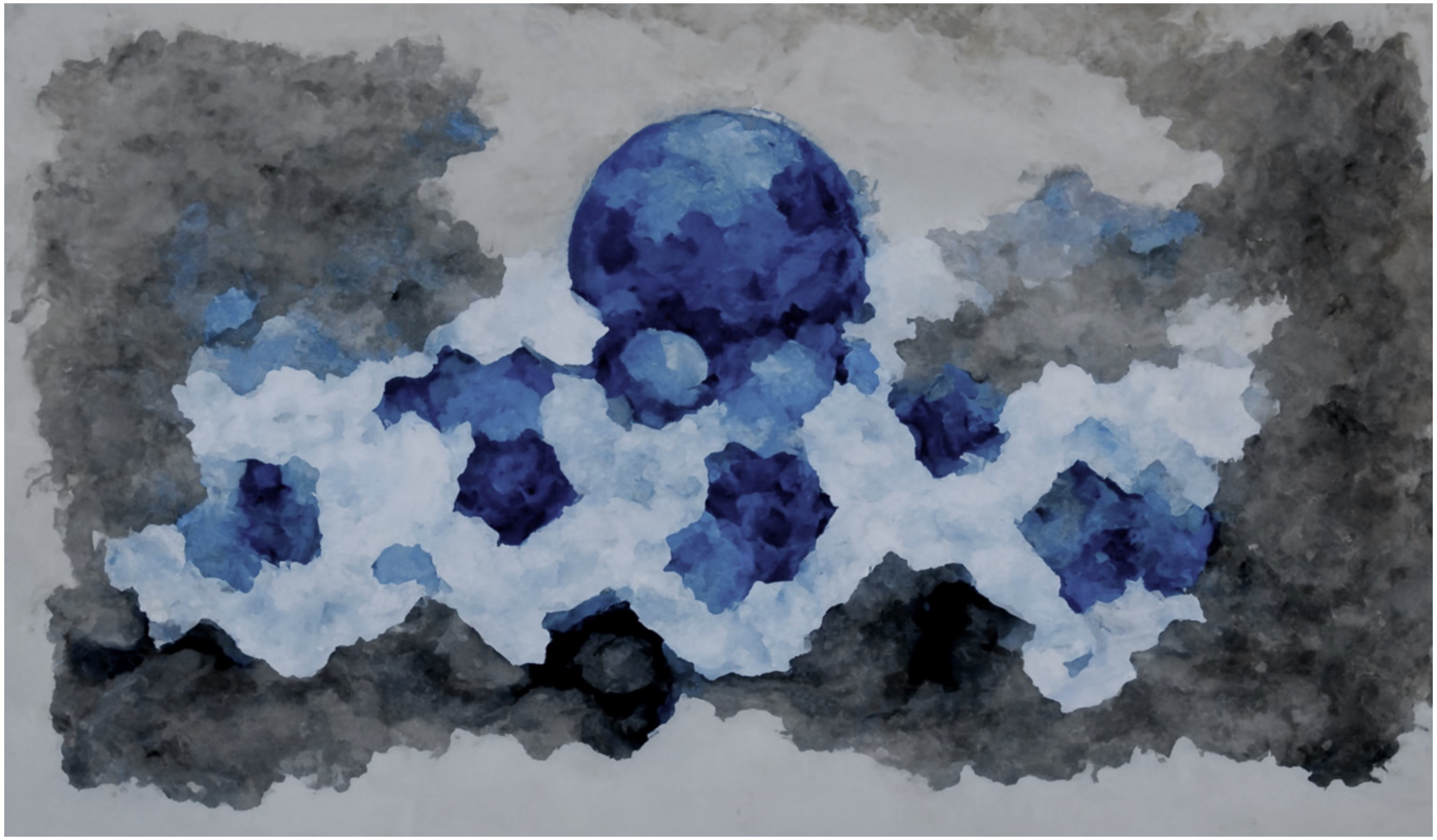
MD Simulator

Final Project of APC 523

Bingjia Yang

https://github.com/iceplusssss/APC523Project



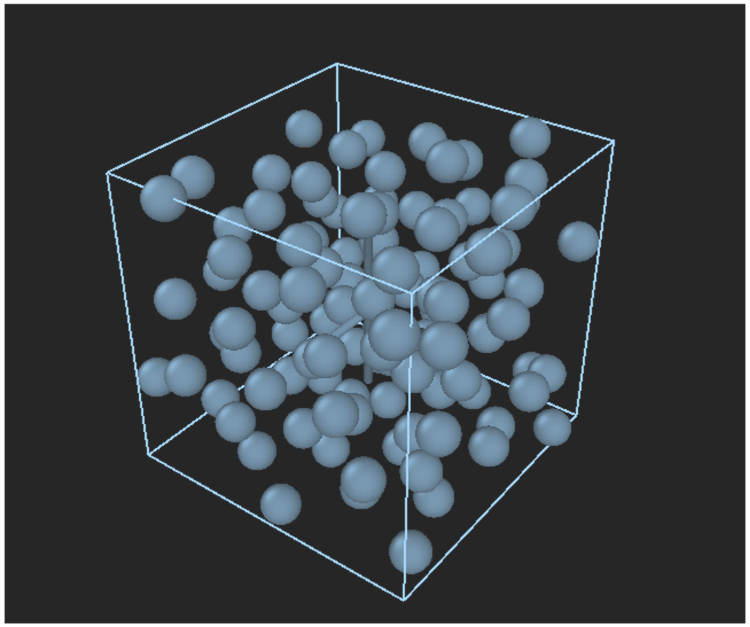
Cover art generated by Disco Diffusion v5.2
keywords: a single molecule, ocean, lonely, cyberpunk, by watercolor)

Outline

Introduction

Consider a classical system composed of N atoms

$$H(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + U(\boldsymbol{q}_1, \ldots \boldsymbol{q}_N)$$



MD Simulator is written in python and the main dependencies are python packages Numpy and ASE. It can perform molecular dynamics simulation for a given simulated ensemble and force field.

Introduction

- Installation

The user could install MDS using pip:

```
$ pip install .
```

- Getting Started

To run a simulation using MDS, the user should first initialize a *dynamics* object. The argument is the path of the input *json* file.

```
my_simulation = mds.dynamics("./input.json")
my_simulation.run()
```

Mathematical Model: Velocity Verlet Algorithm

The evolution of the system can be expressed as

$$\begin{pmatrix} \boldsymbol{q}(\Delta t) \\ \boldsymbol{p}(\Delta t) \end{pmatrix} = \exp\left(\frac{\Delta t}{2}\boldsymbol{F}(0)\frac{\partial}{\partial \boldsymbol{p}(0)}\right)\exp\left(\Delta t\frac{\boldsymbol{p}(0)}{m}\frac{\partial}{\partial \boldsymbol{q}(0)}\right)\exp\left(\frac{\Delta t}{2}\boldsymbol{F}(0)\frac{\partial}{\partial \boldsymbol{p}(0)}\right)\begin{pmatrix} \boldsymbol{q}(0) \\ \boldsymbol{p}(0) \end{pmatrix}$$

For an example of a single particle moving in one dimension, velocity Verlet algorithm can be expressed as a three-step procedure:

$$\begin{aligned} p(\Delta t/2) &= p(0) + \frac{\Delta t}{2}F(x(0)) \\ x(\Delta t) &= x(0) + \frac{\Delta t}{m}p(\Delta t/2) \\ p(\Delta t) &= p(\Delta t/2) + \frac{\Delta t}{2}F(x(\Delta t)) \end{aligned}$$

Time-reversible and Symplectic

Mathematical Model: Force Field

In this work, we take the Lennard-Jones potential as an example.

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

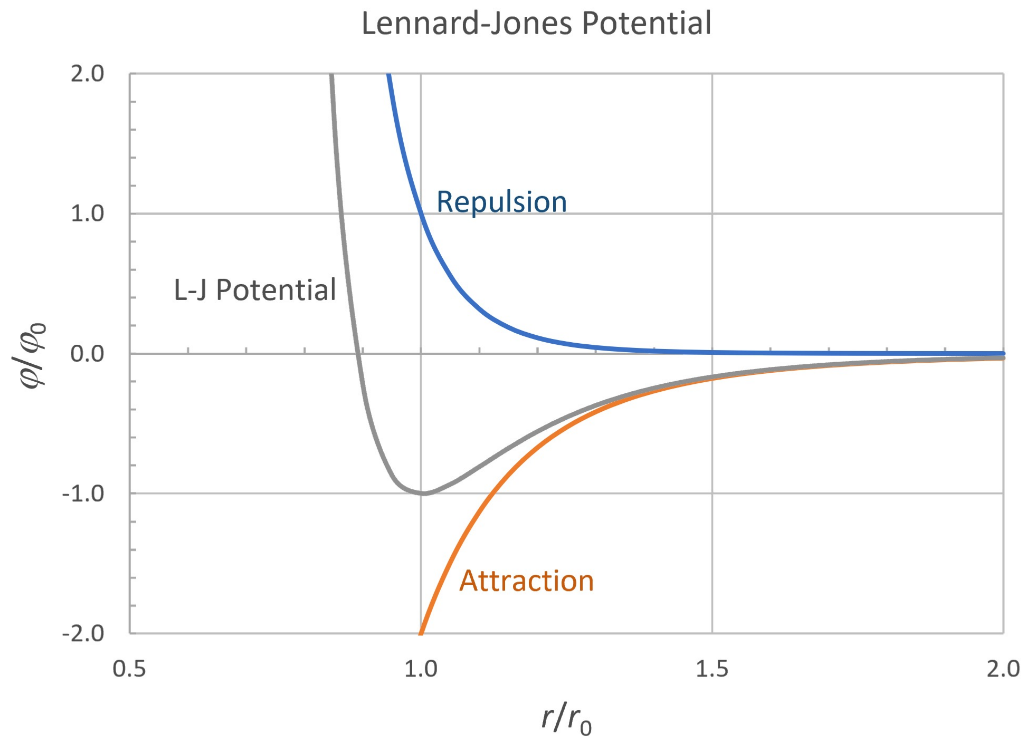


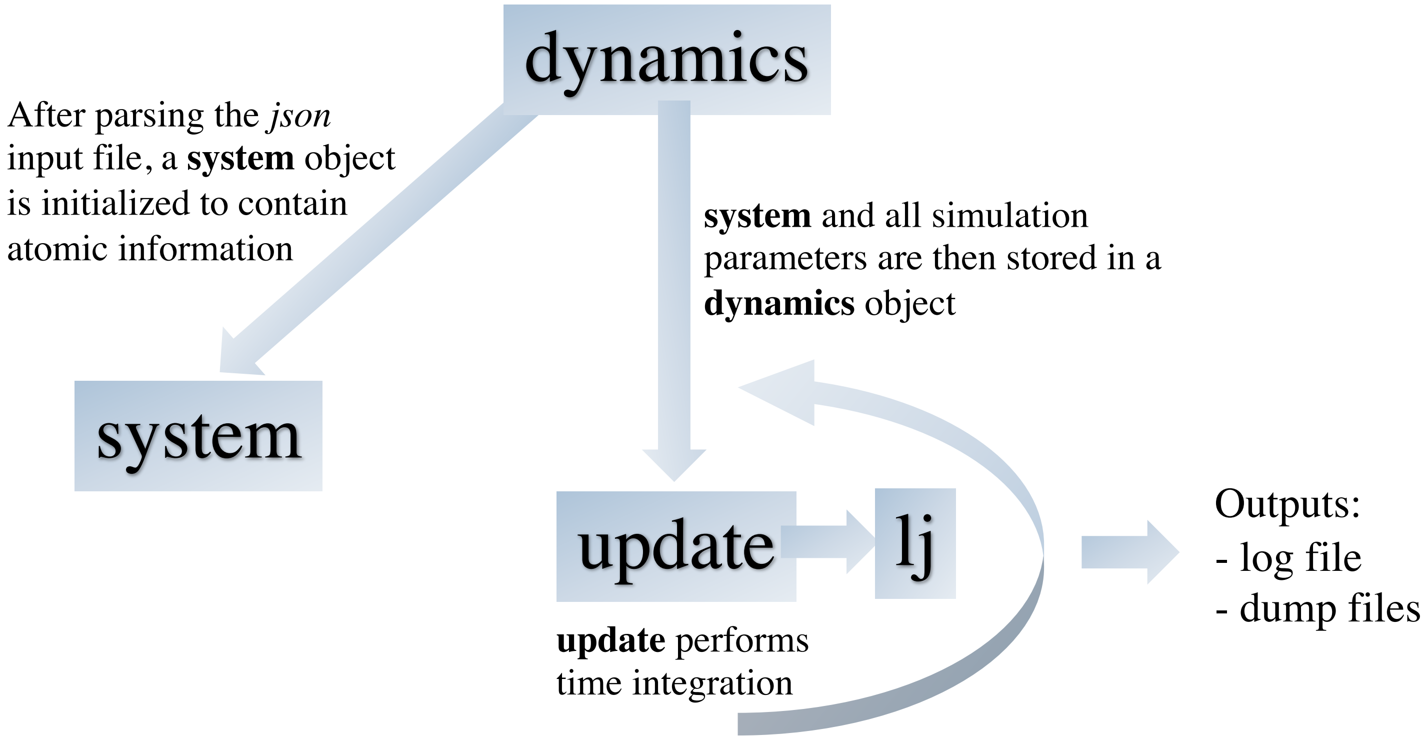
image cr: https://polymerdatabase.com/polymer%20physics/Lennard-Jones.html

Software Architecture: Inputs

The user should prepare a *json* input file determining the initial configuration of the system as well as the simulation parameters. Currently, we support following key words:

mode	defines the simulated ensemble, e.g. nve
init_temp	initial temperature used to create random velocities for all atoms
time_step	the amount of time by which the integrator steps, unit: fs
total_steps	total simulation time is just *total_steps* multiplied by *time step*
init_conf_path	initial configurations (format: xyz) provided by the user
ff_style	defines which force field is used, e.g. lj
ff_coeff	a list of force fields coefficients, e.g. [ε, σ, cutoff] for lj
log_freq	sets the frequency to print thermodynamic information
dump_freq	sets the frequency to dump a snapshots of the system

Software Architecture: Simulation


dynamics
After parsing the *json* input file, a **system** object is initialized to contain atomic information
system
system and all simulation parameters are then stored in a **dynamics** object
update
lj
update performs time integration
Outputs:
- log file
- dump files

At current stage, only micro-canonical ensemble (NVE) and Lennard-Jones potential are implemented.

Software Architecture: Outputs

There are two kinds of outputs in this program:

Log file	**a list of thermodynamic observables printed every *log_freq* steps to a single text file.** The default is time step, potential energy, kinetic energy, total energy and cell volume.
Dump files	**snapshots of atoms which are printed every *dump_freq* steps to separate xyz files.** These files could be then easily imported to visualization softwares like OVITO for analysis.

Examples: Case study

Data at *./example/argon_rdf/backup_results*

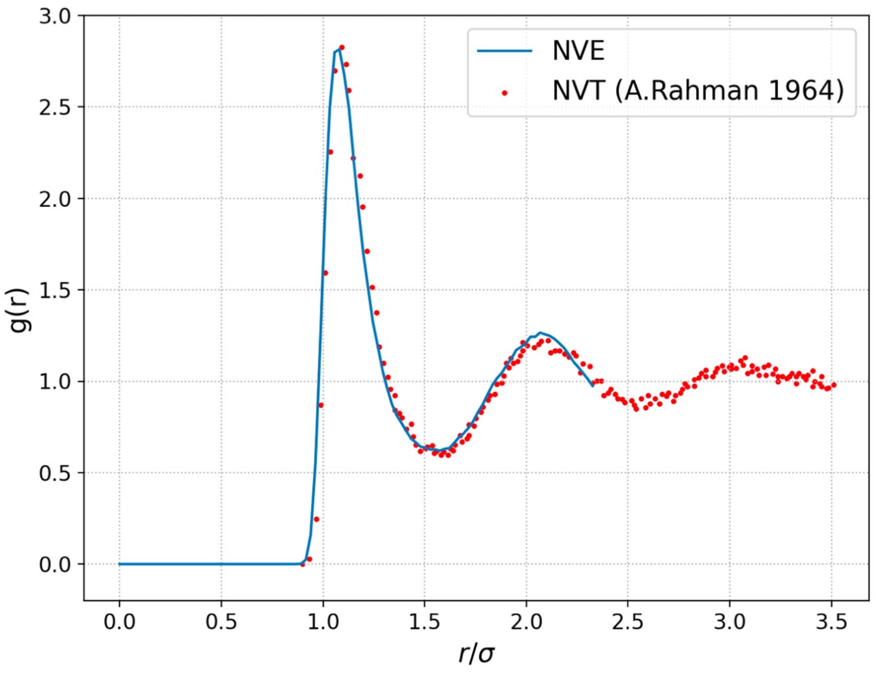
A sample problem: 108 Ar atoms interacting with Lennard-Jones potential (coefficients: $\varepsilon = 0.01\text{eV}$, $\sigma = 3.405$ ˚A and cutoff=8 ˚A).
The *json* input is as follows:

```
{
    "mode": "nve",
    "init_temp": 94,

    "time_step": 1,
    "total_steps": 100000,

    "init_config_path": "./ar108.xyz",
    "ff_style": "lj",
    "ff_coeff": [0.01, 3.405, 8],

    "log_freq": 100,
    "dump_freq": 100
}
```



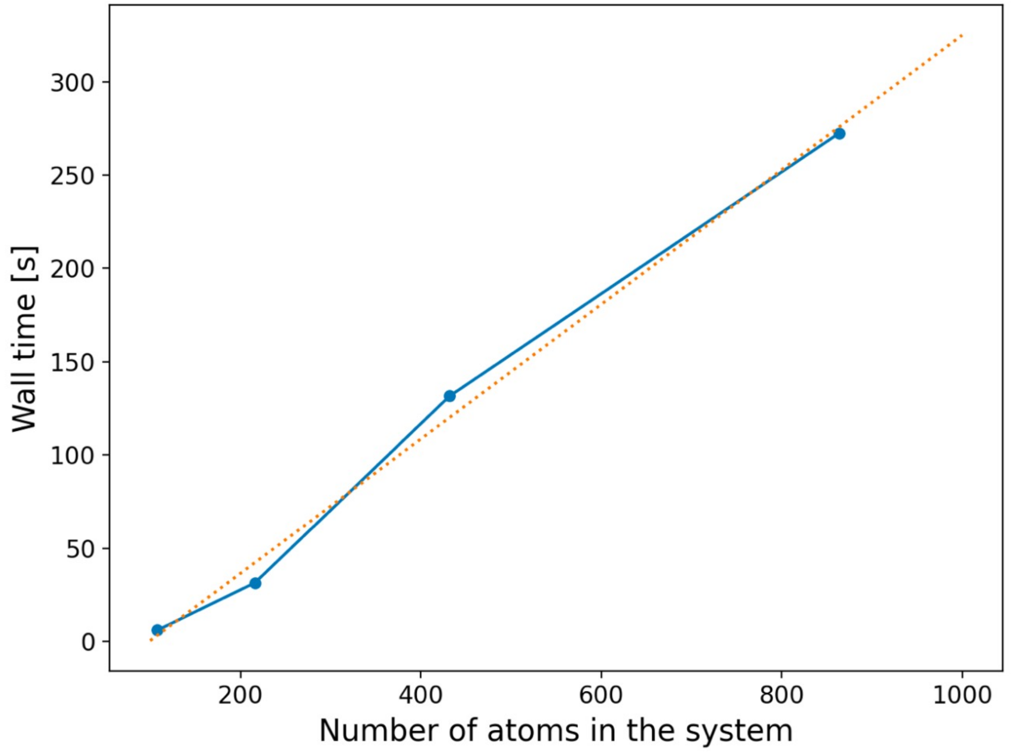
The Radial distribution function (RDF) calculated by **MDS** using NVE ensemble shows a good agreement with the reference data.

reference data: Rahman, A., 1964. Correlations in the motion of atoms in liquid argon. Physical review, 136(2A), p.A405.

Example: Benchmark

Data at *./example/benchmark/backup_results*

On the same problem defined in the last slide, we benchmark our program:



The *total_steps* is 100 for these simulations. They all start from the same initial temperature and atomic density.

MDS scales linearly in system size for this problem

Summary and Outlook

We implemented a program, **MDS**, for integrating Newton's equations of motion with a given simulated ensemble and Lennard-Jones potential.

This code is designed to be easy-to-extend with new features, such as other boundary conditions, new force fields or new simulated ensembles.

Thanks!