Particle-Based Simulations — Assignment 2 (2025)

Molecular Dynamics of n-Butane (United-Atom Force Field)

Context and goals

In this assignment you will implement a molecular dynamics (MD) simulation of liquid n-butane using a united atom force field. You will extend the provided code to include non-bonded, bond-stretch, bond-angle, and torsional interactions. Afterwards, you will analyze the velocity distribution, conformational distributions (trans vs. gauche), and compute a diffusion coefficient from the mean squared displacement (MSD).

System: n-Butane at a density of $\rho = 600$ kg/m^3 and $T = 298$ K.

Force field functional forms and parameters

All interactions are modeled with the following potentials:

Non-bonded (Lennard–Jones)

$$U_{\mathrm{LJ}}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] - U_{\mathrm{LJ}}(r_{\mathrm{cut}}) \quad \text{for } r \leq r_{\mathrm{cut}},$$

and zero otherwise. Use $r_{\mathrm{cut}} = 14$ Å. Do not include the LJ interactions for 1-2, 1-3 and 1-4 bonded pairs.

Site	ϵ/k_B (K)	σ (Å)
CH_3	98.0	3.75
CH_2	46.0	3.95

Use Lorentz–Berthelot mixing rules:

$$\sigma_{ij} = \tfrac{1}{2}(\sigma_i + \sigma_j), \qquad \epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}}.$$

Bond stretch (harmonic)

$$U_{\mathrm{bond}}(r) = \tfrac{1}{2} k_b (r - r_0)^2,$$

with parameters:

$$r_0 = 1.54 \text{ Å}, \qquad k_b/k_B = 3.19 \times 10^5 \text{ K/Å}^2.$$

Angle bend (harmonic)

$$U_{\mathrm{angle}}(\theta) = \tfrac{1}{2} k_\theta (\theta - \theta_0)^2,$$

with parameters:

$$\theta_0 = 114^\circ, \qquad k_\theta/k_B = 6.25 \times 10^4 \text{ K rad}^{-2}.$$

Torsion (dihedral)

For the central CH_3–CH_2–CH_2–CH_3 dihedral angle ϕ (defined by the four consecutive atoms along the chain), use a Ryckaert–Bellemans form:

$$U_{\text{tors}}(\phi) = c_0 + c_1 \cos\phi + c_2 \cos^2\phi + c_3 \cos^3\phi.$$

with parameters:

$$c_0/k_B = 1010.0 \text{ K}, \quad c_1/k_B = -2018.9 \text{ K}, \quad c_2/k_B = 136.4 \text{ K}, \quad c_3/k_B = 3165.3 \text{ K}.$$

This form produces a global minimum at $\phi = 180°$ (trans) and local minima near $\pm 60°$ (gauche), consistent with experimental conformer ratios.

Learning objectives

- Implement non-bonded, bond, angle, and torsional forces in `forces.c`.
- Verify stability in NVE simulations and control temperature with a Berendsen thermostat (NVT).
- Analyze the torsional distribution and quantify trans vs. gauche populations.
- Compute the diffusion coefficient from the long-time slope of the MSD.

Understanding, compiling and extending the code

The source code consists of several `.c` and `.h` files. Together these files constitute the Lennard–Jones molecular dynamics simulation code. Below is a short description of the purpose of each file:

- `main.c`: the main program
- `constants.h`: header file with constants for the entire program
- `struct.h`: header file with structs for global variables, neighbor list, and arrays
- `memory.c`: functions for allocating memory
- `setparameters.c`: parameter settings for the simulation
- `random.c`: random number generators
- `initialise.c`: initialization of variables, particle positions, and velocities
- `nbrlist.c`: creation of neighbor lists
- `forces.c`: calculation of forces on particles
- `dynamics.c`: updating positions and velocities, applying boundary conditions, and a (Berendsen) thermostat
- `fileoutput.c`: output of files (currently for visualization)

To correctly compile and link these source files, open the folder containing them in, e.g., Visual Studio Code. An example `tasks.json` file is included in the `.vscode` subfolder. This configuration file might need some adaptation for your setup. It provides two build versions: `md_debug.exe` for debugging, and `md.exe` for production runs.

The code contains Doxygen-style comments. On Canvas you will find a link to the generated documentation. Use this documentation to better understand the code.

Extending the code

When modifying the code, follow these rules when adding functions, variables, or arrays:
To add a new function,

(i) Add the new function to a suitable `.c` file.

(ii) Add a prototype of the declaration to the corresponding `.h` file.

If you decide to store your new function in a *new* `.c` file, create a corresponding `.h` file. Include the `.h` file in all `.c` files where the declared functions are used.

To add a new variable,

(i) Add the variable to an existing `struct` in `structs.h` (or create a new struct, and instantiate it in `main`).

To add a new array,

(i) Add the array to an existing `struct` in `structs.h` (or create a new struct, and instantiate it in `main`), *and*

(ii) Allocate memory of the right length in the function `alloc_memory` in `memory.c`.

(iii) Free the allocated memory in the function `free_memory` in `memory.c`.

Now try to compile and run the program with the default settings in `setparameters.c`. If all goes well, an MD simulation will run for the specified number of steps. The default parameters should give reasonable results, but you may need to change them for better settings.

A major task in this assignment is implementing bonded forces. For this you need to implement the feature that particles can have different types (CH_3 and CH_2). In the struct `Vectors` an integer array `type` is defined; you should use this. In `initialise.c` there is a minimal implementation of the `initialise_types` function. Extend this to assign types to the particles. In velocity updates and kinetic energy calculations, take into account that particles have different masses (see the table below). Use the `type` array to assign the correct mass to each particle. For the bonded and non-bonded interactions, implement the corresponding forces and energies. Use the skeleton in `forces.c` and fill in the `TODO` sections for bond, angle, torsion, and non-bonded forces. Use the type array to select the correct parameters.

Visualization

The simulation program creates a `.pdb` file with the positions of all molecules at regular intervals. To visualize this data, you can use OVITO, available at https://www.ovito.org/.

Tasks

A. Modeling

A1) **Units.** It is common practice to introduce non-standard units in simulations. In your code use Kelvin for the energy scale (i.e., the energy unit is $k_B \times 1$ K), Å for distances, and atomic mass units (amu) for masses. The time unit follows from this. What is your unit of time expressed in SI units? For a timestep size of 1 fs, what is the timestep size in your internal units?

A2) **Bonded.** From the expressions for the bond, angle, and torsion potentials, derive the corresponding forces. Express the forces in terms of the bond vectors $\mathbf{r}_{ij}$, $\mathbf{r}_{kj}$, and $\mathbf{r}_{kl}$ as defined in `forces.c`.

B. Implementations and Testing

B1) **I/O.** Output diagnostic data (energies, T, P) to CSV each step.

B2) **Non-bonded forces.** Implement particle typing (CH_3 and CH_2) and the non-bonded LJ forces.

B3) **Test non-bonded forces.** Verify correctness by comparing with finite-difference derivatives of the potential for a particle pair. Demonstrate long-time energy conservation for a system at the target density in NVE.

B4) **Bonded forces.** Implement bond, angle, and torsion forces.

B5) **Test bonded forces.** Verify correctness with finite-difference derivatives for a single molecule. Demonstrate long-time energy conservation for a single molecule in NVE.

B6) **Initialization.** Devise a procedure to pack N molecules in a cubic box at target density and assign velocities from a Maxwell–Boltzmann distribution.

B7) **Test energy conservation.** From the packed configuration, run an NVE simulation and check energy conservation.

B8) **Thermostat.** Implement a Berendsen thermostat for NVT.

B9) **Test thermostat.** Starting from the packed configuration, run an NVT simulation and verify equilibration to the target temperature.

C. Production runs and analysis

C1) **Velocity distribution.** Implement creation and on-the-fly updating of the particle-velocity histogram. Plot and discuss the histogram for NVE and NVT runs of n-butane.

C2) **Conformational analysis.** Analyze the distribution of the torsion angle ϕ. Identify trans ($180°$) and gauche ($\pm 60°$) populations. Think yourself of a suited analysis method. (Note the energy barriers between the states is quite high. You may need to run for a long time to get good statistics.)

C3) **Diffusion coefficient.** Implement creation and on-the-fly updating of the MSD of molecular centers of mass. From the long-time slope extract the self-diffusion coefficient D in both internal and SI units.

Deliverables

1. Report (PDF, max 8 pages + appendix) with methods, parameter table, plots, and a "lessons learned" box.
2. Code archive (zip) with `.c` and `.h` files, and a `README.md`. No binaries.