# Lennard-Jones Molecular Dynamic’s Simulation Software

## Introduction

Computer science applications are causing significant improvements in a broad range of areas like medicine, biology, robotics, and entertainment. Problems that a few time ago were thought to be intractable (e.g., the folding of proteins) are getting unexpectedly good results; and everyday new use cases for computer science tools and techniques are being found and deployed.

Even though the applications of computer science in mathematics and physics date back to the very birth of computing, the current state of relatively unexpensive hardware opens a new set of opportunities. Particularly, software applications that once were too machine intensive for a personal computer to run them now are not. It is this context that the current project, which will implement a very specific type of software (i.e., a fluid simulation system), is relevant.

The type of fluid simulations we will implement are the Lennard Jones fluid simulations. This is a way of simulating the behavior of liquids and gases that considers the forces between the particles that make up the fluid. These forces are called intermolecular forces, and they play a big role in how liquids and gases behave. It is important to state that this type of simulations include a number of particles which keeps the system out of the macroscopic order.

The Lennard Jones fluid simulations are based on a mathematical equation that describes the intermolecular forces between particles. This equation takes into account the distance between particles, as well as other properties of the particles themselves. By using this equation, the simulation can predict how the fluid will behave in different situations. For example, the Lennard Jones fluid simulation can be used to study how a gas will expand in a container. It can also be used to study how a liquid will behave when it is heated or cooled.

The goal of this project is to provide scientists and engineers a tool to understand how liquids and gases behave. This tool could be used in many different fields, such as engineering, chemistry, and physics. With it, scientists can study the behavior of liquids and gases in a safe and controlled environment, without the need for expensive and time-consuming laboratory experiments.

## Software requirements

Functional requirements for the Lennard-Jones simulation system are described in the following table.

|  |  |  |
| --- | --- | --- |
| ID | Title | Description |
| FR01 | Inputting simulation parameters | The user should be able to specify the number of particles, the size of the simulation box, the initial configurations for positions (random or cubic lattice) and velocities (random or Boltzmann distribution), the Lennard-Jones potential parameters, the time step size, the number of timesteps to simulate, the time steps needed to output partial results, the simulation ensemble (NVT, NVE). |
| FR02 | Periodic boundary conditions | The software should implement periodic boundary conditions which also imply the use of the minimum image convention. |
| FR03 | Calculation of forces | During the simulation the software should be able to calculate the forces on each particle due to the Lennard-Jones potential. |
| FR04 | Time-stepping algorithm | In order to perform the simulation, the software should include the time-stepping algorithm Velocity Verlet to update the positions and velocities of the particles based on the forces calculated in the previous step. |
| FR05 | Outputting simulation data | The software should be able to output the positions, velocities and energies of the particles at regular intervals during the simulation, as well as the following statistical properties of the system: pressure, temperature, and the radial distribution function. |
| FR06 | List of neighbors | The software should implement a Verlet list which makes computation more efficient but could cause small artifacts in energy conservation when using and NVE ensemble for the simulation. |
| FR07 | User-friendly interface | The software should have a user-friendly interface that allows the user to easily input simulation parameters, view simulation progress, and access analysis tools. Although particle positions will be saved at regular intervals, as stated in FR05, this project will not implement trajectory visualization tools. |
| FR08 | Documentation | The software should be well documented to guide the user on how to use it. |

Nonfunctional requirements for the Lennard-Jones simulation system are described in the following table.

|  |  |  |
| --- | --- | --- |
| ID | Title | Description |
| NFR01 | Performance | The software should have good performance and be able to handle large numbers of particles and long simulation times in a reasonable amount of time. |
| NFR02 | Accuracy | The software should produce accurate results for all calculated quantities. |
| NFR03 | Robustness | The software should be robust and able to handle unexpected inputs and runtime errors without crashing. |
| NFR04 | Scalability | The software should be able to handle an increasing number of particles and longer simulation times as needed. Also, it should be designed in such a way that addition of new functionalities (e.g., the capability of simulating other kinds of particles) could be easily implemented. |
| NFR05 | Maintainability | The software should be easy to maintain and update and should include detailed documentation of its design and implementation. |
| NFR06 | Interoperability | The software should be able to import and export data in a variety of formats, making it easy to use in conjunction with other simulation software or analysis tools. |

## Glossary

### Lennard Jones fluid:

A theorical fluid whose particles interact only via the Lennard Jones potential, *,*  given by:

where is the distance between two given particles, is the potential’s well depth and is the radii of the particles. This potential obeys the superposition principle.

### Boltzmann Distribution:

The Boltzmann velocity distribution, also known as the Maxwell-Boltzmann velocity distribution, is a probability distribution that describes the speed of particles in a gas at a given temperature. It is based on the Boltzmann distribution, which describes the distribution of energy among the particles in a system.

### Minimum image convention

It is a technique used in computational simulations of periodic systems. It is based on the idea that the simulation box is replicated an infinite number of times in all directions, and that the interactions between particles are computed only between the closest images of the particles in the simulation box.

### NVE ensemble

It is a type of statistical mechanics ensemble that stands for "constant number of particles, constant volume, and constant energy". In this ensemble, the number of particles, the volume of the system and the total energy of the system are held constant, while the system is allowed to exchange energy with its surroundings via collisions with particles. The internal energy of the system can fluctuate, but the total energy remains constant.

### NVT ensemble

It is a type of statistical mechanics ensemble that stands for "constant number of particles, constant volume, and constant temperature". In this ensemble, the number of particles, the volume of the system, and the temperature of the system are held constant, while the system is allowed to exchange energy with its surroundings via collisions with particles. The internal energy of the system can fluctuate, but the temperature remains constant. This ensemble is useful for studying systems that are closed to mass and volume.

### Periodic boundary conditions

It’s a type of boundary constraint that can be implemented by applying a geometric transformation to the position of the particles that are at the edge of the box to make them reappear at the opposite side, in such a way that the simulation box is considered as a torus. This way the interactions between all the particles within the box can be calculated as if the box were infinite.

### Radial Distribution Function

It is a statistical measure used in computational simulations and experiments to quantify the spatial distribution of particles in a system. It describes the probability of finding a particle at a certain distance from a reference particle, and is commonly used to analyze the structure of liquids, solids, and other condensed matter systems. It is a useful tool for characterizing the local order and symmetry of a system, and can be used to identify phase transitions.

### Velocity Verlet Algorithm

It is a numerical integration scheme that updates the positions and velocities of particles in a molecular dynamics simulation. It uses the current positions, velocities and forces to calculate the new positions and velocities of the particles at each time step. The algorithm is a combination of the velocity update and the position update, where position update is done using the current velocity and the force update is done using the new position.

### Verlet list

It is a list of all the particles that are within a certain cutoff distance of a given particle. This list is typically updated every few time steps, and the cutoff distance is chosen to be slightly larger than the range of the inter-particle interactions.

## Design

Given the nature of this project, some technical compromises must be made. One issue that must be addressed is how computationally and time-intensive these applications can get as the number of particles increases. Thus, the need for choosing a programming language that can handle those loads in the most efficient manner possible is evident. Hence, we have decided to use the imperative programming paradigm, as the languages under it are designed for high performance and efficiency, making it easier to write code that can model complex systems like molecular dynamics simulations.

Considering the paradigm constraint, which we have just described, we decided that the design tool that is better tailored for our system is the flowchart as it provides a concise visual representation of the logic of the program which can be understood by the technical team but also by the clients. Thus, creating a common starting point to discuss and resolve issues.

### Highest abstraction level of the program

The flowchart on Figure 1 describes the most abstract representation for the happy path of the molecular dynamics system

Imagen en blanco y negro de una señal

Descripción generada automáticamente con confianza media

Figure 1

Figure 2 provides a more detailed representation of the system without getting into the nuances of how each component will be implemented.

Diagrama

Descripción generada automáticamente

Figure 2

## Resources

Implementation Plan:

The implementation phase of the Lennard Jones Molecular Dynamics software will last 6 months. During this time, the team of three developers will work together to implement all the functional requirements of the software. The following is a detailed plan for the implementation phase.

1. Month 1: The first month will be dedicated to setting up the development environment and familiarizing the developers with the software architecture and design.
2. Month 2-3: During the second and third months, the developers will work on the simulation engine, including implementing features such as the calculation of forces and energies, and integrating the simulation engine with the data structures that store and manipulate the molecular data.
3. Month 4-5: In the fourth and fifth months, the developers will focus on implementing the user interface and the visualization component of the software. They will also implement features such as the ability to read and write molecular data from and to files.
4. Month 6: During the final month, the developers will complete any remaining tasks and perform a final round of testing to ensure that the software meets all the functional requirements. They will also prepare the software for deployment, including writing any necessary documentation.

Testing Plan:

The testing phase of the Lennard Jones Molecular Dynamics software will be an ongoing process that begins in the first month of the implementation phase and continues throughout the six months. The following is a general plan for the testing phases:

1. Unit Testing: Unit tests will be written for each component of the software to ensure that each component functions correctly in isolation. Unit tests will be performed after each component is implemented to ensure that any bugs are caught early in the development process.
2. Integration Testing: Integration tests will be performed after all the components of the software are integrated to ensure that the components work together correctly. Integration tests will verify that the simulation engine produces accurate results, the user interface is user-friendly, and the visualization component provides meaningful and accurate representations of the molecular data.
3. System Testing: System tests will be performed on the final version of the software to ensure that it meets all the functional requirements. System tests will include performance tests to ensure that the software runs efficiently and accuracy tests to ensure that the simulation results are accurate.

Workload Distribution:

The development team will consist of three intermediate programmers. To ensure that the workload is equitably distributed among the three developers, each developer will be assigned specific modules or functionalities to work on. The project manager will work with the developers to ensure that each developer's skills and interests are taken into consideration when assigning tasks. The project manager will also monitor the progress of each developer and make any necessary adjustments to the workload distribution if one developer is becoming overwhelmed or if one developer is underutilized.

## Budget

The project is estimated to cost $25k USD, with $15k allocated for the development team's salaries, $5k as a reserve for unexpected costs, and $5k for management costs.