Course:	Computational Sciences 20/21
Institute:	Freie Universität Berlin
Lecturer:	Sebastian Matera, Frank Noe
Assistant:	Gottfried Hastermann

MOLECULAR DYNAMICS OF WATER
PROJECT ASSIGNMENTS FOR 5 – 7 STUDENTS

GOTTFRIED HASTERMANN

1. Introduction

Solving Newton's equations of motion does not immediately suggest activity at the cutting edge of research. The molecular dynamics algorithm in most common use today may even have been known to Newton. [1] In strong constrast to the prepending quote, taken from an introductory text on molecular dynamics, one can argue that the simulation of large particle systems on multiple scales is non trivial and an ongoing endevour. To familiarize with the computational tasks, this project considers one of the most ubiquitous molecules, water. Despite its relative simple structure, the importance of water as solvent results in a whole zoo of available models.

2. Model

Consider n particles which positions and momenta are denoted by $q_i \in \mathbb{R}^3$ and $p_i \in \mathbb{R}^3$ respectively. Let $I := \{1, \dots, n\}$, then the set of all indices describing Oxygen atoms is denoted by $\mathcal{O} \subseteq I$, the set of indices for Hydrogen atoms by $\mathcal{H} \subseteq I$ and the set of index triples (H, O, H) for water molecules $\mathcal{M}$. The Hamiltonian energy functional for a flexible SPC water model is given by

(1)

$$H(p_1, \dots p_n, q_1 \dots q_n) = T(p_1, \dots, p_n) + V_{\text{int}}(q_1, \dots, q_n) + V_{\text{ext}}(q_1, \dots, q_n)$$

$$T(p_1, \dots, p_n) = \frac{1}{2} \sum_{i \in I} \frac{\|p_i\|^2}{m_i} \tag{2}$$

$$V_{\text{int}}(q_1, \dots, q_n) = \frac{1}{2} \sum_{(h_1, o, h_2) \in \mathcal{M}} k_\theta (\theta(q_{h_1}, q_o, q_{h_2}) - \theta_{eq})^2 + \frac{1}{2} \sum_{l=1}^{2} k (q_{h_l} - q_o - q_{eq})^2 \tag{3}$$

$$V_{\text{ext}}(q_1, \dots, q_n) = \sum_{i \neq j \in \mathcal{O}} \frac{A}{\|q_i - q_j\|^{12}} - \frac{B}{\|q_i - q_j\|^6} + C \sum_{M_1 \neq M_2 \in \mathcal{M}} \sum_{\substack{i \in M_1 \\ j \in M_2}} \frac{c_1 c_2}{\|q_i - q_j\|}. \tag{4}$$

Hereby $m_i > 0$ and $c_i \in \mathbb{R}$ denote the atomic masss and charges respectively. Furthermore C is the Coloumb constant, $k > 0$ denotes the stiffness of the covalent bonds and $k_\theta > 0$ the stiffness of the angle between both covalent bonds. The constants θ_{eq} and k_{eq} denote the equilibrium angle and length respectively. A and B are the constants which determine the shape of the Lennard-Jones potential terms.

3. Tasks

Tasks marked by * are optional. You need to solve all mandatory assignments. The maximum number of points to achieve is 20 points per project member. Please use the Wiki of the repository for documentation and answering the questions or provide a LaTeX report. The final software should be a python package including a `setup.py`, contain a `Readme.md` explaining the usage as well as a file tracking the dependencies named `requirements.txt`.

3.1. **Project management.** Apart from the scientific content you should work on the following organizational assignments.

- Setup a development process. For this purpose do research on the matter, document and report your findings. You do not have to use an existing one, but they could serve as a blueprint for your own. **(10 pts)**
- Maintain a proper git history for each project member in line with the chosen development process. **(10 pts)**

* Use GitLab for issue management. **(5 pts)**

* Setup and utilize GitLab's continuous integration tools for your tests. **(5 pts)**

3.2. **Preliminaries.** Familiarize with the given dynamical system and discuss the properties.

- Explain the dynamics induced by each of the terms of the potential energy V and determine the equations of motion for this Hamiltonian system. **(5 pts)**
- Discuss the computational restrictions raised by the individual terms in the potential energy function. **(5 pts)**

3.3. **Direct Simulation.** To check your findings, simulate water molecules on the whole space $\mathbb{R}^3$.

- Choose an appropriate numerical integration method to compute trajectories of the resulting system of ordinary differential equations. Implement tests and the algorithm in a vectorized way. **(10 pts)**

* Implement a rigid water version by additionally enforcing the constraint

$$(q_{h_i} - q_o - q_{eq})^2 = 0 \quad \forall i \in \{1, 2\}, \forall (h_1, o, h_2) \in \mathcal{M} \tag{5}$$

This should be done by solving the resulting differential algebraic equations by the means of the SHAKE/RATTLE algorithm. **(10 pts)**

- Find an appropriate file format to save your simulation data. Again implement the tests first and subsequently add input/output functionality to your code. Visualize your simulation results. **(10 pts)**

* Implement a Verlet list to accelerate the force field evaluation. **(10 pts)**

3.4. **Periodic Boundary Conditions.** In many cases one would like to consider a microscopic box of water taken from a (much) larger medium. Therefore periodic boundary conditions seem to be a good assumption.

- Not all of the terms in the potential energy function are local. What are the implications for your algorithm? Document your findings. **(5 pts)**
- Use a cut off for all short range interactions and implement periodic boundary conditions on a box which has edges at least as long as the cut off radius. (5 pts)
- Approximate the far field interaction by the Ewald Summation. **(10 pts)**
- Improve the far field evaluation by an implementation of the particle Ewald Summation. **(10 pts)**
- Explain the computational complexity of your algorithm and validate by benchmarks. Compare the different approximations. **(5 pts)**

3.5. **Parallelize your code.** Once you optimized for algorithmic efficiency, you should make sure to leverage all (or at least more) of the computational resources on your computer hardware efficiently. For this purpose

- Determine and explain the time critical parts in your code by benchmarks. Discuss the different parallelization options fitting to your code. Can you implement all of them in python? **(10 pts)**

Apply one or more of following strategies. Test and benchmark your code against unoptimized results. Report and explain your findings.

* Improve performance by using just in time compilation and vectorization (SIMD) from `numba`. Parallelize your software using `DASK` where applicable. Do both for the particle-particle interactions first. Consider the (less straight forward) approximated force field evaluation subsequently. **(10 pts)**
* Improve performance by offloading particle-particle interactions to accelerators via `pyopencl` or `pycuda`. **(15 pts)**
* Improve performance via an optimized version of the Verlet list using `pybind11` and `C++`. **(20 pts)**

References

[1] N. Attig, K. Binder, H. Grubmueller, and K. Kremer, *Publication Series of the John von Neumann Institute for Computing (NIC) NIC Series Volume 22*, vol. 22. 2004.