Report – Assignment 3

* Group D: Jongtae Park, Siyuan Huang
* Pull request: <https://github.com/Eismcsquared/MolSim/pull/3>
* Ideas/Design Decision
  + We use C++/Tree mapping for xml file input, because it is easier for coding. Since the xml schema is not very large, this should not cause large performance issues.
  + Since the particle’s motion is influenced by particles within a specified distance, called the "cutoff", the cell size must be at least as large as the cutoff radius. To keep the cell size constant, we take the smallest number of that is greater equal the cutoff radius and divides the domain size as the cell size.
  + We use halo cells to handle boundary conditions. Depending on whether the boundary condition is “outflow” or “reflecting”, the particles are either removed from the domain or the sign their velocity is changed.
  + The system is made extensible by modifying the XML configuration to support the choice between the linked cell algorithm and the direct sum algorithm, as well as providing various particle sets for simulation.
* Implementation
  + **Cell:** A cell represents a subsection of the domain divided based on a specific range, storing indices of particles that fall within its boundaries. Since only particles within the cutoff distance interact with each other, the size of each cell along any axis should be greater than or equal to the cutoff distance for efficiency. If the domain dimensions are not divisible by the cutoff, cell sizes may slightly exceed the cutoff distance to cover the entire domain. Instead of directly storing particle objects, cells store the indices of particles in the particle container, optimizing memory usage and computation.
  + **LinkedCellContainer:** The previous ParticleContainer is renamed to DirectSumContainer, which is derived from an abstract class ParticleContainer, together with a new LinkedCellContainer that implements the linked cell algorithm. The LinkedCellContainer is a specialized container designed for interactions between particles only within the cutoff distance. If particles move beyond the domain boundaries, they are stored in a halo cell. The behavior of particles in the halo cell depends on the boundary condition strategy: Reflect: Particles are adjusted to reflect into the domain. Outflow: Particles are effectively removed from the domain.
  + Force Calculation: Force calculations are also performed on a per-cell basis:
    1. Select the target cell to update.
    2. Extract neighboring cells adjacent to the target cell.
    3. Within these neighboring cells, consider only particles whose distances to the particles in the target cell are within the cutoff range. The particle indices meeting this criterion are used for force calculations.
    4. Finally, update the forces for particles within the target cell itself, ensuring that interactions are computed efficiently and without redundancy.
  + **BoundaryCondition**: The BoundaryCondition is implemented as an enumeration type that specifies the behavior of each domain boundary. Each boundary can independently be set to either:
* Reflecting: Particles bounce back into the domain.
* Outflow: Particles exit the domain and are no longer part of the simulation.
* Results
  + Efficiency of Linked Cell Algorithm: The primary advantage of the Linked Cell Algorithm is achieving satisfactory results in less time without significant loss of computational accuracy. In the Lennard-Jones force model, particles beyond the cutoff distance have a significantly weaker influence compared to those within the cutoff range. By excluding interactions with particles beyond the cutoff, this algorithm provides both speed and accuracy for simulations.
  + Benchmark Performance: Benchmarks were conducted on cuboids with 1000, 2000, 4000, and 8000 particles. Direct Sum Algorithm: The computation time per iteration increased approximately 4 times as the number of particles doubled, as respected (Times observed each: 0.092s, 0.372s, 1.50s, 5.97s). This reflects calculation time growing quadratically as data grows. Linked Cell Algorithm: The computation time per iteration grew slightly with increasing particle counts. (Times observed each: 0.018s, 0.023s, 0.0324s, 0.0510s). This modest growth is due to limiting calculations to neighboring particles within the cutoff, showing that the algorithm scales efficiently.
  + Why delta to has to be small enough in order to guarantee the stability of the method: If 𝛥𝑡 is too large, particles may move significantly in one step, potentially bringing them very close to the boundary. This can result in the following issues in the ghost particle approach: If a particle comes very close to the boundary, it may interact with ghost particles at small distances. And it causes extremely large forces, due to the 1/𝑟12 dependency of the Lennard-Jones potential. This results in inaccurate forces that do not reflect physical reality. It turns out that even with very small time step, no stable simulation is observed. Hence, as stated above, we applied the changing sign approach for the reflective boundary condition.
* Build instruction:
  + mkdir build
  + cd build
  + cmake ..
  + make
  + ./MolSim <input\_file>
* For the collision simulation, run the program with the following parameters:
  + ./MolSim ../input/assignment3.xml
* For the falling drop simulation, run
  + ./MolSim ../input/falling\_drop.xml
* Generating document
  + make doc\_doxygen

