User Guide

Classical Molecular Dynamics & Friction Tensor Calculation

Alberto Prato - University of Padua

albertoprato@studenti.unipd.it

Contents

1 Introduction

This Fortran program simulates the classical molecular dynamics of a solute consisting of four spheres immersed in a viscous bath. The simulation uses Lennard-Jones potentials and integrates numerically the equations of motion using the velocity Verlet algorithm.

The goal is to compute the friction tensor of the solute as a function of time through the time-autocorrelation function of the forces experienced by the solute particles.

2 Usage

2.1 Prerequisites

To compile and run this software, the following tools and libraries are required:

- GFortran: GNU Fortran compiler.
- FFTW3: "Fastest Fourier Transform in the West" library.
- Make: GNU Make build tool.

2.2 Compilation and Running

A `Makefile` is provided in the source directory. To compile and run the program:

1. Navigate to the directory containing the source code (`verlet-LJ/`).
2. Run the command:

   ```
   make
   ```

   This will generate an executable named `simulation`.

   Note: Ensure that the linker flags in the Makefile (`-lfftw3 -lm`) match your system's configuration.

3. Once compiled, the simulation requires the input files (`input.txt` and `system.xyz`) to be present in the directory `input/`. Run the executable from the terminal:

   ```
   ./simulation
   ```

3 Code Structure and Features

The source code is modular, with distinct responsibilities assigned to separate Fortran modules.

3.1 Main Program

The driver of the molecular dynamics simulation is the `main.f90`. Its primary responsibility is the time-evolution of the system through velocity Verlet integration scheme. Additionally it handles memory allocation, input reading and calls to other modules for initialization, minimization, and final analysis.

3.2 Modules

`force_module.f90`
: Responsible for the evaluation of the forces and the potential energy. It models inter-atomic interactions via the Lennard-Jones potential. The module implements periodic boundary conditions utilizing the minimum image convention and applies a spherical cutoff.

`minimization_module.f90`
: Performs an initial geometric optimization of the solvent structure, utilizing the conjugate gradient method (Polak-Ribière variant) to drive the system towards a local energy minimum.

`velocity_init_module.f90`
: Initializes the solvent velocities from the Maxwell distribution at a target temperature. The module explicitly removes any net center-of-mass drift resulting from the stochastic sampling.

`fft_correlation_module.f90`
: A module acting as an interface to the FFTW3 library. It computes time-correlation functions though FFT.

`friction_module.f90`
: It computes the time-dependent memory kernel (friction tensor) via the autocorrelation function of the instantaneous forces exerted by the solvent on the solute.

`kinds.f90`
: Defines numerical precision parameters (single and double precision) used throughout the simulation for consistency.

3.3 Dependencies

The program handles the compilation hierarchy:

- all modules depend on `kinds`.
- `minimization_module` depends on `force_module`.
- `friction_module` depends on `fft_correlation_module`.
- `main` depends on all modules.

4 Input Files Structure

4.1 Simulation Parameters: `input.txt`

The program reads simulation parameters from `input.txt`. The file format is rigid and parameters must be provided in the exact order shown below.

Line	Parameters	Description
1	`n`, `dt`	total number of steps, time step
2	`mass`, `epsilon_ss`, `sigma_ss`	solvent mass, solvent-solvent LJ parameters
3	`epsilon_int`, `sigma_int`	solute-solvent LJ parameters
4	`temp`, `k_B`	temperature, Boltzmann constant
5	`box_L`	length of the cubic box

Table 1: Structure of `input.txt`

Example of a `input.txt` file:

```
50000    0.001
18.015   63.597   3.1507
43.472   3.338
298.15   0.831
20.0
```

where the units of measurement used are picoseconds, Daltons, Kelvin, and angstroms.

4.2 Initial Configuration: `system.xyz`

The initial configuration must be provided in standard XYZ format, typically generated by software like Packmol. The file structure is rigid and must adhere to the specific line order described below.

Line	Description
1	total number N of atoms in the system
2	arbitrary string (skipped by the parser)
3 – 6	coordinates of the 4 solute particles
7 – $N+2$	coordinates of the remaining solvent particles

Table 2: Structure of `system.xyz`

Example of a `system.xyz` file:

```
267
Built with Packmol
C             1.250000         1.510000         0.000000
C             0.000000         0.670000         0.000000
C             0.000000        -0.670000         0.000000
C            -1.250000        -1.510000         0.000000
O             4.814346       -10.023884         6.353803
O             1.027902         9.993801         0.817163
O             2.820898         8.581818        -9.999455
O             6.655889         6.137787         9.888166
....
```

5 Output Files

The simulation produces three output files:

- `trajectory.xyz`: contains the atomic coordinates of the system at regular intervals (every 100 steps) during the simulation, in the standard XYZ format. This file is compatible with visualization software such as VMD.
- `equilibration_stats.dat`: records the system properties during the equilibration phase.
  - Column 1: time
  - Column 2: potential energy
  - Column 3: mean squared displacement
- `friction_tensor.dat`: contains the elements of the friction tensor, calculated as the time-autocorrelation of the forces acting on the solute.
  - Column 1: time
  - Columns 2-13: friction tensor diagonal components for each of the 4 solute particles.

A supplementary jupyter notebook (`Plot_Friction_Tensor.ipynb`) is provided in the `python_tool/` directory. It allows to visualize the output data and graphically inspect the decay of the friction memory kernel.