by gravity. This means that the interaction potential between two bodies with masses m_1 and m_2 changes to

$$V(\vec{r}) = -G \frac{m_1 m_2}{r} \,, \tag{5}$$

which is long ranged in contrast to the previous implemented Lennard Jones potential. This long ranged nature of the potential makes it impossible to simulate by using the cell method (as this algorithm only accounts for small neighborhoods of interactions).

An efficient algorithm for these long ranged type of interactions is the **Barnes-Hut tree code algorithm** which was developed in the 80s by Josh Barnes and Piet Hut in order to simulate galactic dynamics. The core principle of this algorithm is to divide space in a tree code fashion into boxes which are either empty, or themselves are divided into smaller boxes and so on or contain exactly one body. For calculation of interactions, this subdivision can be truncated and treated as an effective body if the center of mass is relatively far away. Such a procedure reduces the number of necessary interactions by a huge factor.

For the precise working and implementation of the Barnes-Hut algorithm, we refer you to the very nice documentation of a graph library called arbor.js which features a section on the Barnes-Hut algorithm at arborjs.org/docs/barnes-hut.

e) Change your code of part a) to implement the graviational potential as well as the Barnes-Hut algorithm to update your forces. Further implement galaxies as initial conditions of your particles and visualize the dynamics of two colliding systems as an animation, similar to this movie.