Universidad de Valladolid

Towards Long-Timescale Molecular Dynamics of Cytoplasmic Subdomains: Modeling Crowding, Polydispersity, and Pairwise Interactions

María Ximena Vargas Chaverri

LAAS-CNRS, Toulouse, France
Universidad de Valladolid
Valladolid, Spain

Supervisor: Antoine Jay LAAS-CNRS,
Morgan Delarue LAAS-CNRS, and
María del Pilar Redondo Cristóbal UVA

June 2025

Acknowledgements

I would like to thank my supervisors, Antoine Jay and Morgan Delarue, for their trust in me, which allowed me to carry out my internship at the LAAS-CNRS research center in Toulouse, France. I appreciate their guidance throughout the process and their availability to address questions and provide support when needed.

I am also grateful to LAAS-CNRS for providing the necessary resources to carry out my work, including access to high-performance computing facilities.

I would also like to thank my supervisor at the University of Valladolid, María del Pilar Redondo Cristóbal, for her continuous support and for always being helpful and patient with administrative matters since the beginning of the Master's program.

Además, me gustaría expresar mi agradecimiento por el apoyo constante que he recibido a lo largo de todos mis años de estudio por parte de mi familia: mi papá, mi mamá y Diego. Por estar presente en todo momento y brindarme su apoyo incondicional.

También agradezco profundamente a mi prometido Danilo. Su cariño, paciencia y confianza incondicional han sido fundamentales para mantenerme motivada.

Universidad de Valladolid

LAAS CNRS

Contents

Abstract

The cell cytoplasm is a crowded environment that is host to a variety of macromolecules. The motions of these molecules deviate from simple Brownian statistics, yet the physical origin of the heavy-tailed, non-Gaussian step-length distributions reported in the live-cell experiments remains unclear. This work presents a fully automated, colloidal molecular dynamics framework in LAMMPS to assert whether size polydispersity by itself can cause such heterogeneity. Proteins and complexes are represented as rigid, colloidal spheres (log-normal distributed radii from 2 to 40 nm) and are packed at 25-45% volume fractions and evolved for up to 0.5 ms with Langevin dynamics. A Python code is developed that fully automates the process of creating a simulation setup, from generating overlap-free initial configurations, to calculating all $N(N+1)/2$ cutoff distances (where N is the number of different radii present in the simulation) for the colloid pair coefficients. Strong-scaling benchmark and efficient neighbor list parameters optimization resulted in reaching a performance of $\approx 420\,\mu s$ of simulation time per one day of real time.

When the colloid potential Hamaker constant is set to $10^{-5}\,eV$ the colloids behave as hard spheres: diffusion coefficients follow the inverse-radius Stokes-Einstein trend and retain Brownian motion statistics across all crowding levels. Raising A above $0.1\,eV$ introduces short-range attraction that leaves small particles almost unchanged but doubles the long-time diffusivity of the largest particles and, crucially, produces the exponential tails in the log-probability of the step-lengths - mirroring the experimental results. Thus, polydispersity and steric crowding alone insufficient. Weak inter-colloidal attractions are likely essential for the observed cytoplasmic heterogeneity.

Resumen

El citoplasma celular es un entorno congestionado que alberga una amplia variedad de macromoléculas. Los movimientos de estas moléculas se apartan de las estadísticas brownianas simples; sin embargo, el origen físico de las distribuciones de longitudes de paso con colas pesadas y no gaussianas observadas en los experimentos *in vivo* permanece sin esclarecerse. Este trabajo presenta un marco de dinámica molecular coloidal totalmente automatizado en LAMMPS para determinar si la mera polidispersidad de tamaños puede generar tal heterogeneidad. Las proteínas y los complejos se representan como esferas coloidales rígidas (radios distribuidos log–normalmente de 2 a 40 nm), se empaquetan a fracciones de volumen del 25–45 % y se hacen evolucionar durante hasta 0.5 ms mediante dinámica de Langevin. Se desarrolló un código en Python que automatiza completamente la creación de la configuración de simulación, desde la generación de configuraciones iniciales sin solapamientos hasta el cálculo de las $N(N+1)/2$ distancias de corte (donde N es el número de radios distintos presentes) para los coeficientes de pares coloidales. Las pruebas de escalado fuerte y la optimización eficiente de los parámetros de las listas de vecinos permitieron alcanzar un rendimiento de $\approx 420\,\mu$s de tiempo de simulación por día de tiempo real.

Cuando la constante de Hamaker del potencial coloidal se fija en 10^{-5} eV, los coloides se comportan como esferas duras: los coeficientes de difusión siguen la tendencia de Stokes–Einstein inversa al radio y mantienen estadísticas brownianas en todos los niveles de hacinamiento. Elevar A por encima de 0.1 eV introduce una atracción de corto alcance que apenas afecta a las partículas pequeñas, pero duplica la difusividad a largo plazo de las partículas más grandes y, de forma crucial, produce colas exponenciales en el logaritmo de la probabilidad de las longitudes de paso, reproduciendo los resultados

experimentales. Por lo tanto, la polidispersidad y el hacinamiento estérico por sí solos son insuficientes: las débiles atracciones intercoloidales son probablemente esenciales para la heterogeneidad observada en el citoplasma.

1 Introduction

A cell is the most basic unit of life in all living organisms, with the exception of viruses.[1] Throughout evolution, life has progressed from simple unicellular forms to increasingly complex multicellular organisms. This transformation has involved both the multiplication of cells and the specialization of their functions. As a result, the diversity of life seen today is deeply rooted in the structure and behavior of cells. While cells can vary widely between species and among tissues within the same organism, they share fundamental characteristics that support life. Gaining a clear understanding of how a cell works is essential to grasp how organisms develop, survive, and respond to their environment. Cellular activity underlies all physiological processes, and even small disruptions at the cellular level can lead to significant consequences.[1] For these reasons, the study of cells is fundamental to understanding life at its most essential level. Although the term "cell" is familiar to most people and widely employed across many disciplines, the cell itself remains an active subject of research; several of its properties and the phenomena that occur within it are still not fully understood.[2–7] continue to investigate basic questions, such as how molecules move through the crowded environment of the cytoplasm. These studies reveal that even the most familiar aspects of cell biology involve complex dynamics that are not yet fully explained. Gaining a deeper understanding of how cells function is key to explaining how organisms maintain balance, how diseases begin at the cellular level, and how living systems respond to their surroundings. This knowledge can be of great value in fields such as scientific research in health, development, and biological function, among others.

Inside a cell, the cytoplasm is the space where most biochemical reactions take place. It lies between the membrane and the organelles, and it might seem like a simple fluid at first, but it is actually very crowded and much more complex than it was once thought to be. This space is filled with proteins, nucleic acids, and many other large molecules that are constantly moving and interacting. Because of this, molecules cannot travel freely. Their movement is affected by how much space is available and by interactions such as attraction or repulsion based on their charge. This makes the environment inside the cytoplasm very different from a dilute solution. Molecules might move slowly, follow unusual paths, or even get temporarily stuck. These conditions influence how reactions happen and how efficiently the cell can carry out its functions. The structure of the cytoplasm is not fixed either. It can change depending on what the cell needs to do at a given time. The way molecules are packed also creates differences in how fast or slow they move from one area to another. Some areas inside the cytoplasm may allow easier movement, while others act more like barriers. This uneven movement adds another layer of complexity to how the cell manages its internal processes.[2]

As outlined earlier, the cytoplasm has been shown in multiple studies to consist of a diverse range of elements distributed throughout its aqueous environment.[8] Continuing with the ideas described before, it can be said that the density in the cytoplasm reflects the combined presence of small molecules, such as osmolytes, and larger macromolecules. While both contribute to the physical properties of the cytoplasm, macromolecules are considered to account for a substantial portion of its overall density under normal physiological conditions.[2,9]

The concentration of macromolecules within the cytoplasm influences how substances move at intermediate scales by limiting the available space for diffusion.[2] For this reason, particles in the range of 2 nm to 40 nm radius are the focus of this work. Other recent works,[2–4] in mostly experimental ways, have studied this range of sizes of particles; for example, in fission yeast, 40 nm particles move significantly slower under crowded conditions, while in E. coli, smaller particles around 20 nm tend to

accumulate in the nucleoid and larger ones, such as 50 nm, are excluded. This size-dependent behavior suggests that particle distribution inside cells is not random but shaped by physical properties like size, charge, and crowding. These factors directly affect how molecules move, interact, and localize within the cytoplasm.[2–4]

Although this is a field in constant development, given its potential to provide explanations and interpretations of various biological phenomena, it still presents several challenges. So far, most studies have been carried out experimentally. While these efforts have yielded valuable insights, they are not without limitations. Despite the availability of advanced methodologies well fitted to specific cellular phenomena, broader investigations comparing variations across cell types and organisms remain limited.[2]

Moreover, there is a growing demand for the development of computational models capable of shedding light on the physical and causal mechanisms behind the dynamic properties of the cytoplasm observed in experiments. There are open-source simulation software such as LAMMPS or GROMACS, which of course open new possibilities in this area. While the idea of creating computational systems to model certain attributes or effects is not entirely new[4,10] , it remains a field with significant unexplored potential when applied to complex biological environments like the cytoplasm. There are many advantages to simulating such systems, but one of the most compelling—especially in the context of this work—is the ability to visualize particles and dynamic phenomena that are currently beyond the reach of experimental techniques.

Experiments conducted by the MILE group in LAAS-CNRS have reported non-Brownian particle displacements in the cytoplasm of cells from diverse species (see Fig.1). The resulting heavy-tailed distribution points to some underlying heterogeneity, but its origin is still uncertain. One hypothesis explored here is that this heterogeneity could be linked to cytoplasmic polydispersity—that is, the broad range of macromolecular sizes present within the cell.

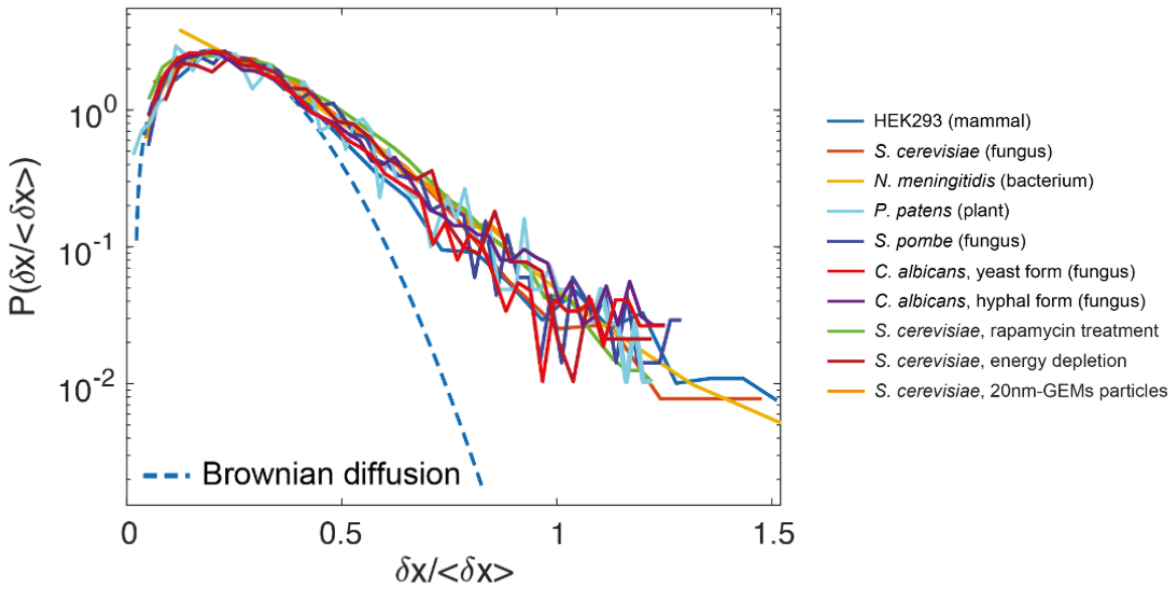


Figure 1: **Universal non-Gaussian shape of intracellular step-length distributions.** Log–lin plot of the probability density $P(\delta x/\langle\delta x\rangle)$ for one-frame displacements δx of 40 nm fluorescent GEM probes in the cytoplasm of very different organisms and metabolic states (solid Coloured curves; species/conditions listed at right). Displacements are normalized by the corresponding mean step length $\langle\delta x\rangle$ so all curves are dimensionless and directly comparable. The dashed blue line is the Rayleigh form expected for purely Brownian (thermal) diffusion in two dimensions. Every biological condition deviates strongly from the Brownian reference, showing a heavy-tailed distribution that is nevertheless conserved in shape across species ranging from bacteria to plants and mammalian cells, as well as under chemical (rapamycin) and energetic perturbations. This collapse onto a single master curve highlights a common, active and heterogeneous character of cytoplasmic dynamics.

1.1 Objective

The purpose of this work is to initiate a theoretical investigation of the cytoplasm by developing a simplified computational model using LAMMPS. This model is based on classical molecular dynamics and simulates colloids to represent macromolecules ranging from 2 nm to 40 nm, distributed according to a log-normal profile. The default colloid input settings are modified to more accurately mimic a small, biologically relevant portion of the cytoplasm—excluding the nucleus—to manage complexity while preserving physical accuracy. Both monodisperse and polydisperse systems are considered, with particular emphasis on long-timescale simulations in the millisecond range. These extended simulations are meant to be possible through the use of high-performance computing resources. Additionally, a systematic analysis of the simulation potential and its key parameters will be carried out to gain a better understanding of the system's physical properties and emergent behavior under varying conditions.

The central question is whether introducing polydispersity into this confined patch of cytoplasm can reproduce the exponential tail observed in Figure 1. If the simulated step-length distribution does (or does not) show such a tail, the result will help evaluate—and potentially eliminate—one of the proposed explanations for the cytoplasm's heterogeneous dynamics.

General steps to reach the goal

1. **Assess molecular-dynamics methodology**
   Review the relevant literature and clarify what classical MD can—and cannot—capture for the cytoplasmic system of interest.

2. **Select an MD platform and force-field package**
   Work with colloidal systems

3. **Build and test initial input files**
   Familiarize oneself with key parameters and run short, exploratory simulations to verify basic stability.

4. **Refine parameters and automate setup**
   Tune interaction parameters, customize input scripts, and develop Python utilities to streamline configuration, enabling trajectories that reach the millisecond timescale required for this project.

5. **Optimize computational performance**
   Conduct benchmarking runs and iterative adjustments to maximize efficiency and minimize computational cost without sacrificing physical fidelity.

6. **Perform preliminary "sanity-check" analyses**
   Extract diffusion coefficients from short simulations and compare them with theoretical expectations (e.g. via the Stokes–Einstein relation).

7. **Run extended simulations and analyze outputs**
   Generate long trajectories, compute step-length distributions, and determine whether polydispersity in the model can reproduce the exponential tail observed experimentally, thereby testing the core hypothesis.

2 Theoretical background

This section outlines the fundamental concepts and theoretical principles that support the development of this work. It includes the necessary background to understand the methodologies applied and provides context for the computational strategies used in the later sections. The theoretical framework presented here serves as the foundation for the interpretation of the results and the justification of the chosen approaches.

2.1 Molecular dynamics (MD)

Molecular Dynamics (MD) is a powerful computational technique used to study the behavior of matter at the atomic and molecular level. It simulates the movement of atoms based on their initial positions and velocities. By simulating the interactions between particles over time, MD helps understand how materials behave under different conditions. Instead of performing experiments in a laboratory, it uses computers to model systems and observe how atoms and molecules move and interact.[11]

This method is especially useful when studying systems that are difficult or impossible to examine experimentally. MD is based on classical physics, where Newton's laws are applied to many-body systems to calculate the motion of particles. With the rise of computational power, MD has become an essential tool in physics, chemistry, biology, and materials science.[11]

2.1.1 Equations of motion

The motion of atoms in a system is governed by Newton's second law, which relates the force acting on a particle to the time derivative of its momentum:[12]

$$F_i = \frac{dp_i}{dt} \tag{1}$$

Here, F_i is the force and p_i is the momentum of atom i. For conservative systems, the force is also defined as the negative gradient of the potential energy V with respect to position:

$$F_i = -\frac{dV}{dr_i} \tag{2}$$

Equating the two expressions yields the first Hamilton equation:

$$\frac{dp_i}{dt} = -\frac{dV}{dr_i} \tag{3}$$

The second Hamilton equation is derived by differentiating the kinetic energy T_i with respect to the momentum p_i. Given that $T_i = \frac{p_i^2}{2m_i}$, where m_i is the mass of the particle, the result is:

$$\frac{dT_i}{dp_i} = \frac{d}{dp_i}\left(\frac{p_i^2}{2m_i}\right) = \frac{p_i}{m_i} = v_i = \frac{dr_i}{dt} \tag{4}$$

These two equations describe the time evolution of both the coordinates and momenta of atoms in a molecular system.

2.1.2 Time integration algorithm

In 1967, Loup Verlet introduced a numerical integration scheme based on the central differences method that is still widely used in molecular dynamics (MD) simulations today.[13] This method is appreciated for its simplicity and for conserving total energy over long simulations.

The Verlet algorithm is derived by doing a Taylor expansion of the position vector $\mathbf{r}(t)$ forward and backward in time:

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{5}$$

$$\mathbf{r}(t-\Delta t) = \mathbf{r}(t) - \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{6}$$

Adding Equations 5 and 6 eliminates the velocity term:

$$\mathbf{r}(t+\Delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\Delta t) + \mathbf{a}(t)\Delta t^2 + \mathcal{O}(\Delta t^4) \tag{7}$$

This is the Verlet algorithm. It is symmetric and time-reversible, and it shows good energy conservation over long simulations. However, the velocity is not explicitly included. If needed, it can be estimated by using:

$$\mathbf{v}(t) = \frac{\mathbf{r}(t+\Delta t) - \mathbf{r}(t-\Delta t)}{2\Delta t} + \mathcal{O}(\Delta t^2) \tag{8}$$

One limitation is that the Verlet algorithm is not self-starting; it needs two previous positions to begin the simulation. To fix this, the Velocity Verlet algorithm was introduced.[14] This version explicitly updates both the position and velocity at each time step. This algorithm is given by two main equations:

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \mathbf{a}(t)\frac{(\Delta t)^2}{2} \tag{9}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{1}{2}(\mathbf{a}(t) + \mathbf{a}(t+\Delta t))\Delta t \tag{10}$$

This algorithm requires only position, velocity and acceleration at a time t. Each timestep, position is updated based on current velocity and acceleration. This is followed by recalculating forces and thus accelerations at the new positions. Finally, velocity is updated using the average acceleration over the timestep.

2.1.3 Force fields

In classical molecular dynamics, a force field is a mathematical model that defines how the potential energy of a system depends on the spatial configuration of its constituent particles. This energy is typically expressed as a function $U(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N)$, and it incorporates various parameters that describe the nature of interparticle interactions. These parameters are often derived from quantum mechanical computations or are adjusted to reproduce experimental data obtained from techniques such as X-ray or neutron diffraction, Raman and infrared spectroscopy, or nuclear magnetic resonance.[15]

The goal of a force field is to approximate the true potential surface of the system using a form

that is computationally tractable yet sufficiently accurate to capture the physical properties of interest. In molecular systems, atoms are frequently modeled as masses connected by springs, representing bonds and angles, while non-bonded interactions are commonly handled through Lennard-Jones and Coulombic terms. A typical expression for a classical force field can be written as:

$$\begin{aligned} U = & \sum_{\text{bonds}} \frac{1}{2} k_b (r - r_0)^2 + \sum_{\text{angles}} \frac{1}{2} k_a (\theta - \theta_0)^2 + \sum_{\text{torsions}} \frac{V_n}{2} [1 + \cos(n\phi - \delta)] \\ & + \sum_{\text{improper}} V_{\text{imp}} + \sum_{\text{LJ}} 4\varepsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \sum_{\text{elec}} \frac{q_i q_j}{r_{ij}}, \end{aligned} \tag{11}$$

This general formulation underpins many of the empirical force fields used in classical simulations.[15] This general formulation contains terms that can describe complex molecular motions such as bond stretching, angle bending, and dihedral torsions. However, systems such as colloidal suspensions can effectively be modeled as assemblies of rigid spherical particles. Rather than from explicit chemical bonds, the dominant interactions in such systems arise from excluded volume effects and short- or long-range repulsions or attractions.

In such cases, the pairwise interaction potentials are usually simply a function of interparticle distance and can be chosen to capture certain essential physics such as van der Waals attraction or Coulombic interaction, which is the case with the last two terms in Eq 11.

Lennard Jones Potential

The Lennard-Jones potential is a pair potential widely used in molecular simulations to represent the interaction between atoms or molecules.[16,17] The equation most commonly employed is

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

Also, the graphic representation is

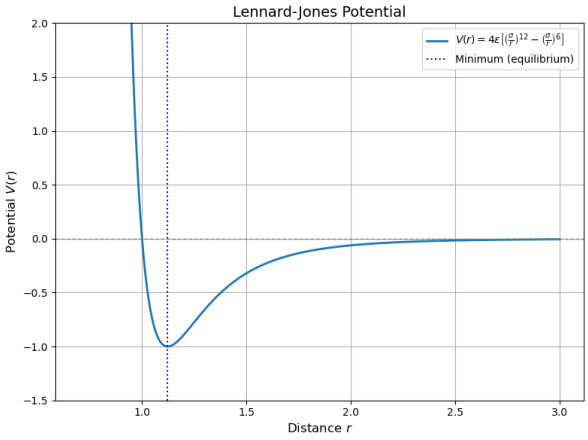


Figure 2: Lennard-Jones potential curve showing the balance between repulsive and attractive forces between neutral atoms. The curve reaches its minimum at the equilibrium distance where the net force is zero. In this plot, the parameters are set to $\varepsilon = 1.0$ and $\sigma = 1.0$.

where $u(r)$ is the potential energy as a function of the distance r between two particles. The

parameter ϵ is the depth of the potential well and σ is the distance at which the potential energy is zero.[17] The r^{-12} term represents short-range repulsion, primarily due to Pauli exclusion, and the r^{-6} term represents the long-range dispersion attraction or van der Waals interaction.[16]

This potential was originally developed to describe the cohesive energy of crystals of noble gases such as argon. After London derived that dispersion interactions decay as r^{-6}, the attractive exponent was set to six, and the repulsive exponent was set to twelve for computational simplicity.[16,17] The Lennard-Jones potential has become a standard model for simple atomic systems and is commonly used for benchmarking and as a reference in the development and testing of simulation methods.[16]

In practice, the Lennard-Jones potential is often truncated at a finite cutoff distance r_c to reduce computational cost. Various truncation and shifting schemes exist, and these can produce significantly different thermodynamic and transport properties, even though the underlying potential is referred to as "Lennard-Jones" in each case. There is therefore no unique standard, and care is required when comparing results across different studies.[17]

The Lennard-Jones potential is well-suited for modeling simple liquids and gases, particularly monatomic systems. However, it is also commonly applied to more complex systems, even when its underlying assumptions may not fully apply. In such cases, more advanced or system-specific potentials may be more appropriate. Still, the Lennard-Jones potential remains widely used due to its simplicity, efficiency, and long-standing role in molecular simulations.[16,17]

Pair style colloid

To accurately describe the interactions between large colloidal particles or anisotropic molecules in simulations, it is very important to consider that each particle comprises many smaller interacting sites, rather than treating them as point particles. In this context, the total interaction energy between two such bodies can be derived by summing, or in the continuum limit, integrating the pairwise Lennard-Jones (LJ) potential over the volumes of both particles. This approach yields the so-called Hamaker potential, which provides a rigorous, parameter-free description of colloidal interactions with a clear microscopic interpretation.[18]

For two spherical particles of radii a_1 and a_2, separated by a center-to-center distance r, the attractive part of the interaction is derived from the van der Waals (r^{-6}) component of the LJ potential and can be written as:

$$U_{\mathrm{A}}(r) = -\frac{A_{12}}{6}\left[\frac{2a_1a_2}{r^2-(a_1+a_2)^2} + \frac{2a_1a_2}{r^2-(a_1-a_2)^2} + \ln\left(\frac{r^2-(a_1+a_2)^2}{r^2-(a_1-a_2)^2}\right)\right] \tag{13}$$

where A_{12} is the Hamaker constant, encapsulating material properties and number densities.[18] The

repulsive part, originating from the short-range (r^{-12}) component of the LJ potential, is given by:

$$
\begin{aligned}
U_R(r) = \frac{A_{12}}{37800}\,\frac{\sigma^6}{r}\Bigg[& \frac{r^2 - 7r(a_1+a_2) + 6\left(a_1^2 + 7a_1a_2 + a_2^2\right)}{(r-a_1-a_2)^7} \\
& + \frac{r^2 + 7r(a_1+a_2) + 6\left(a_1^2 + 7a_1a_2 + a_2^2\right)}{(r+a_1+a_2)^7} \\
& - \frac{r^2 + 7r(a_1-a_2) + 6\left(a_1^2 - 7a_1a_2 + a_2^2\right)}{(r+a_1-a_2)^7} \\
& - \frac{r^2 - 7r(a_1-a_2) + 6\left(a_1^2 - 7a_1a_2 + a_2^2\right)}{(r-a_1+a_2)^7}\Bigg]
\end{aligned}
\tag{14}
$$

where σ is the size of the constituent Lennard-Jones particle, with the size of the colloidal particles being greater than σ, a_1 and a_2 are the radii of two particles, r is the distance from the center of one colloid to the center of the other one.[19]

The total potential function is then given by:

$$
U(r) = U_{\mathrm{A}}(r) + U_{\mathrm{R}}(r) \tag{15}
$$

The Hamaker-derived potentials are rooted directly in the physical properties and geometry of the interacting particles. Specifically, the interaction strength and distance dependence naturally follow from the underlying LJ pair potential, the sizes of the spheres, and the Hamaker constant, without the need for arbitrary fitting or empirical correction factors. This physical basis ensures that the model remains predictive for a wide range of particle sizes and compositions, making it particularly valuable for studies of colloidal suspensions, liquid crystals, and related systems where anisotropy and finite-size effects are important.[18]

Figure 3 shows the colloid potential curves for a pair of particles of radius 20 Å. It can be seen that increasing the value of the Hamaker constant from very small values results in appearance of an attraction well. The value of σ which scales the U_{R} component causes opposite effect. Furthermore, it appears that increasing both A and σ shifts the potential to the right, effectively increasing the distance at which the particles start interacting. It can also be noted that in the limits of very low A and σ one can mimic a hard-sphere potential.

Hard-sphere

The hard-sphere potential (see Fig 4), one of the simplest and most thoroughly developed pair interaction models, serves as a reference system for tuning particle behavior.[20] In the hard-sphere model, each particle behaves as a perfectly rigid sphere: the moment their surfaces touch, they experience an infinitely large repulsive energy, and when they are separated, they are not interacting. This interaction is defined as:

$$
U(h) = \begin{cases} \infty, & h \le 0, \\ 0, & h > 0, \end{cases}
$$

where h is surface-to-surface distance. This means that the only interactions with hard-sphere

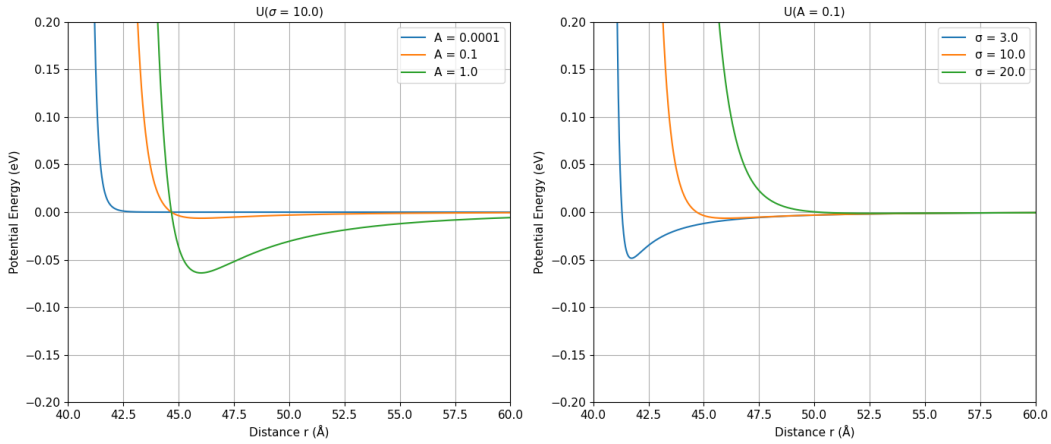


Figure 3: Colloid potential function plotted for two particles of radius 20 Å. Left graph shows curves for value of $\sigma = 10$ Å with varying A, and the right graph shows curves for value of $A = 0.1\,eV$ with varying σ

potential are instantaneous elastic collisions. Although simple by definition, such potential definition is hard to simulate using time-integration, as the potential energy of an interaction can jump from 0 to ∞ in one timestep. This necessitates modeling hard-sphere potential with approximated steep soft repulsion potentials and sufficiently small time-steps. As an alternative to time-integration simulations, event-driven simulations avoid this issue by jumping directly from one collision to the next.

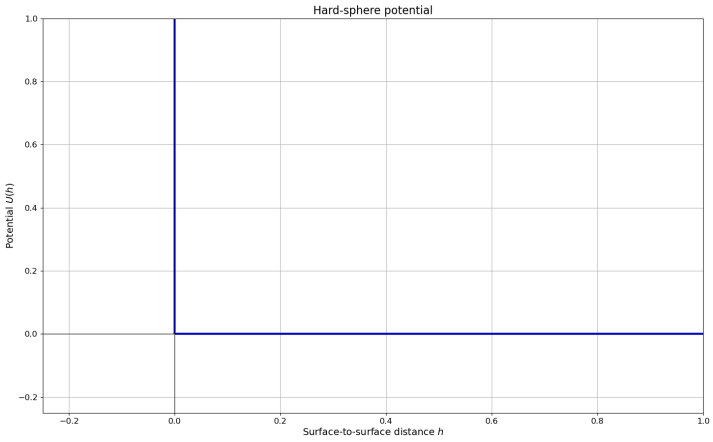


Figure 4: Hard-sphere potential.

2.1.4 Periodic Boundary Conditions

Periodic boundary conditions provide a method for simulating large or infinite systems by using a smaller, repeating segment called a unit cell as it is exemplified in Fig 5. This approach allows the system to be represented as if it extends endlessly in all directions, making complex systems more manageable to study.

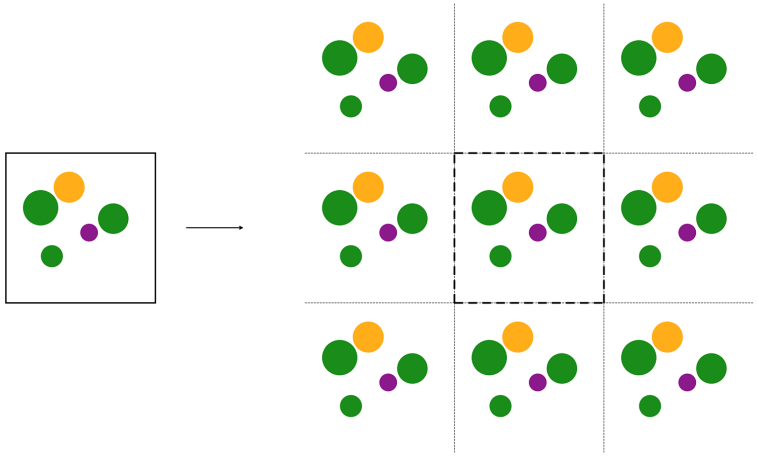


Figure 5: Simple pictorial representation of periodic boundary conditions. A very large or infinite space is considered, which is difficult or impossible to study in its entirety, so a representative sample of it is taken instead.

To implement this idea in practice, periodic boundary conditions are applied to the simulation region. This technique replicates the simulation box in all directions, creating a continuous and uniform system. When an atom leaves one side of the box, it reenters from the opposite side. Atoms near the boundaries interact with those in neighboring copies of the box, which prevents artificial effects caused by physical walls and ensures consistent behavior throughout the system. Atomic positions are adjusted when they move beyond the defined region to maintain correct calculations of distances and interactions. The simulation box is often shaped as a rectangular prism, which is a three-dimensional object with six rectangular faces and right angles between all sides. It is defined by specific lengths along the x, y, and z directions and is commonly used for its simplicity in numerical implementation. However, other geometries such as hexagonal or truncated octahedral cells may be chosen to reduce surface effects or better match the symmetry of certain structures. Although periodic boundaries greatly improve the physical realism of simulations, some finite-size effects may still occur, particularly in systems with long-range interactions or strong spatial correlations. These effects should be carefully considered during both the setup and analysis of the simulation.[11]

2.1.5 Temperature in molecular dynamics simulations

Molecular dynamics simulations are often described as computational experiments because they follow a procedure similar in spirit to laboratory-based investigations. To begin, one defines a model system composed of a set of interacting particles, assigns physical parameters such as mass and initial velocity, and allows the system to evolve according to classical mechanics. The evolution is governed by Newton's second law, and trajectories are generated by integrating the resulting equations of motion over small

time steps.[12]

Just as in physical experiments, measurements obtained from simulations require care and planning. A poorly prepared initial state, or insufficient relaxation time, can lead to incorrect conclusions. Before extracting physical quantities, it is necessary to ensure that the system has reached equilibrium and that any transient behaviors have subsided. Furthermore, to minimize the effects of random fluctuations, time averaging over long trajectories is typically required. The calculation of thermodynamic observables in MD relies on quantities that are accessible from the simulation output, such as particle positions and velocities. One central example is temperature. In a classical system, temperature is defined through the kinetic energy of the particles. According to the equipartition theorem, each quadratic degree of freedom contributes an average energy of $\frac{1}{2}k_BT$ where k_B is the Boltzmann constant and T is the temperature. This leads to a practical expression for computing the instantaneous temperature based on particle velocities.[12] For a system with N particles and of N_f degrees of freedom, the temperature at a given time t is estimated by using:

$$T(t) = \frac{1}{k_B N_f} \sum_{i=1}^{N} m_i v_i^2(t) \tag{16}$$

Because particle velocities fluctuate throughout the simulation, so does the instantaneous temperature. These fluctuations are statistical in nature and tend to diminish with larger system size, scaling inversely with the square root of the number of degrees of freedom. In practice, averaging the temperature over many time steps provides a stable and reliable estimate.[12]

2.1.6 Ensembles

In classical molecular dynamics, energy is conserved as it transitions back and forth between potential and kinetic energy, keeping the total constant throughout the simulation.[21] Although the total energy remains conserved, it is constantly exchanged between particles within the system. As a result, multiple microscopic configurations can exist that share the same total energy.[21] These configurations collectively form what is known as an ensemble — a conceptual collection of a large number of identical systems, each representing a possible microstate consistent with the same macroscopic conditions.[22]

Once the idea of an ensemble is established, it becomes essential to distinguish between different types based on the physical constraints imposed on the system. The microcanonical ensemble describes an isolated system with fixed energy E, volume V, and particle number N. Under these conditions, no exchange of energy or particles with the surroundings is allowed, and all accessible microstates are considered equally probable. The entropy S is related to the logarithm of the number of microstates, and thermodynamic quantities such as temperature T and pressure P are obtained through appropriate derivatives of S.[22]

The canonical ensemble applies to systems in thermal equilibrium with a heat reservoir at a fixed temperature T, allowing energy exchange while keeping V and N constant. As a result, the system's energy fluctuates, and microstates are no longer equally probable. Instead, their probabilities follow the Boltzmann distribution, which decreases exponentially with energy. The *canonical partition function* $Q(T, V, N)$, from which all thermodynamic properties can be derived, including the Helmholtz free energy A.[22]

The grand canonical ensemble extends this framework to open systems that can exchange both energy and particles with external reservoirs. These systems are characterized by constant temperature

T, volume V, and chemical potential μ. Here, both the energy and the particle number fluctuate. The probability of a particular microstate depends on both its energy and particle number, weighted by the *grand partition function* $\Xi(T, V, \mu)$. This leads to the grand potential Ψ, from which one can compute pressure, entropy, and average particle number.[22]

Each ensemble corresponds to a distinct thermodynamic potential and is suited to particular experimental or theoretical conditions. Despite their differences in formulation, all ensembles yield equivalent thermodynamic predictions in the thermodynamic limit, making them flexible and powerful tools in statistical mechanics.[22]

In summary, molecular dynamics naturally is *NVE*.[23] However, to more closely reproduce experimental conditions, it is often desirable to control the temperature of the system. For this reason, methods that allow simulations to sample an NVT ensemble are introduced. This work focuses on both of these ensembles.

There are many different thermostats available in LAMMPS. In the present work, the only two different ones to be discussed are: Langevin (also for more about Langevin dynamics see section 2.2) and Nosé-Hoover.

2.1.7 Thermostat: Langevin and Nosé-Hoover

Langevin

To simulate systems at constant temperature, the Langevin thermostat is commonly employed in classical molecular dynamics (*MD*). This approach modifies the equations of motion by introducing additional forces that account for thermal effects due to collisions with an implicit heat bath. In contrast to the conservative form of Newton's second law given in Eq. 3, the Langevin equation adds a friction force and a random force to the momentum update:

$$\frac{dp_i}{dt} = -\frac{dV}{dr_i} - \gamma_i p_i + f_i, \tag{17}$$

where γ_i is a friction coefficient representing viscous damping, and f_i is a random force accounting for thermal collisions with other particles.[24] The friction term slows down particle motion (*viscosity*), while the random force models thermal agitation (*collisions*). Together, they regulate the system's temperature and ensure sampling from the canonical NVT ensemble.

The random force f_i is drawn from a Gaussian distribution with zero mean and a variance given by:

$$\sigma_i^2 = \frac{2m_i\gamma_i k_B T}{\Delta t}, \tag{18}$$

where m_i is the particle mass, k_B is Boltzmann's constant, T is the desired temperature, and Δt is the integration timestep.[24] These forces are applied at each timestep or at regular intervals defined by a collision frequency.

Although the Langevin formulation alters the momentum evolution relative to Eq 3, it remains consistent with the velocity definition in Eq 4 via $v_i = p_i/m_i$.

As seen in Section 2.1, the evolution of atomic positions and momenta is typically described by Newton's second law (see Eq. 1-4). These equations describe an isolated system where total energy is conserved, *NVE*.

Nosé-Hoover

The Nosé–Hoover thermostat is one such method, widely used due to its ability to generate a canonical distribution without disrupting the deterministic and time-reversible nature of Newtonian dynamics.[25,26]

The Nosé–Hoover method modifies the classical equations of motion by introducing an additional variable, $\zeta(t)$, which acts as a feedback control term representing the coupling to a thermal reservoir. The force equation is altered by adding a friction-like term that scales with the momentum, resulting in the modified momentum equation:

$$\dot{P}_i = F_i - \mu\zeta(t)P_i(t) \tag{19}$$

Here, μ is a coupling constant, and the term $-\mu\zeta(t)P_i$ dynamically adjusts the system's kinetic energy. The evolution of $\zeta(t)$ itself is governed by:

$$\dot{\zeta}(t) = \mu\frac{T(t) - T_0}{T_0} \tag{20}$$

where $T(t)$ is the instantaneous temperature of the system, and T_0 is the desired target temperature. This integral feedback mechanism ensures that when the system's temperature deviates from T_0, the friction term adjusts accordingly to return the system toward equilibrium.

The main advantage of the Nosé–Hoover thermostat is that it enables correct sampling from the canonical ensemble while preserving the time-reversibility of the equations of motion. However, if the thermostat parameters are poorly chosen, it can lead to oscillatory behavior or even persistent deviations from the canonical distribution.[27]

2.2 Langevin dynamics

Langevin dynamics offers a way to simulate particle motion while accounting for thermal fluctuations, without explicitly including solvent molecules.[28] Instead, it represents the solvent's influence through additional frictional and random forces, allowing the system to behave similarly to how colloidal particles move in a fluid, consistent with Brownian motion.[24]

In this framework, the total force acting on a particle is composed of three main components: conservative force F_c , a frictional drag force - viscous damping proportional to particles' velocity F_f, and a random force simulating solvent particles bumping randomly into the colloid particles F_r. These forces combine to form the Langevin force:[24]

$$F_{\text{Langevin}} = F_c + F_f + F_r \tag{21}$$

The conservative force F_c coming from the pair-style/inter-particle interactions. The frictional force opposes the motion of the particle and is linearly proportional to its velocity:[24]

$$F_f = -\frac{m}{\gamma}v \tag{22}$$

In this equation, m is the mass of the particle, v is its velocity, and γ is the damping constant. This parameter controls how strongly the particle's velocity is suppressed to drive the system toward

thermal equilibrium.[24,29]

The damping constant γ can also be expressed in terms of a damping time τ:[24,29]

$$\tau = \frac{m}{\gamma} \tag{23}$$

For spherical particles immersed in a viscous medium, the damping constant can be estimated using Stokes' law:[24,29]

$$\gamma = 3\pi\eta d \tag{24}$$

where η is the viscosity of the surrounding fluid and d is the diameter of the particle. Substituting this into the expression for τ, it is obtain:[24,29]

$$\tau = \frac{m}{3\pi\eta d} \tag{25}$$

This relation helps connect microscopic properties of the particle and solvent to the thermostat parameters in simulations. If one knows the desired viscosity and particle size, one can compute τ and assign it to each particle type accordingly—particularly when configuring 'fix langevin' in LAMMPS or similar tools. A shorter τ corresponds to stronger damping (high viscosity), and a longer τ suggests a low-viscosity environment.[24,29]

The stochastic force F_r accounts for the random impacts from solvent molecules. While each collision is unpredictable, their collective influence is statistically described by the fluctuation-dissipation theorem, which ensures that the energy introduced by thermal kicks is balanced by the energy lost through friction. The variance of this random force is expressed as:[24]

$$F_r \propto \sqrt{\frac{k_B T m}{\gamma\, dt}} \tag{26}$$

where k_B is the Boltzmann constant, T is the temperature, and dt is the simulation time step.

The choice of γ significantly affects system behavior. A small damping constant corresponds to rapid temperature relaxation and mimics a low-viscosity solvent, while a larger γ leads to slower thermal adjustment. The particle mass m should also be scaled according to particle size, as it impacts both the drag force and the intensity of the random fluctuations.[24]

Langevin thermostats are particularly useful for maintaining temperature in molecular dynamics simulations while preserving realistic thermal motion without modeling solvent particles explicitly.

Hydrodynamic interactions

Particles moving through a viscous fluid generate a flow field that alters the motion of surrounding particles even if they are separated by a considerable distance. The disturbance in the fluid does not fade quickly but instead decreases slowly over space. These solvent-mediated, long-range effects occur only when particles are in motion and are known as hydrodynamic interactions.[30]

The system is modeled via Langevin dynamics with an implicit solvent, a standard approach that neglects explicit hydrodynamic interactions (HIs) while still capturing Brownian motion in colloids.[31] In this approximation, solvent-mediated coupling between particles is ignored.[31] At the present volume

fraction ($\approx 25\%$), hydrodynamic coupling can influence the dynamics; however, many studies of colloidal diffusion at similar concentrations omit HIs for simplicity, especially when focusing on short-time or intermediate-time behavior.[32] Incorporating full many-body HIs via methods such as Stokesian dynamics or lattice Boltzmann is computationally intensive and often unnecessary for moderately sized systems.[33]

2.3 Dynamic of colloids

Colloidal systems involve large molecules or aggregates termed colloidal or Brownian particles, exhibiting random motion due to collisions with solvent molecules, a phenomenon known as Brownian motion. Brownian motion reflects thermal agitation observable through microscopy and highlights the solvent's molecular structure indirectly. Such systems are defined by size ranges: colloidal particles must be substantially larger than solvent molecules but small enough to remain thermally agitated.[20]

The minimum colloidal particle size is determined by ensuring multiple solvent molecules simultaneously interact with its surface, approximately 1 nanometer or larger, typically about ten times the solvent molecule size. The maximum particle size, typically around 10 micrometers, is limited by gravitational influences overshadowing thermal motion. Particles surpassing this upper limit settle due to gravity, preventing observable Brownian motion.[20]

These colloidal dispersions vary from rigid particle systems to flexible macromolecules or aggregates of smaller molecules. Macromolecular colloids, like large proteins or polymer chains, embody thermodynamic equilibrium with their surroundings, revealing insights into molecular behavior and solution dynamics at microscopic levels. Understanding colloids bridges microscopic molecular interactions to macroscopic properties like viscosity and temperature dependence.[20]

Colloidal suspensions are common in both natural and industrial systems. In biological environments, the interior of a cell contains proteins, organelles, and other macromolecules that, while structurally diverse, can be modeled as colloidal particles to capture essential aspects of their behavior in a simplified form. As mentioned previously in Section 1, these macromolecules contribute to the crowded nature of the cytoplasm. According to both previous and current studies, modeling this medium as a dense colloidal suspension provides a useful starting point for understanding key processes linked to cellular function.[8,30] This approach offers a less complex representation of the cytoplasm that can be extended progressively toward a more detailed understanding of its supramolecular organization.

To reduce computational costs, this study employs a coarse-grained modeling approach, representing complex biological macromolecules as colloidal particles. Fully representing macromolecules at an atomic level, including structural details such as side chains and internal flexibility, drastically increases computational complexity and cost, particularly for simulations aimed at reaching millisecond durations.[12] By modeling macromolecules as colloidal particles with effective size and interaction parameters, the simulations become computationally feasible while retaining the essential physical characteristics. This coarse-grained approach thus provides an efficient method for exploring the collective dynamics of macromolecules in crowded environments such as the cytoplasm.

2.3.1 Solvent

Accurately capturing solvent effects is important for developing simulations that reliably reproduce both microscopic and macroscopic behavior. Modeling the solvent explicitly requires much more computational effort than an implicit one because including individual solvent molecules prolongs the simulation runtime. For example, a typical simulation setup in this work is a 5000^3 Å^3 cell with 1000 colloidal particles and 25% packing. Explicitly adding water molecules to the remaining volume would require simulating additional $3.14 \cdot 10^9$ particles (see Appendix B for an example of this estimation). This issue is especially acute for systems that already demand lengthy simulation periods such as this one, where simulations are meant to run a total time of microseconds to milliseconds.[34,35] An alternative to explicit solvent representation is to employ an implicit approach, in which the solvent is treated as a continuous medium instead of discrete particles.[36,37] In an implicit framework, the solvent's dynamic influence must still be captured through effective terms that reproduce its essential behavior.[36,37] Because many phenomena of interest occur over extended time and length scales, the implicit model needs to replicate not only the primary dynamical features but also preserve microscopic coupling with colloidal particles. For example, collisions between particles should still influence the surrounding fluid flow.[35,38]Maintaining a simplified description allows simulations to reach time-frames on the order of microseconds to milliseconds.[35]One way to achieve this balance between accuracy and efficiency is to use Langevin dynamics, as it was explained in section 2.2 , which incorporates stochastic and frictional forces to model solvent effects without tracking individual molecules.[38]

2.3.2 Diffusion

Brownian motion describes the random, thermally induced motion of particles suspended in a fluid. This irregular motion results from frequent and uneven collisions with surrounding solvent molecules. The theoretical explanation linking thermal fluctuations to observable particle displacements was first developed by Einstein in 1905[39] and later confirmed experimentally by Jean Perrin.

Diffusion is fundamentally associated with the displacement of molecules resulting from their thermal motion.[40] Diffusion processes are generally classified into three distinct types:

- **Inter-diffusion**: Occurs when two different molecular species mix via diffusion, driven by opposing concentration gradients.
- **Collective (Gradient) Diffusion**: Describes the joint motion of Brownian particles under density gradients.
- **Self-diffusion**: Describes the dynamics of a single particle within a system of homogeneous density.

Self-diffusion, particularly relevant to this study, describes the dynamics of a single particle within a system of homogeneous density. Typically, the single particle under investigation is referred to as the *tracer particle* or *tagged particle*, whereas the remaining particles constitute the *host medium*.[20]

The fundamental quantity characterizing the motion of a Brownian particle is the mean squared displacement (MSD), defined as:[20]

$$\mathrm{MSD}(t) = \left\langle |\mathbf{r}(t) - \mathbf{r}(0)|^2 \right\rangle , \tag{27}$$

where The *mean-squared displacement* $\mathrm{MSD}(t)$ measures how far, on average, a particle has wandered after a time t. It is defined as the ensemble average of the squared distance between its position at time t, $\mathbf{r}(t)$, and its initial position, $\mathbf{r}(0)$.

At very short timescales, a particle's motion is predominantly ballistic; it moves nearly linearly due to inertia and has yet to undergo substantial collisions. In this ballistic regime, the MSD grows quadratically:

$$\mathrm{MSD}(t) = v_0^2 t^2, \tag{28}$$

where is the particle's initial velocity and t the time.

At longer timescales, the particle experiences numerous collisions with solvent molecules, resulting in randomized and diffusive motion. In this diffusive regime, the MSD increases linearly with time:

$$\mathrm{MSD}(t) = 6 D_s t, \tag{29}$$

where D_s is the self-diffusion coefficient. The factor of 6 applies to diffusion in three-dimensional space.

In a dilute suspension (one particle in pure solvent), the self-diffusion coefficient equals the free-particle diffusion coefficient (D_0) , described by the classical Stokes–Einstein equation:

$$D_0 = \frac{k_B T}{6\pi\eta r}, \tag{30}$$

where k_B is Boltzmann's constant, T is absolute temperature, η is the dynamic viscosity of the fluid, and r is the particle radius. This relationship indicates that smaller particles, lower viscosity, or higher temperatures increase diffusion rates.

However, in concentrated suspensions containing multiple Brownian particles, particle interactions significantly influence diffusion, reducing the self-diffusion coefficient from its infinite-dilution value . These interactions may be hydrodynamic, arising from fluid-mediated forces, or steric, due to direct physical exclusion effects. Consequently, the self-diffusion coefficient depends on particle concentration, typically expressed by the volume fraction , resulting in distinct short-time and long-time diffusion coefficients.[20]

For hard-sphere suspensions, the short-time self-diffusion coefficient characterizes diffusion at short observation times when the suspension structure remains nearly unchanged:[20]

$$D_s^S = D_0 \left(1 + \alpha_1^s \varphi\right), \quad \alpha_1^s \approx -1.83. \tag{31}$$

Conversely, the long-time self-diffusion coefficient describes diffusion after significant structural rearrangements, capturing sustained particle interactions and cage effects:[20]

$$D_s^L = D_0 \left(1 + \alpha_1^L \varphi\right), \quad \alpha_1^L \approx -2.10. \tag{32}$$

These equations illustrate that increased particle concentration systematically reduces diffusion rates. The linear relationships provided above are accurate for low volume fractions, typically up to $\phi = 0.05$. Beyond this concentration, deviations from linear behavior arise due to higher-order

interactions and complex structural changes.[20]

In simulations where particle size is the only varying parameter, the diffusion coefficient shows an inverse relationship with particle size. This relationship allows for model validation by comparing the ratios of particle radii to their corresponding diffusion coefficients:[20]

$$D_s \propto \frac{1}{r} \tag{33}$$

2.3.3 Step-size distribution

In Brownian motion, each particle experiences many small, random bumps from the surrounding particles.[39] These bumps are independent and random over short timescales. Over time, these small steps sum into particle displacements:

$$\Delta x(t) = x(t + \Delta t) - x(t) \tag{34}$$

where Δx is a particle's displacement in the x direction, i.e. the step-size in the x direction. This net displacement is the result of many independent, random moves made in a period of time Δt. According to the Central Limit Theorem (CLT), the sum (or the average) of many independent, random variables tends toward a Gaussian distribution, as the number of variables increases.[41] This means, that given a sufficiently large Δt (in order for many random bumps to happen), the resulting step-sizes across many particles or across long time will tend to obey the Gaussian distribution. In other words, the probability distribution of of Δx will be proportional to a Gaussian-type function:

$$p(\Delta x) \propto e^{-(\Delta x)^2} \tag{35}$$

and that:

$$ln(p(\Delta x)) \propto -(\Delta x)^2 \tag{36}$$

In the case that the particle movements are non-Brownian, the probability distribution of particles' Δx values will diverge from the bell-shaped distribution and will show exponential tails. In turn, this would be reflected in the logarithm of $p(\Delta x)$ deviating from the $-(\Delta x)^2$ shape. This means that the Brownian motion can be confirmed by examining the step-size distributions.

3 Computational details

This section provides a detailed description of the computational methods, parameters, and software used throughout the study. The choices made in terms of theoretical approaches and simulation settings are justified to ensure the reliability and reproducibility of the results. Every explanation in the following input files refers to changes that were made and studied before adding or removing new lines from the default input (Appendix A), in order to build a model aligned with the desired objective of this project. Several modifications were applied, and all are thoroughly explained.

3.1 LAMMPS

LAMMPS stands for Large-scale Atomic/Molecular Massively Parallel Simulator. It is a software package used to perform molecular dynamics simulations and was originally developed by researchers at Sandia National Laboratories. The code is written in C++ and is designed to take advantage of parallel computing, which allows simulations to run efficiently on high-performance systems. This makes LAMMPS suitable for studying large systems with many particles or for running simulations over long time scales. Molecular dynamics simulations, in general, are used to study the behavior of systems at the particle level, and LAMMPS is one of the most widely used programs for this purpose, but it can also be implemented for energy minimization or Monte Carlo simulations. One reason for its popularity is that it is open source and actively maintained, with a large user community that contributes new features. Because of its flexibility, LAMMPS can be applied to a wide range of physical systems, from simple coarse-grained models to detailed atomistic simulations in materials science, soft matter, and biophysics.[42,43]

3.1.1 Initialization

In LAMMPS, the initialization phase defines the simulation environment, including unit conventions, particle description, spatial dimensions, and boundary conditions. Each of these settings must be carefully chosen to match the physical model being simulated. This is an example of the beginning of an input script, specifically for colloid. In the next paragraphs, each of the lines is going to be explained.

Initialization Script (LAMMPS)

```
units metal # mass in gmol^-1, distance in Å, ...
atom_style sphere # particles are defined by diameter and density
dimension 3 # dimension of simulation
boundary p p p # periodic boundary conditions in all 3 directions
```

The *units* command in LAMMPS is used to define the physical unit system that applies throughout a simulation. It affects how all numerical values are interpreted in the input script and data files, including quantities such as distance, energy, mass, and time. It also controls the units of the values written in output files and on-screen results. This command is usually placed at the very start of the input script, since all other commands depend on it.[44] Available unit styles include *real*, *metal*, *si*, *cgs*, *electron*, and *lj*. The *metal* unit style is suitable for the present work because its base units align with the physical scales relevant to the system, as summarized in Table 1. It was also chosen for practical

reasons, as it had been used in previous projects within the group and was familiar to the supervisor, which helped ensure a more efficient setup and interpretation of simulation parameters.

Table 1: Base units for the *metal* unit style in LAMMPS which are the units used for the present project.[44]

Quantity	**Unit (metal)**
Distance	Angstrom (Å)
Time	Picosecond (ps)
Mass	Gram per mole (g/mol)
Energy	Electronvolt (eV)
Temperature	Kelvin (K)
Pressure	Bars
Force	eV/Å
Velocity	Å/ps
Charge	Electron charge (e)

The $atom-style$ command in LAMMPS determines what kind of information is stored for each particle in the simulation. This includes attributes like position, velocity, mass, and others, depending on the chosen style. Different interaction models in LAMMPS require specific particle properties, so the atom style must be compatible with the type of simulation being performed.[45] The atom style used in this project is sphere. This style includes basic properties like diameter, mass (via density), and optional angular velocity. These are enough to represent the particles in the model, which simulate macromolecules in a simplified, spherical form. More advanced styles exist, but are not needed for this colloid system. The dimension of the working system is three-dimensional and it is confined within a cubic simulation box with dimensions of 500 nm $\times$ 500 nm $\times$ 500 nm.

In LAMMPS, the boundary command defines how particles behave at the edges of the simulation box. For each dimension, different styles can be assigned depending on the physical scenario being modeled. The f style applies fixed non-periodic boundaries, meaning particles do not interact across the edge, and any that move outside the box may be removed from the simulation. The s style uses shrink-wrapping, where the boundary automatically adjusts to fit the current position of the atoms in that direction. This can be useful in systems with dynamic size changes but may lead to instability in parallel runs if the box size changes significantly. The m style also uses shrink-wrapping but maintains a minimum box size set in the input file, which can prevent atoms from being lost when the system shrinks too far.[46]

In this project, the $p\ p\ p$ option is used to apply periodic boundary conditions in all three directions. This means particles that leave the box on one side re-enter from the opposite side with the same properties, as can be seen in Fig 6.[46] This set-up creates a seamless, repeating environment that mimics a small section of the cytoplasm without introducing artificial borders or edge effects.

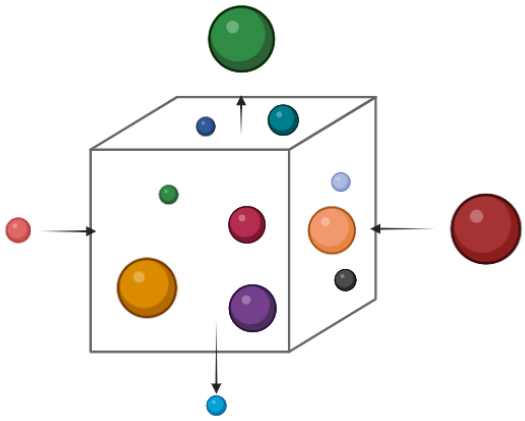


Figure 6: Schematic representation of periodic boundary conditions. Particles that leave one side of the simulation box re-enter from the opposite side, maintaining continuity across the system. This meant to represent that small space within the cytoplasmic environment that is being studied.

3.1.2 System definition

To compute short-range interactions efficiently, LAMMPS uses a neighbor list. This list stores atom pairs that are close enough to interact, which avoids checking all possible pairs and greatly reduces the computational cost, especially in large systems. Each processor builds its own list, including both atoms it owns and nearby atoms from neighboring regions, known as ghost atoms.[47]

To speed up the neighbor search, LAMMPS divides the simulation space into small boxes. Each atom is placed into a box based on its position, and it only checks for neighbors in nearby boxes. This limits the number of comparisons and keeps the process fast. The decision of whether an atom pair goes into the neighbor list is based on distance. Fig 7 illustrates this idea: atoms within the cutoff radius are close enough to interact directly, while the outer skin radius adds a margin of safety.[47]

The cutoff radius defines the maximum interaction range between two atoms. However, because atoms move during the simulation, the neighbor list would quickly become outdated without some buffer. To solve this, LAMMPS adds a skin distance, which allows the same neighbor list to remain valid for multiple steps. The list is only rebuilt when an atom moves more than half the skin distance, which helps reduce unnecessary recalculations while still capturing all relevant interactions.[47]

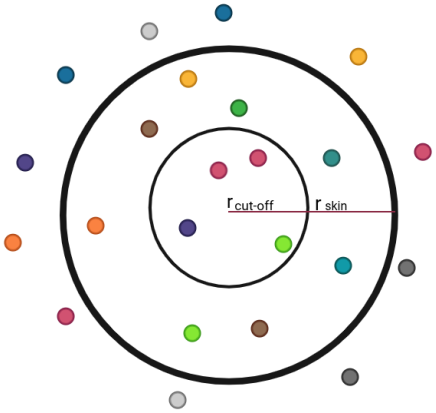


Figure 7: The purpose of this image is to simplify the concept of neighbor lists itself. You can observe the cutoff radius (r cut-off) and the skin radius (r skin).

The commands used in the input script configure how these neighbor lists are constructed and maintained. The line *neighbor* 50.0 *multi* sets the skin distance to 50 Å. The *multi* keyword is necessary when interactions involve many different cutoff distances.[47]

The line neigh_modify delay 250 tells LAMMPS to rebuild the neighbor list every 250 steps. This setting increases the frequency of neighbor list updates, which can improve accuracy if atoms move rapidly.[47]

Lastly, comm_modify mode multi adjusts how atoms are communicated between processors during neighbor list construction. Although not strictly necessary without hybrid potentials, enabling this mode can improve performance or compatibility when multiple neighbor lists are being used internally.[47]

These settings are important for balancing performance and accuracy in the simulation by controlling how and when neighbor lists are updated and managed.

System definition

```
read_data lammps.pos # file specifying simulation cell size and initial particle
positions

neighbor 50.0 multi
neigh_modify delay 250 every 1 check yes
comm_modify mode multi
```

read_data lammps.pos can be checked in section 3.2.

3.1.3 Simulation settings

This input section should be given special attention (without diminishing the importance of the other sections), as it forms the core of the simulation. It is here that the interactions between the particles in the system are defined and understood. For a better understanding of the potential section 2.1.3 must be check and for $lammps.forcefield$ section 3.2

Simulation Settings

```
pair_style colloid
include lammps.forcefield # force field / pair_coeffs
```

3.1.4 Simulation time-step

The choice of timestep plays a critical role in determining the accuracy of molecular dynamics simulations. A smaller timestep generally leads to better energy conservation and more stable integration of particle trajectories. However, using very small timesteps significantly increases the total computational cost, as more steps are needed to simulate a given period of physical time. Conversely, larger timesteps reduce the simulation time but can introduce integration errors, particularly in systems with high-frequency motions, and may result in artificial energy drift. For this reason, it is essential to perform a timestep sensitivity analysis, where the system's total energy is monitored across simulations using different timestep values. This allows for the selection of a timestep that balances numerical stability with computational efficiency.[48]

Time Integration Settings

```
timestep    0.25    # picoseconds
```

3.1.5 Minimization

The systems simulated in this work are defined by randomly packed spheres of different radii in a box of a given size. The random packing is done by a custom Python script, which is explained in the section 3.2. As this random packing may result in unphysical forces or overlaps of particles, an energy minimization step is performed to relax the initial configuration by reducing the system's potential energy. This is done via *minimize* and *min_style* commands.

The *minimize* command iteratively adjusts particles' coordinates until one of the criteria is met. The criteria are respectively: stopping tolerance for energy, stopping tolerance for force, maximum iterations of the minimizer, and maximum number of force/energy evaluations. The *min_style* commands sets the minimization algorithm, which in this case is the Polak-Ribiere version conjugate gradient (cg) algorithm. This algorithm uses both current step and previous step force gradient information to update the search directions.

Energy Minimization

```
min_style    cg
minimize     1.0e-4 1.0e-6 1000 10000  # minimization thresholds
run            0
```

3.1.6 Heating

The simulated systems are initialized as randomly distributed particles in a box of given size that underwent an energy minimization at $0\,K$. Such an initial arrangement does not represent a thermally equilibrated state and particle distribution due to thermal motion.

Therefore, gradually heating the system to the target temperature allows the particles to explore configuration space, and ensures that the system evolves towards a physically meaningful and thermally relaxed state. This is done via an initial heating stage in all the performed simulations.

The *velocity* command in LAMMPS assigns velocities to particles. In this specific example, the *velocity all create 1.0* assigns random velocities that correspond to the temperature of $1.0\,K$ to all of the defined particles. A random number seed is provided that is used to generate the velocities. Net linear momentum is removed from the system after assigning the velocities with *mom yes* keywords. Velocity distribution is set to Maxwell-Boltzmann distribution via the *dist gaussian* keywords. The system is then evolved to the desired temperature of $303.15\,K$ using the $fixnvt$ command.

The *thermo_style* command defines which thermodynamic data will be output to the screen and log files and in which style, with *thermo* command specifying the output frequency in number of steps. Here, *thermo_style* command specifically asks for step number, temperature, potential, kinetic, and total energies to be printed every 100 steps. The *log heating.log* command instructs that everything output to the screen during this stage is saved into a heating.log file.

In Fig. 8 can be seen the increase of temperature until it reaches the wanted one for the biological-like system.

Initial Heating Stage

```
velocity all create 1.0 5982396 mom yes dist gaussian
fix heating all nvt temp 1.0 303.15 $(100.0*dt)
log heating.log
thermo_style custom step temp pe ke etotal
thermo 100
run 300000
unfix heating
```

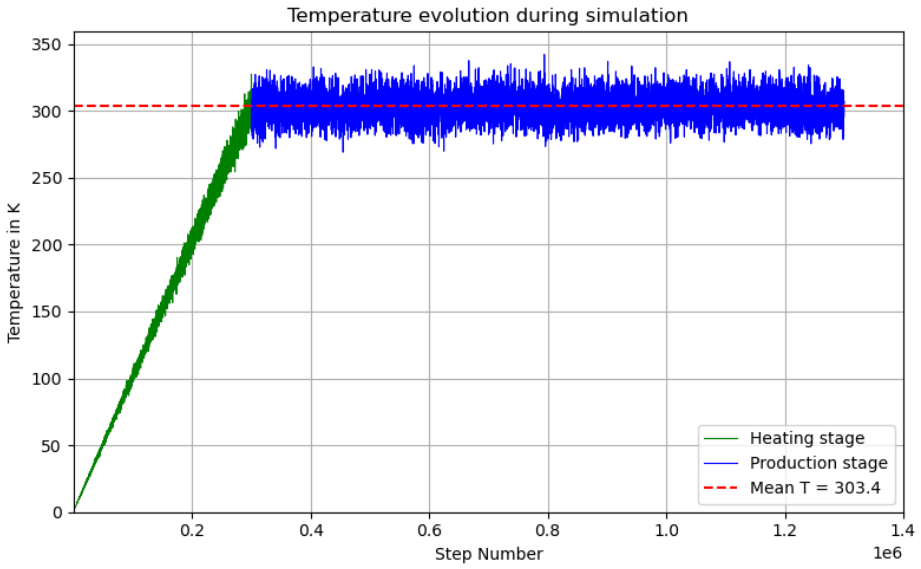


Figure 8: Temperature of the simulation versus step number. First phase shown in green color represents temperature evolution during the heating phase. The temperature oscillation during the simulation phase is shown in blue color and is cut at 1 million steps (not counting first 300 thousand heating steps) for visual purposes. Red dashed line represents the mean temperature during the cut simulation phase. This is obtained from a simulation with a heating phase set to heat from 1.0 to 303.15 K, and a simulation phase meant to maintain 303.15 K.

3.1.7 Simulation

After the system is brought to the target temperature using a Nose-Hoover thermostat during the heating stage, the simulation enters the production phase. This stage operates under the micro-canonical ensemble (NVE)(see sections 2.1.6-2.1.7), but it must be noted that the true NVE nature is altered by the use of the Langevin thermostat. The primary objective of this phase is to observe the time evolution and transport properties of the system under equilibrium conditions.

To introduce viscous damping representative of a realistic solvent environment, Langevin dynamics (see section 2.2) are included via the command *fix langevin*. This fix applies the thermostat to a group of atoms, along with the viscous and stochastic forces. The viscous and the stochastic forces depend on the damping time which is proportional to the mass and radius of a particle. This means that a separate *fix langevin* command which specifies the desired temperature and damping time needs to be defined for each particle type when running the polydisperse simulations. For that purpose, each particle type is first defined as a group and then the fix commands are defined to be applied to the respective groups (where a given group includes all particles of a given radius). For clarity of the input file, the fix commands are defined in an external file and included via the *include* command. LAMMPS natively allows computation of MSD of the particles via *compute msd* commands which are defined for any desired group . The NVE integration is then called for by the $fix\,nve$ command. The thermodynamic output is now customized to also output computed MSD values. A trajectory file can be output for any specific group of particles or all particles via the *dump* command. In the example input file presented, the trajectory file is specified to output the Cartesian coordinates and forces acting

on each individual particle in the given group, which is particularly useful for the visualization of the particle movements throughout simulation. Lastly, the production stage is run for the desired number of steps via the *run* command.

In figure 9 can be seen the final simulation, how it looks, and shows a clear polydisperse environment.

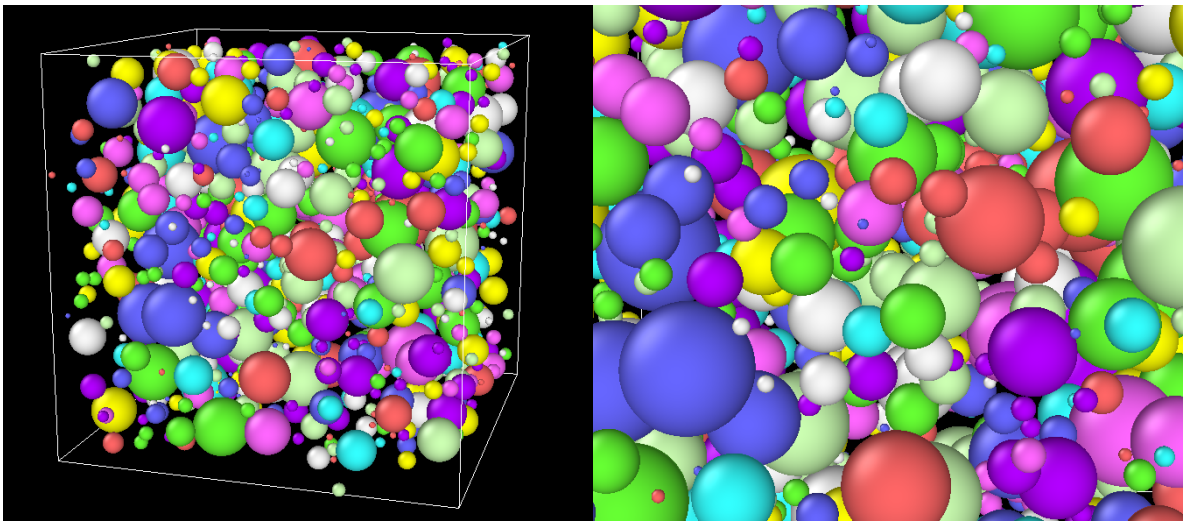


Figure 9: Visualization from OVITO showing macromolecules as spheres of different sizes and colors, reflecting their dimensional differences.

Simulation

```
-group definitions here-
include lammps.langevin
-compute definitions here-
fix simulation all nve
log simulation.log
variable nsteps equal 100000000
variable output_freq equal 10000
thermo_style custom step temp pe ke etotal c_2msd[4] ... c_40msd[4]
thermo $(output_freq)
dump example: dump 2nm custom $(output_freq) 2.lammpstrj id type diameter x y z
fx fy fz vx vy vz
run $(nsteps)
```

3.2 Python script

In order to run a simulation, one needs to define the system with all of its particles and their properties such as initial positions and particle densities, as well as force-field pair coefficients for all of the possible pairwise interactions, and particle type-specific Langevin thermostat parameters. A Python script is written to facilitate the workflow and efficiently obtain essential data for the simulation. The code was designed to automate key tasks and ensure a consistent and reproducible setup of the simulation environment. This section provides a brief overview of the most important pieces of code. Overall, it is supposed to, based on user inputs passed as command line arguments, execute the following tasks:

1. Generates the discrete lognormal radii distribution in a given range, and with a given mode radius, such that it satisfies the desired number of particles or packing fraction within the given volume.

2. Creates a random initial configuration of the particles within defined simulation cell dimensions, ensuring no initial overlap of the particles. Writes this configuration in a *lammps.pos* file, which is to be read by the program on execution.

3. Writes a $lammps.forcefield$ file which defines all $N(N+1)/2$ colloid pair style pair coefficients, where N is the number of particle types. Pair coefficients include cutoff distances. This code also calculates the cutoff distances dynamically based on a given energy threshold.

4. Writes a *lammps.langevin* file which defines the $fix\,langevin$ commands for each particle type. This includes dynamically calculating the damp time parameters based on particle's mass and input viscosity.

5. Writes the *in.lammps* file, which is the input file that LAMMPS code executes. The code can adjust certain sections of the input such as compute and dump coefficients, and other parameters that user can change to automate creation of multiple system specific files for running batch jobs.

The snippet of code given in Listing 1 is in charge of item 1. of the above list. It takes as inputs a packing fraction d, volume of the simulation cell V, minimum radius r_{min}, maximum radius r_{max}, mode radius (most probable radius) r_{mode}, radius step Δr, the width of the underlying Gaussian distribution σ, and optionally the number of particles N. The function first creates an array of radii in the wanted range:

$$radii = r_{min}, r_{min} + \Delta r, r_{min} + 2\Delta r, \ldots, r_{max} \tag{37}$$

A log-normal distribution is given by:

$$p(r) = \frac{1}{r\sigma\sqrt{2\pi}} exp\big[- \frac{(lnr - \mu)^2}{2\sigma^2}\big] \tag{38}$$

where $p(r)$ is the probability of the radius r occuring. Here, μ is given by:

$$\mu = ln(r_{mode}) + \sigma^2 \tag{39}$$

which is done by the function line 3. The probabilities of each radius $p(r_i)$ are then calculated using the lognorm.pdf function from the SciPy library (line 4). This creates an array of $p(r_i)$ values.

The probabilities are then normalized to obtain a discrete distribution (as the radii take on values in steps of Δr, rather than a continuous set of values):

$$p_i^{norm} = \frac{p_i}{\sum_j p_j}, \ \sum_i p_i^{norm} = 1 \tag{40}$$

which is done in line 5.

Volumes of spheres of each defined radius are then calculated with:

$$V_i = \frac{4}{3}\pi r^3 \tag{41}$$

and stored in an array called volumes. The user can either provide the volume of the simulation cell and packing fraction, or provide a desired number of particles. In the case of the former, the number of particles has to be calculated in such way to conform to the specified cell dimensions and packing fraction, and abide to the probability distribution. The expected relative volume to be occupied by the particles of radius r_i with probability p_i^{norm} is:

$$V_i^{prob} = p_i^{norm} V_i \tag{42}$$

This means that the number of particles N_{total} needed to fill the desired occupied volume $d \cdot V$ is given by:

$$N_{total} = \frac{d \cdot V}{\sum_i V_i^{prob}} = \frac{d \cdot V}{\sum_i p_i^{norm} V_i} \tag{43}$$

This is done in lines 10 and 11. The number of times a particle of radius r_i with probability p_i^{norm} will occur in the distribution is then given by:

$$N_i = floor(p_i^{norm} N_{total}) \tag{44}$$

As a given probability value is typically not a whole number, the product $p_i N$ will not give a whole number either. For this reason, the product is rounded to the nearest lower integer using the $floor()$ function. This rounding can lead to the number of particles generated being slightly lower than the desired one, which is handled by the lines of code 20 to 25. In practice, the script was more commonly used with providing a wanted number of particles directly (with the cell volume being adjusted dynamically for a given packing fraction). However, this functionality is useful for generating systems with constant cell dimensions while changing packing fraction.

Listing 1: Generation of lognormal distributed radii

```
1 def generate_discrete_lognormal_radii(d, V, r_min, r_max, r_mode, delta_r,
      sigma=0.8, N_particles=None):
2     radii = np.arange(r_min, r_max + delta_r, delta_r)
3     mu = np.log(r_mode) + sigma**2
4     probability_values = lognorm.pdf(radii, s=sigma, scale=np.exp(mu))
5     normalized_probability_values = probability_values / np.sum(
          probability_values)
6     volumes = (4/3) * np.pi * radii**3
7
```

```
    if N_particles is None:
        # Calculate N_total based on packing fraction and box volume
        result = np.sum(normalized_probability_values * volumes)
        N_total = int((d*V) / result)
    else:
        # Use provided N_particles
        N_total = N_particles

    expected_frequencies = normalized_probability_values * N_total
    floored_frequencies = np.floor(expected_frequencies).astype(int)
    total = np.sum(floored_frequencies)
    deficit = N_total - total
    if deficit > 0:
        remainder = expected_frequencies - floored_frequencies
        top_bins = np.argsort(remainder)[-deficit:]
        floored_frequencies[top_bins] += 1
    samples = np.repeat(radii, floored_frequencies)
    return radii, samples, normalized_probability_values, N_total
```

Figure 10 shows an example resulting distribution generated by the above function.

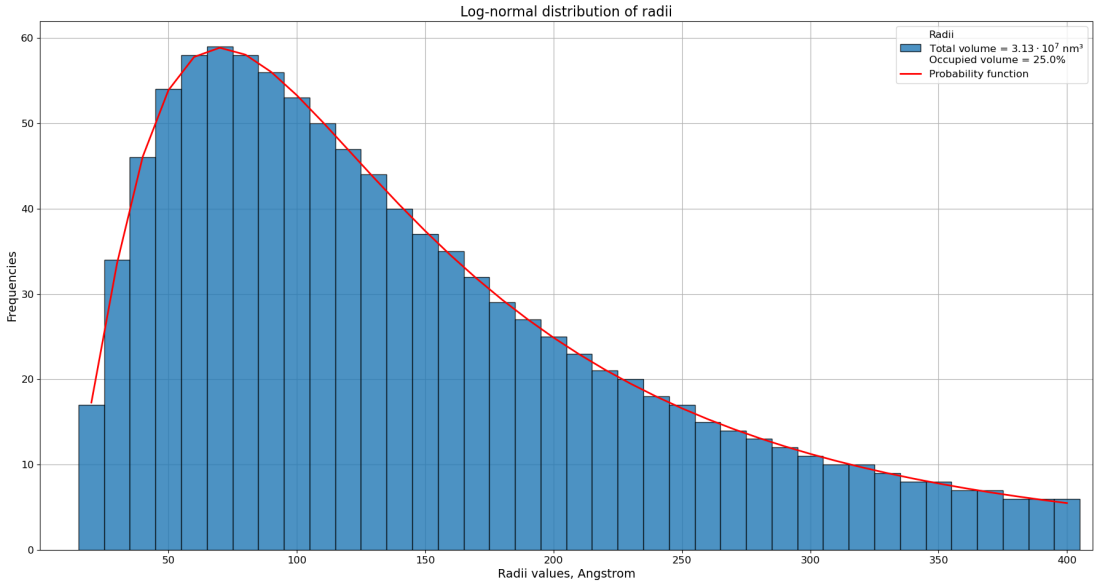


Figure 10: Histogram of generated particle radii following a discrete log-normal distribution. The bars represent the frequency of radii sampled within the range of 20 Å to 400 Å in steps of 10 Å, while the red curve corresponds to the scaled log-normal probability density function used for sampling. The total volume occupied by the particles is approximately 3.13×10^7 nm^3, representing 25.0% of the total simulation box volume (500^3 nm^3).

The next step is to generate initial Cartesian coordinates of each of the particles from the generated distribution. The function in charge of that is given in Listing 2. The function takes the list of all

individual particles (it is a list of radii values, where each radius is repeated the number of times that radius appears in the distribution) and it attempts to generate a random set of (x, y, z) coordinates for each of them, such that $0 \leq x, y, z \leq a$, where a is the length of the simulation cube's side (the box_size variable). This is done particle by particle, starting from the largest particle and going down to the smallest one. Each time the random (x, y, z) coordinates are generated, the code checks whether particle at this position would overlap with any of the previously generated particles, regenerating new random coordinates until a position is found where the newly added particle i is at least $r_i + r_j + 1\text{Å}$ apart from all other particles j. The 1 Å buffer is added to ensure no contacts, which could cause potential energy spikes at the start of the simulation. This is done for maximum 10000 attempts per particle. For reproducibility, same random seed is used for generating the random numbers.

Listing 2: Generation of initial particle positions

```
def place_particles(particles, box_size=5000, max_attempts=10000):
    np.random.seed(12345)

    positions = []
    placed_particles = []
    for particle in particles:
        valid = False
        attempts = 0
        while not valid and attempts < max_attempts:
            new_pos = np.random.uniform(r + 0.5, box_size - r - 0.5, 3)
            if is_valid_position(new_pos, r, positions, placed_particles):
                positions.append(new_pos)
                placed_particles.append(r)
                valid = True
            attempts += 1
        if not valid:
            raise RuntimeError(f"Failed to place particle with radius {r}
                after {max_attempts} attempts.")
    return np.array(positions)
```

The pair style colloid potential function defines the interaction potential of two particles i and j of radius a_i and a_j at a distance $r_{ij} < r_{cutoff}$, for a given values of A and σ (see equations 14 and 13). LAMMPS requires a set of pair coefficients to be defined, which give the values of A, σ, a_i, a_j, and r_{cutoff} for all possible interactions ij. In this work, the cutoff distances were calculated such that

$$F(r_{cutoff}) = 0\,eV \tag{45}$$

where F is the resulting force from the interaction of particles i and j. The analytical expression for the force expression is rather complex (due to complexity of potential function U itself), and it can be seen that the distance r where $U \approx 0\,eV$ is the same distance at which $F \approx 0\,eV/\text{Å}$. Because of this, it was chosen to calculate the cutoff distances such that the potential evaluated at that distance is arbitrarily close to 0 eV. This is done by the snippet of code shown in Listing 3. Given an energy threshold ε, the code attempts to solve the following equation for r:

$$U(A, \sigma, a_1, a_2, r_{cutoff}) - \varepsilon = 0 \tag{46}$$

Throughout all simulations, a value of $\varepsilon = 0.00001\,eV$ was used. This means that for the particles at distances greater than r_{cutoff}, interaction potential $U < 0.00001\,eV$ and $F \lessapprox 0.00001\,eV/\text{Å}$ and are considered as not interacting.

Listing 3: Calculation of cutoff distances

```
def U_total(r, A, sigma, a1, a2):
    UA = -(A / 6) * (
        (2 * a1 * a2) / (r**2 - (a1 + a2)**2) +
        (2 * a1 * a2) / (r**2 - (a1 - a2)**2) +
        np.log((r**2 - (a1 + a2)**2) / (r**2 - (a1 - a2)**2))
    )
    prefac = A * sigma**6 / (37800 * r)
    term1 = (r**2 - 7*r*(a1 + a2) + 6*(a1**2 + 7*a1*a2 + a2**2)) / (r - a1 -
        a2)**7
    term2 = (r**2 + 7*r*(a1 + a2) + 6*(a1**2 + 7*a1*a2 + a2**2)) / (r + a1 +
        a2)**7
    term3 = (r**2 + 7*r*(a1 - a2) + 6*(a1**2 - 7*a1*a2 + a2**2)) / (r + a1 -
        a2)**7
    term4 = (r**2 - 7*r*(a1 - a2) + 6*(a1**2 - 7*a1*a2 + a2**2)) / (r - a1 +
        a2)**7
    UR = prefac * (term1 + term2 + term3 + term4)
    return UA + UR

def find_rcut(A, sigma, a1, a2, threshold=0.00001, xtol=1e-5, rtol=1e-8,
    maxiter=200):
    D = a1 + a2
    r_start = D + 1e-3
    r_end = D + 1000.0
    f = lambda r: abs(U_total(r, A, sigma, a1, a2)) - threshold
    try:
        r_cut = brentq(f, r_start, r_end, xtol=xtol, rtol=rtol, maxiter=
            maxiter)
        return r_cut
    except ValueError:
        return D + 500.0
```

The use of the Langevin thermostat requires defining a *fix langevin* command. This command requires the following parameters: T_{start}, T_{end}, τ and a random seed. T_{start} and T_{end} are the desired temperature at the start and the end of the simulation, τ is the damping time, and the random number seed is for generating the stochastic forces (see equations 25 and 26). This fix is applied to a specified group of atoms. As τ depends on the mass and radius of a particle, this fix needs to be specified for each group of particles with the same radius. The code shown in Listing 4 will dynamically calculate the required values for each radius present in the simulation. It uses the viscosity of water at 303 K by default, but can optionally adjust the damp times to correspond to viscosities that are equal to

multiples of the water viscosity. The resulting list of fix commands is written in a *lammps.langevin* file, read by the input script and executed by the program.

Listing 4: Calculation of damp times for the Langevin thermostat.

```
def write_langevin_file(type_mapping, viscosity_multiplier, seed, filename="
    lammps.langevin"):
    base_viscosity = 0.0007978  # Pa s
    viscosity_Pas = base_viscosity * viscosity_multiplier
    density = 500  # kg/m^3
    with open(filename, "w") as f:
        for radius, type_id in type_mapping.items():
            r_m = radius * 1e-10  # Angstrom to m
            d_m = 2 * r_m
            volume = (4/3) * np.pi * r_m**3
            mass = density * volume  # in kg
            gamma = 3 * np.pi * viscosity_Pas * d_m  # in kg/s
            tau_s = mass / gamma  # seconds
            tau_ps = tau_s * 1e12  # convert to ps
            f.write(f"fix langevin_{type_id} {0.5 * d_m * 1e9:.0f}nm langevin
                303.15 303.15 {tau_ps:.4f} {seed}\n")
```

The code can take the following parameters as command line arguments (all have default values set to match the typical system settings used in this work): neighbor list skin distance and delay value, timestep, number of simulation steps, packing density, simulation cell size, radii range values, width of the underlying Gaussian distribution, colloid potential parameters A and σ, energy threshold for calculating cutoffs, and viscosity. This is very convenient for automated generation of input files for many different systems at once.

4 Results and discussion

In this section, the main findings of the study are presented and analyzed. The results are discussed in relation to what the research wants to achieve and relevant literature, highlighting both expected outcomes and any unexpected observations. This discussion seeks to provide a better understanding of the implications of the findings within the context of the study.

4.1 Simulation Setup and Model Development Challenges

Before the actual implementation of the model and generation of meaningful simulation results, a significant portion of the work was dedicated to developing a functional and adaptable simulation environment. The initial focus was not on producing results directly, but rather on constructing a reliable system capable of capturing the complexity of the intended biological scenario. This involved understanding and modifying the baseline simulation framework provided by LAMMPS to reflect the specific needs of the project. Therefore, the outcomes discussed in this section correspond to the iterative process of model development, highlighting the practical and conceptual challenges encountered in building a suitable simulation foundation.

Once it was established that the system would be modeled using colloidal particles, the first phase of the project was to modify the default input given by LAMMPS (see Appendix A) to check changes). The objective during this stage was to understand the basic structure and functionality of the simulation script and to begin tailoring it to meet the specific requirements of the project. Some of the early stages of the work involved adapting particle interactions, system dimensions, and physical constants to better align with the biological context.

All the substantial modifications that were made to the original default input over the course of the project culminated in the current working version. These modifications are detailed in Section 3. Changes were implemented methodically and with careful consideration of their implications, as even small parameter variations could significantly impact system behavior. Selecting the appropriate interaction potential was particularly challenging because there are many different potentials for colloidal systems, and each of them has its peculiarities and difficulties. While there is a substantial body of theoretical and experimental work on the cytoplasmic environment, to the best of my knowledge, there is still a lack of clear quantitative data for some of the parameters needed in this work. One example is the use of interaction potentials, which are often discussed in general terms but not always specified in a way that supports direct implementation in simulations like the one developed here. Choosing a potential (see section 2.1.3 to check the theory and the chosen colloid-colloid potential) that could plausibly represent such a heterogeneous environment required a balance between physical realism and computational feasibility. This selection process involved reviewing literature, comparing multiple potential forms, and conducting exploratory simulations to evaluate their behavior in simplified test cases.

Early simulations often failed due to significant particle overlaps, which indicated fundamental issues in the input configuration. These overlaps stemmed primarily from three sources: (1) poor initial packing of colloidal particles, especially challenging due to the large differences in particle sizes in the polydisperse system; (2) an initial misunderstanding of key interaction parameters, particularly the Hamaker constant (A) and the interaction scale (σ), both of which directly influence the strength and range of interparticle forces; and (3) inconsistent or arbitrary choices of cut-off distances in the

potential, which prevented systematic control of the interaction range.

The point 3 here primarily came from the fact that many different radii are considered in these simulations, and that the radii appear in the colloid potential function (see Eq 13-14). Addressing these issues required iterative refinement of the system, including parameter sweeps, test simulations with varying densities and box sizes, and in-depth consultation of documentation and related literature.

In order to better understand the role of the Hamaker constant in the system, the interaction potential was plotted while all other parameters were held fixed. The analysis focused on how varying this constant influenced the nature of particle interactions. It was observed that increasing its value introduced stronger attractive forces, whereas reducing it diminished these attractions, which aligns with the theory found about the Hamaker constant.[18] Based on this understanding, the constant was initially set to a sufficiently low value to ensure that repulsive interactions were dominant. This choice aimed to approximate hard-sphere-like behavior during the early stages of model development. Since it was explained before, the hard-sphere-like behavior is particularly good for early stages analysis because it is the simplest form of the system.

It is important to clarify that the generality about the theory behind the Hamaker constant, and its relationship with Van der Waals interactions was understood; the issue was related to what should be the right value to give to the constant for the system. To my understanding, this is a non-trivial question and can require much more time; therefore, based on the time and needs of the project, the conclusion was to start with a small value of Hamaker. A very similar analysis was done to σ (see Eq 14 -13 to check the constants in the corresponding potentials).

4.2 Tuning of Neighbor List Parameters

To improve the efficiency of the simulations, I performed a parameter scan of the neighbor list settings. In LAMMPS, neighbor lists are used to avoid recalculating pairwise interactions at every timestep. The two key parameters that influence their behavior are the skin distance and the rebuild delay value. The skin determines the buffer region around each particle, and the delay sets how many time-steps pass before the list is checked or rebuilt. The results of this optimization are shown in Figure 11.

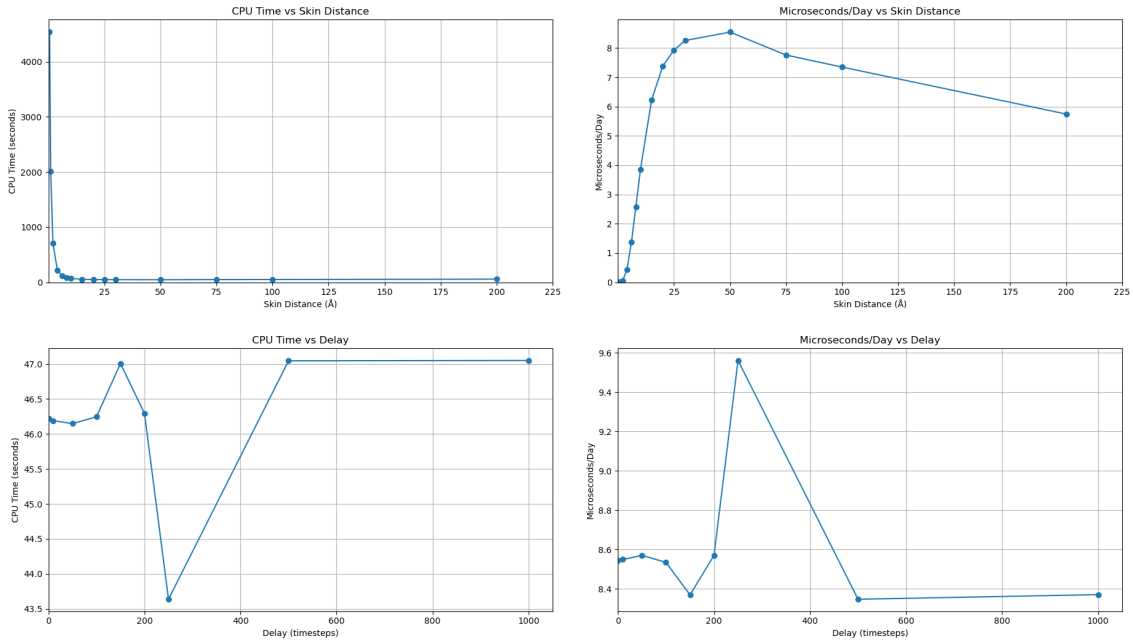


Figure 11: Optimization of the neighbor list parameters. Upper two graphs show CPU time and microseconds/day versus skin distance, with the delay value being 0 (list rebuilt every step). The lower two graphs show CPU time and microseconds/day versus delay value at skin distance of 50 Å. Here, microseconds/day is a performance measure that indicates how much of simulation time in microseconds would be achieved if the simulation ran for a day of real time. This optimization is done with 1 M steps simulations of 0.25 packing lognormal distribution polydisperse system with 1000 particles.

The upper two plots in Figure 11 show the total CPU time and throughput (in microseconds per day of real time) for different skin distances, with the rebuild delay fixed at zero. In this setting, the neighbor list is rebuilt at every timestep. When the skin is small (e.g., 5 Å), the simulation is extremely slow. This is because the neighbor list needs to be rebuilt very frequently due to particles quickly moving out of range. As the skin increases to 30–50 Å, the performance improves significantly. This reflects a balance where the neighbor list is large enough to remain valid for several steps, but not so large that it includes unnecessary pair checks. Above 50 Å, the throughput begins to decline slightly. This is expected, since a very large skin includes more neighbors than needed, increasing the cost of force calculations.

The lower two plots show how performance is affected by the rebuild delay, with the skin distance fixed at 50 Å. As the delay increases from 0 to 250 time-steps, the simulation becomes more efficient. The throughput peaks at this point, reaching just over 9 μs/day. This is because the overhead of rebuilding the neighbor list is reduced by checking it less frequently. However, beyond 250 time-steps, the performance becomes inconsistent and eventually declines. This is likely due to the neighbor list becoming outdated, which can lead to increased force errors and the need for emergency rebuilds. A very high delay value (e.g., 1 000 time-steps) results in poor performance and introduces instability into the integration.

Based on these results, the optimal settings were determined to be a skin distance of 50 Å and a delay of 250 time-steps. This combination provides a good balance between accuracy and efficiency.

These values were used for all subsequent simulations in this study to ensure consistent and reliable performance across different core configurations and simulation lengths.

4.2.1 Parallel Scaling Behavior and CPU Core Utilization

To evaluate the parallel performance of the simulation setup, I conducted a strong scaling test using a 1000-particle polydisperse system at a packing fraction of 0.25. Each simulation consisted of 10 million time-steps and employed the neighbor list parameters optimized in the previous section 4.2. The results are shown in Figure 12, which displays both the total CPU time per simulation and the number of microseconds of simulated time per day of real time, plotted as functions of the number of physical CPU cores.

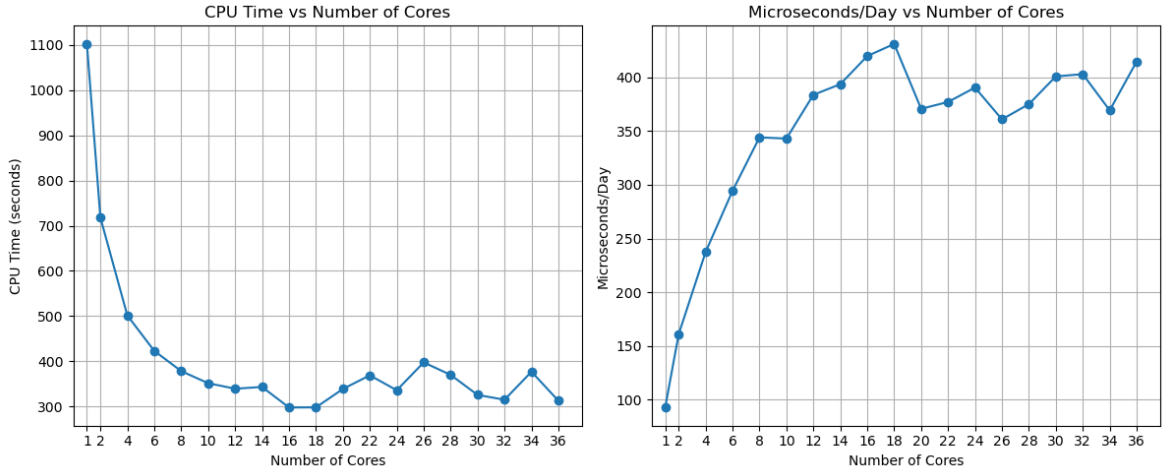


Figure 12: CPU time and microseconds of simulated time per one day of real time versus number of cores used to run a simulation. This is obtained for 10 M steps on a 0.25 packing polydisperse system of 1000 particles with neighbor list parameters skin 50 and delay 250.

The performance behavior can be divided into three regions. In the low-core range (from 1 to 4), the CPU time decreases rapidly, dropping from approximately 1100 seconds to around 500 seconds. Simultaneously, the throughput increases from about 100 to 240 μs/day. This reflects efficient scaling, where each processor performs a substantial amount of computational work, and the time spent on communication is minimal.

From 4 to 16 cores, the performance gains become less significant. Although both the runtime and throughput continue to improve, the rate of change slows. This behavior indicates that the cost of inter-process communication begins to grow. With fewer particles per core, the time required for data exchanges (such as ghost particle communication and collective operations for temperature, pressure, and MSD calculations) becomes more prominent and begins to limit overall efficiency.

The maximum throughput is observed at 18 cores, where the simulation achieves approximately 425 μs/day. However, this improvement compared to 16 cores (around 420 μs/day) is marginal. Beyond this point, performance plateaus and fluctuates. Increasing the number of cores further, such as to 24, 28, or 36, results in inconsistent throughput and even increases in wall time in some cases. This indicates the strong scaling limit of the system has been reached. At these higher core counts, the cost of communication and synchronization outweighs the computational benefit of distributing the work

more finely.

To provide a quantitative estimate of the serial portion of the simulation, I applied Amdahl's Law, which relates the maximum achievable speed-up S_N on N cores to the serial fraction f_{serial} of the computation:

$$S_N = \frac{1}{f_{\text{serial}} + \frac{1 - f_{\text{serial}}}{N}}. \tag{47}$$

Using the observed performance at 36 cores, the estimated serial fraction was approximately 0.27. This value is consistent with the known non-parallel components of the simulation, such as neighbor list rebuild checks and global reductions, which are repeated at every timestep.

Based on these results, I conclude that the most efficient parallel configuration for this system lies between 12 and 16 cores. At 16, the simulation achieves over 90% of the maximum throughput, while avoiding the inefficiencies and overhead observed at higher core counts. Although 18 cores provide a slightly higher throughput, the marginal gain does not justify the increased use of computational resources. In cases where full-node usage is required, it is more efficient to run multiple simulations in parallel—for example, two 18-core jobs—rather than a single 36-core simulation. This approach maintains high resource utilization while avoiding the limitations imposed by communication overhead.

These findings guide the choice of core count for all subsequent production simulations in this study and ensure that computational resources are used effectively without compromising performance.

4.3 Sanity checks

Before analyzing the simulation results in detail, it is essential to perform sanity checks to ensure the reliability and consistency of the data. Sanity checks are straightforward, intuitive tests used to validate whether a system behaves as expected under well-understood conditions. They do not guarantee correctness but help detect gross errors, implementation issues, or flawed assumptions early in the process.

In this context, I use sanity checks to compare simulated diffusion coefficients with theoretical predictions across a range of packing fractions and particle sizes.

Table 2: Simulated diffusion coefficients D_{exp} and theoretical diffusion coefficients D_{theo} in cm^2/s for each packing fraction ϕ (%). The simulated coefficients are obtained from monodisperse, 25 μs long simulations with A = 0.00001 eV and σ = 10.0 Å. The theoretical coefficient are calculated using Equation 32. Tables (a), (b), (c), (d), and (e) show data for radii 2, 10, 20, 30, and 40 nm respectively.

(a)

ϕ	D_{exp}	D_{theo}
1	1.34e-06	1.36e-06
3	1.29e-06	1.31e-06
5	1.29e-06	1.25e-06
10	1.08e-06	1.11e-06
15	1.00e-06	9.74e-07
20	8.24e-07	8.35e-07
25	7.47e-07	6.96e-07

(b)

ϕ	D_{exp}	D_{theo}
1	2.72e-07	2.73e-07
3	2.60e-07	2.62e-07
5	2.55e-07	2.50e-07
10	2.47e-07	2.23e-07
15	2.22e-07	1.95e-07
20	1.95e-07	1.67e-07
25	1.78e-07	1.39e-07

(c)

ϕ	D_{exp}	D_{theo}
1	1.39e-07	1.36e-07
3	1.35e-07	1.31e-07
5	1.35e-07	1.25e-07
10	1.32e-07	1.11e-07
15	1.29e-07	9.74e-08
20	1.22e-07	8.35e-08
25	1.14e-07	6.96e-08

(d)

ϕ	D_{exp}	D_{theo}
1	9.57e-08	9.09e-08
3	9.36e-08	8.72e-08
5	9.29e-08	8.35e-08
10	9.20e-08	7.42e-08
15	9.06e-08	6.49e-08
20	8.53e-08	5.57e-08
25	7.92e-08	4.64e-08

(e)

ϕ	D_{exp}	D_{theo}
1	6.98e-08	6.82e-08
3	6.95e-08	6.54e-08
5	6.93e-08	6.26e-08
10	6.87e-08	5.57e-08
15	6.79e-08	4.87e-08
20	6.44e-08	4.17e-08
25	6.32e-08	3.48e-08

Table 2 summarizes the diffusion coefficients obtained from both simulations and theoretical predictions at varying packing fractions (ϕ). A noticeable and consistent trend emerges: as the packing fraction increases, the diffusion coefficients decrease for all particle sizes. This behavior is expected, as higher ϕ values indicate a more crowded environment where particle mobility becomes increasingly restricted due to limited free space and frequent interactions. In a study conducted in 2010, it was found that the diffusion coefficient of macromolecules decreased significantly with increasing volume fraction, an effect attributed to both macromolecular crowding and hydrodynamic interactions.[49] Similar observations were reported by other researchers which demonstrated that protein dynamics in highly concentrated systems slow down substantially as a function of packing density.[50] These results align with those findings, suggesting that the present simulations reasonably capture the primary influences, such as steric hindrance and hydrodynamic effects, on diffusion under crowded conditions.

To further investigate how particle size influences diffusion at different crowding levels, the simulated diffusion coefficients were plotted against the inverse particle radius for each packing fraction, as shown in Figure 14. This representation is directly motivated by the theoretical relationship introduced in Equation (33), which predicts an inverse proportionality between diffusion and particle size in the dilute limit. According to what is explained in Section 2.3.2, this relation holds accurately

for low volume fractions, typically up to $\phi = 0.05$. The linear trends observed for all values of ϕ, with coefficients of determination (R^2) equal to 1.000, suggest that the simulation results follow this theoretical scaling. Smaller particles consistently display higher diffusion coefficients, and this pattern can be seen across all crowding conditions considered.

As the packing fraction increases, the slopes of the fitted lines tend to decrease. This reflects a general decline in diffusion that cannot be attributed to particle size alone. In more crowded environments, particles encounter spatial restrictions that limit their ability to move. To my understanding, these effects are not directly accounted for in the dilute-limit theoretical scaling but emerge naturally in the simulations due to the frequency of inter-particle interactions. The fact that this reduction in mobility is reproduced in the simulations suggests that the crowding-induced constraints are being reasonably represented, even though the Langevin model assumes an implicit solvent and size-dependent frictional damping.

Further comparisons can be made using Figure.13, which presents the simulated diffusion coefficients alongside the theoretical predictions for each particle size. At low packing fractions, the two sets of values remain close, indicating that the simulations reproduce the expected behavior under those conditions. As ϕ increases, however, deviations begin to appear, especially for larger particles. To my understanding, this may be due to the stronger influence of volume exclusion and particle caging effects, which impact larger particles more significantly. Similar outcomes have been reported in the literature, where diffusion was observed to decrease more rapidly for macromolecules of larger size as crowding increased.[49,50]

Altogether, these results support the idea that diffusion is influenced not only by the viscous drag and particle size, but also by crowding effects that become more prominent at higher volume fractions. The simulation results appear to reflect both factors, and the agreement with theoretical expectations at low ϕ, together with consistency with prior findings at higher densities, suggests that the model used here captures the relevant features of the system.

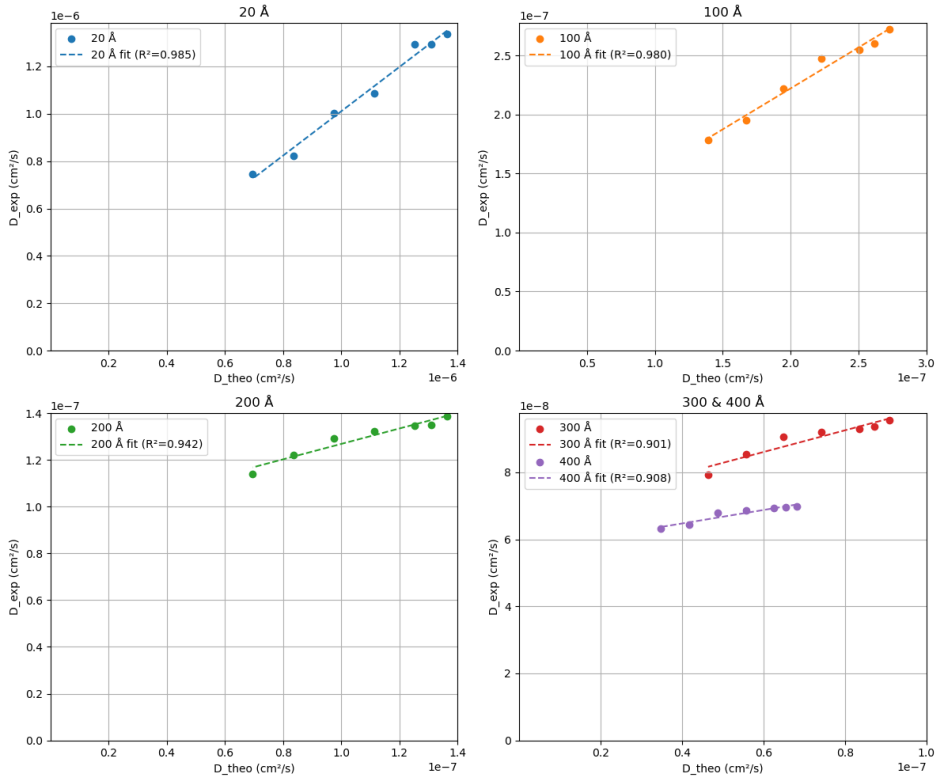


Figure 13: Comparison of theoretical and experimental diffusion coefficients (D_theo vs. D_exp) for spherical macromolecules of different sizes: 2 nm, 10 nm, 20 nm, 30 nm, and 40 nm in diameter. Each subplot shows data corresponding to a specific sphere size, with linear regression fits and associated R^2 values indicating the quality of the fit. Different colors and markers are used to visually distinguish the various sphere sizes and highlight the effect of particle size on diffusion behavior. Note that each subplot has individual y-axis scales due to large differences in D coefficients between particles from 2 to 40 nm.

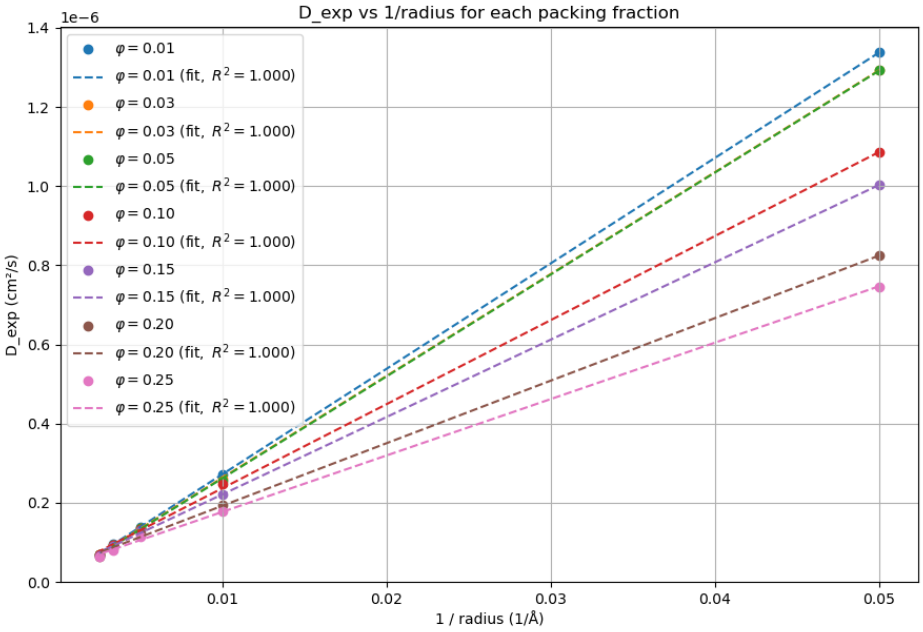


Figure 14: Experimental diffusion coefficients D_{exp} plotted against the inverse particle radius (Eq. 33) for for each packing fraction $\phi = 0.01$–0.25. Linear fits (with $R^2 \approx 1.000$) confirm the expected inverse relationship between diffusion and particle size, and show the decline of D_{exp} with increasing ϕ.

4.4 Effect of Hamaker constant

The Hamaker constant A is a multiplicative factor in both the attraction and repulsion terms of the pair style colloid potential function (see equations 13 and 14). In the U_R term, it appears as a $\frac{A}{37800}$ multiplying the rest of the expression, and in the U_A term it appears as $\frac{A}{6}$. This means that the value of A greatly affects the attractive nature of the interaction. For this reason, a wide range of values of $\text{A} = 1 \cdot 10^{-6}, 1 \cdot 10^{-5}, \ldots, 3.0\,eV$ have been studied for their effects on the particles interactions and to see how it affects their diffusion. The results of $25\,\mu s$ long simulations of 25% packing polydisperse system (lognormal distributed radii in the range from 2 nm to 40 nm) with varying A are presented and discussed in this section.

One of the primary effects of increasing A is the shift of the potential function to greater inter-particle distance and appearance of an attractive potential well (see Figure 3). This results in the interactions becoming more long range. This is immediately reflected on the computational cost of a simulation. Table 3 shows the increase of CPU time with increasing the attractive and long-range nature of the interactions. The computation time of a simulation of the same system with just changing the Hamaker constant from $A = 1 \cdot 10^{-6}\,eV$ to $A = 3.0\,eV$ increases ≈ 3 times. This is a direct consequence of the increase in the cutoff distances, which greatly increases the total number of interactions compared to the very short-range hard-sphere-like scenario of $A = 1 \cdot 10^{-6}$.

Table 3: Wall-clock CPU time in hour:min:second for each of the 25 μs long simulations with varying the value of Hamaker constant A in eV.

A	1.00E-06	1.00E-05	1.00E-04	1.00E-03	0.01	0.1	1	2	3
CPU time	3:41:40	3:57:36	4:14:56	4:47:41	5:37:22	7:11:42	9:23:26	9:23:58	11:37:08

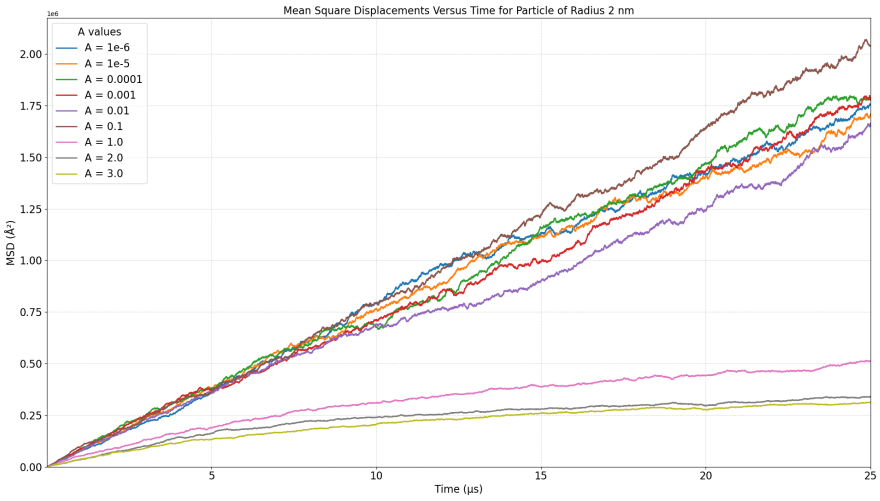
Mean square displacements (MSD) of the particles is greatly affected by the increase of attraction potential strength. The point of this simulation is to understand the evolution of MSD over time for three representative cases: 2 nm, 20 nm, and 40 nm radius particles are presented in Figure.15. The plots show the MSD curves of each group of particles for each of the 9 Hamaker constant values tested. Starting from the smallest particle, the first thing that can be noticed with the MSD curves of the radius 2 nm particles is the significant difference between the curves for $A < 1\,eV$ and $A > 1\,eV$. It appears that the mean square displacement is greatly reduced when $A > 1\,eV$. This is most likely a consequence of the 2 nm particles tending to stay in the regions close to each other over a long time due to mutual attraction, rather than diffusing freely. Furthermore, it can be seen that the MSDs for $A < 1\,eV$ are quite similar in the short time (up to $\approx 8\,\mu s$), and then diverge as the simulation progresses. This can be explained in the following way: the changes brought to the shape of the potential function by increasing A from $1 \cdot 10^{-6}\,eV$ to $0.1\,eV$ are nuanced compared to the changes for $A > 1\,eV$. For example, for the interaction of 2 nm particles, the potential at $A = 1 \cdot 10^{-6}$ is nearly hard-sphere like with only short-range repulsion present and a cutoff distance of less than 5 Å surface-to-surface distance. Increasing A to 0.0001 eV results in a slightly less steep repulsion and a slight increase in the cutoff distance of ≈ 1 Å. At $A = 0.01\,eV$, a potential well appears with the minimum of $-0.0006\,eV$ at ≈ 6 Å surface-to-surface distance. As this value is about 44 times less than kT at 303 K, this is still practically only a slightly longer-range repulsive potential. Increasing further to $A = 0.1\,eV$ results in a potential well minimum of $-0.006\,eV$, which is about 4 times less than kT, but can be considered a short-range repulsion and a very weak longer-range attraction potential. This leads to very similar particle behavior in the short time, with the differences coming to effect only at very long times ($> 10\mu s$). Within the $1 \cdot 10^{-6} \leq A \leq 0.1\,eV$ range, it can be seen that the MSD generally decreases with increasing A, and the outliers in the trend possibly indicating that even longer simulation times are needed to truly capture the subtle effects of A in this range of values. In the $A > 1.0\,eV$ range, it can be clearly seen that the MSDs not only decrease with A, but appear to remain relatively constant after $\approx 15\mu s$. This is most likely a consequence of the particles clustering together rather than moving freely in all directions.

Taking a look at the other extreme - the 40 nm radius particles, completely opposite effects of A are observed. The largest MSD is seen with the largest value of $A = 3.0\,eV$, followed by relatively large drops going down to $A = 2.0\,eV$ and then $A = 1.0\,eV$, and a clear separation from the mean square displacements at values of $A < 1.0\,eV$. Similarly to the 2 nm particles, it can again be observed that the mean squared displacements are quite similar in short-time (up to $\approx 5\mu s$, and then diverge as the time evolves. However, despite the divergence, they still remain relatively similar and no clear trend in how they change can be observed. The 40 nm particles are the rarest in the lognormal distribution. Furthermore, due to their low numbers and sheer size of the simulation box, they are typically not found close to each other and rarely are within the interaction distance. This implies that the movement of the 40 nm particles is mostly impacted by:a) their interactions with all other, smaller particles b) the general movement of all other, smaller particles occupying the remaining space, rather than just simple interactions between just the 40 nm particles. Based on those two reasoning, it can be assumed that the effects of the low values of the Hamaker constant are more complex and more subtle, thus most likely requiring very long simulation times to be quantified. As the products of radii of the interacting particles are found in the numerators of the potential function (see Eq 13), the strength of the interaction is also intensified by particle sizes. The potential functions for the 40 nm particle with

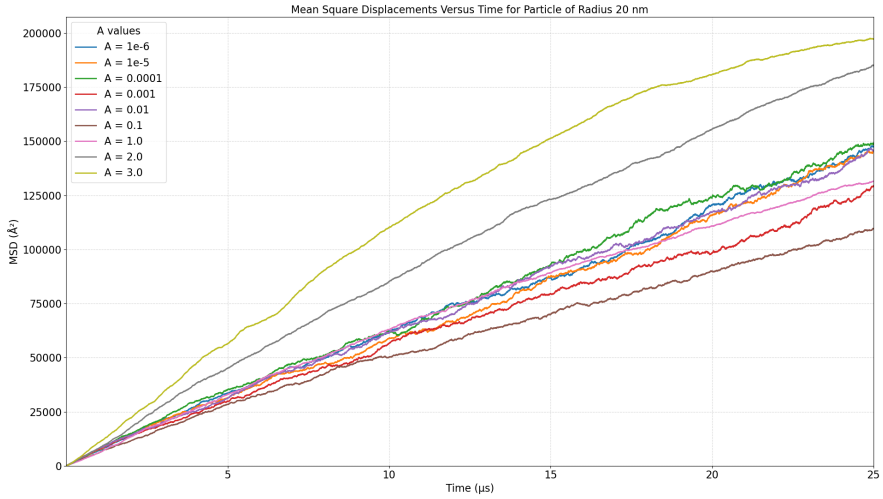
a 40 nm particle interactions have the potential well minima of ≈ -4.40, ≈ -8.80, and $-13.15\ eV$ for the values of $A = 1.0, 2.0$, and $3.0\,eV$ respectively. Said minima are all located around 5.5 Å surface-to-surface distance and are significantly greater than kT. This increase in the attraction strength is also followed by a great increase in the (surface-to-surface) cutoff distance from 2.8 Å at $A = 1 \cdot 10^{-6}$ to 500 rA at $A = 3.0$. This means that the 40 nm - 40 nm interactions potentials are only very short-range repulsive, strongly attractive at a short-range, and weak attractive at very long ranges. These long-range attractions steer the 40 nm particles towards each other, and the resulting increases in the particle accelerations can explain the drastic increase of MSD with increasing $A > 1\,eV$. In addition to this, the movement of the 40 nm particles may also be less hindered by the smaller particles which tend to cluster together at higher values of A, which leads to more free volume for the 40 nm particles to diffuse through, compared to when the smaller particles freely move around.

Lastly, the moderately sized particles with 20 nm radius show similar behavior to the 40 nm radius particles. The greatest values of MSD are seen for $A = 3.0\,eV$, followed by the $A = 2.0\,eV$ curve. The MSD at $A = 1.0\,eV$ is seen to be in the similar range as the mean squared displacements for $A < 1.0\,eV$. The fact that the fastest movement of particles is seen with the 2.0 and 3.0 A values can be explained with the same reasoning as with the 40 nm particles - two possibly synergistic effects of a) intensified long-range 20 nm - 20 nm attraction due to the $a_1 a_2$ terms in Eq 13 resulting in particle accelerations, and b) the smaller, $r < 10$ nm particles which dominate in numbers (due to lognormal distribution) possibly tend to stay close to each other rather than freely move around at large A values, thus liberating volume for the larger particles' movements. It can again be seen that the mean squared displacements of the lower A values (now with the inclusion of $A = 1.0\,eV$ as well) show similar values up to $\approx 10\,\mu s$, and diverging later on. This again suggests that the particles are not immediately affected by the interaction strengths, but that values of A do impact how will particles move on the average in the long run.

(a)



(b)



(c)

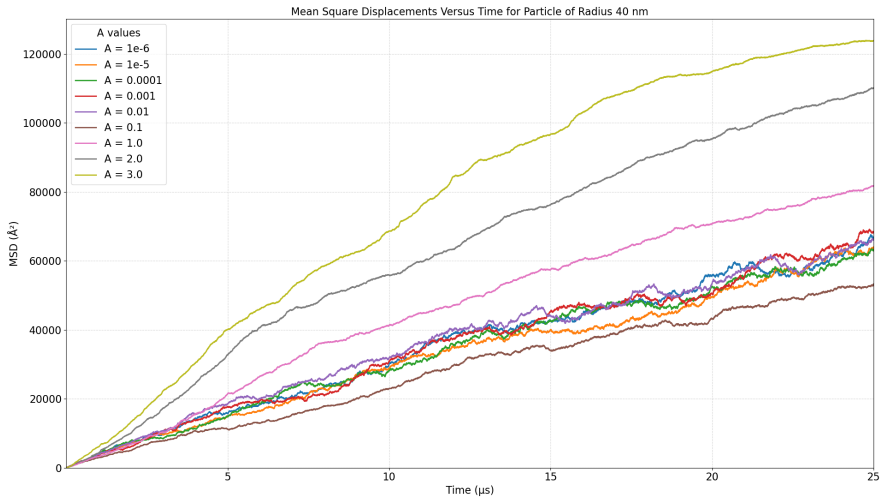


Figure 15: Mean square displacements of particles with radius 2 nm (a), 20 nm (b), and 40 nm (c) for values of Hamaker constant ranging from $1 \cdot 10^{-6}$ eV to 3.0 eV.

The mean square displacements of the 2, 10, 20, 30, and 40 nm radius particles are linearly fitted and the respective diffusion coefficients are extracted as $\frac{1}{6}d(MSD(t))/dt$. The resulting values are collected and presented in Table 4 for all the values of A.

The 2 nm particle diffusion coefficients generally decrease with increasing A. This is particularly noticeable for $A > 0.1\,eV$. The values in the range of $A < 0.1\,eV$ exhibit very similar values around $1.17 \cdot 10^{-6} \pm 6.60 \cdot 10^{-8}\,cm^2/s$. The 40 nm particles see an opposite trend - a steady increase of diffusion coefficients with increasing A for $A > 0.1$, with the $D_{40nm}(A = 3.0)/D_{40nm}(A = 0.1) \approx 2.64$. While the D_{40nm} coefficients generally also increase with increasing A from $1 \cdot 10^{-6}$ to 0.01, they show rather similar values around $4.52 \cdot 10^{-8} \pm 1.49 \cdot 10^{-9}\,cm^2/s$. Similarly to the 40 nm particles, the 20 nm radius ones also show a steady increase in D coefficients with increasing A in the $A > 0.1$ range. The coefficients for $A < 0.1$ don't show clear trends, but can all be found to be around

$9.53 \cdot 10^{-8} \pm 6.17 \cdot 10^{-9}\, cm^2/s$. These numbers all quantify what has been visually observed with the mean squared displacements in the discussion above. The particles with the 10 nm radius, whose mean squared displacements have not been visualized, show the following behavior: D_{10nm} coefficients generally decrease with increasing A up to 1.0 eV, and then increase in the range from $A = 1.0$ to $A = 3.0\, eV$. This shows a somewhat intermediate behaviour between the two extremes of the smallest particles D_{2nm} coefficients decreasing with increasing A, and the largest particles D_{40nm} coefficients generally increasing with increasing A. Furthermore, it can be noted that the D_{10nm} coefficients vary little in the $1 \cdot 10^{-6} \leq A \leq 0.01$ range, all being around $\approx 2.13 \cdot 10^{-7} \pm 1.0 \cdot 10^{-8}\, cm^2/s$, which indicates somewhat indifferent behaviour of the 10 nm particles in respect to changing A in that range. Lastly, the 30 nm radius particles show behavior very close to the 40 nm radius ones - D_{30nm} coefficient at $A = 3.0$ is slightly over 2 times higher than the D_{30nm} at $A = 0.1\, eV$. The behaviour in the low limits of A doesn't show clear trends and likely needs longer simulation times for the more complex and subtle effects to come into play.

Table 4: Simulated diffusion coefficients D (in cm^2/s) for various values of the interaction strength A in eV and particle radii $r = 2,\ 10,\ 20,\ 30,\ 40\,\mathrm{nm}$. Values are obtained from 25 μs long simulations at 25% packing polydisperse systems.

A	D_2	D_{10}	D_{20}	D_{30}	D_{40}
1.00E-06	1.21e-06	2.27e-07	9.51e-08	6.56e-08	4.57e-08
1.00E-05	1.18e-06	2.16e-07	9.39e-08	5.68e-08	4.31e-08
1.00E-04	1.23e-06	2.12e-07	1.03e-07	5.96e-08	4.44e-08
1.00E-03	1.17e-06	2.15e-07	8.62e-08	6.24e-08	4.62e-08
0.01	1.06e-06	2.14e-07	9.82e-08	6.12e-08	4.68e-08
0.1	1.35e-06	1.96e-07	7.64e-08	5.41e-08	3.74e-08
1	1.84e-07	1.12e-07	8.76e-08	7.79e-08	6.01e-08
2	9.66e-08	1.61e-07	1.31e-07	1.14e-07	8.17e-08
3	2.57e-07	1.83e-07	1.55e-07	1.20e-07	9.88e-08

4.5 Step-size distributions

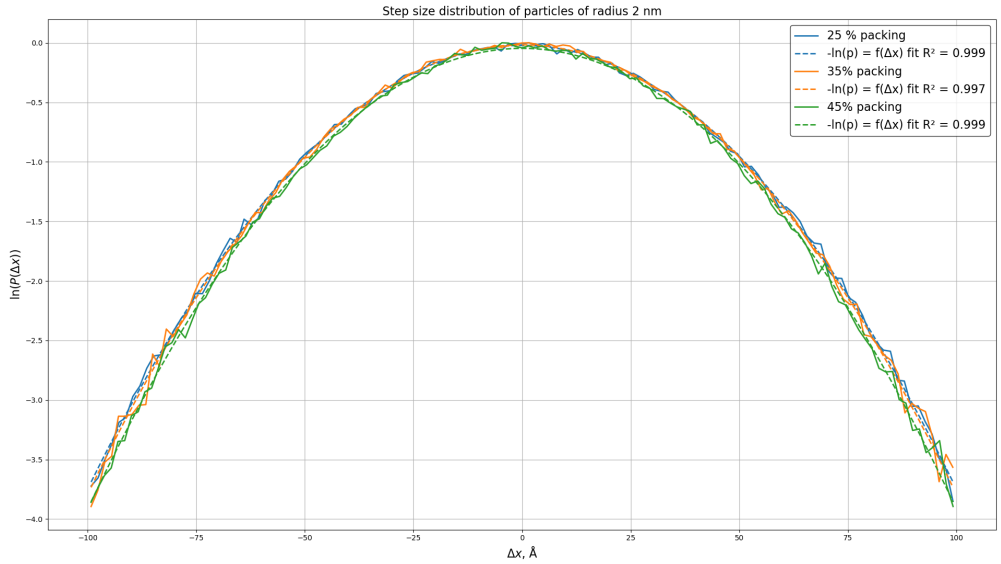
One of the objectives of this work is to investigate whether polydispersity can be contributing to the heterogeneous dynamics observed in the cytoplasm. To answer this question, step-size distributions of particles obtained from simulations are analyzed.

Both polydisperse and monodisperse systems are simulated over a total duration of $0.5\, ms$. Step-size (Δx) distributions are calculated using time intervals of $\Delta t = 50\, ns$, resulting in a total of 10000 Δx values per particle of a given radius over the full simulation time. This ensures sufficient statistical sampling for the analysis. For each radius, the resulting Δx values are binned according to the Freedman-Diaconis rule[51] , and the probability of each step size, $p(\Delta x_i)$, is calculated as $N(\Delta x_i)/N(\Delta x)$, where $N(\Delta x_i)$ is the number of occurrences in bin i, and $N(\Delta x)$ is the total number of displacements. The logarithms of the resulting probabilities $ln(p(\Delta x_i))$ are then plotted against the respective displacements Δx_i. Any sign of exponential tails would indicate deviation from the Brownian motion, and would point to some underlying heterogeneity, which in turn could help explain the behaviour seen in the cytoplasm.

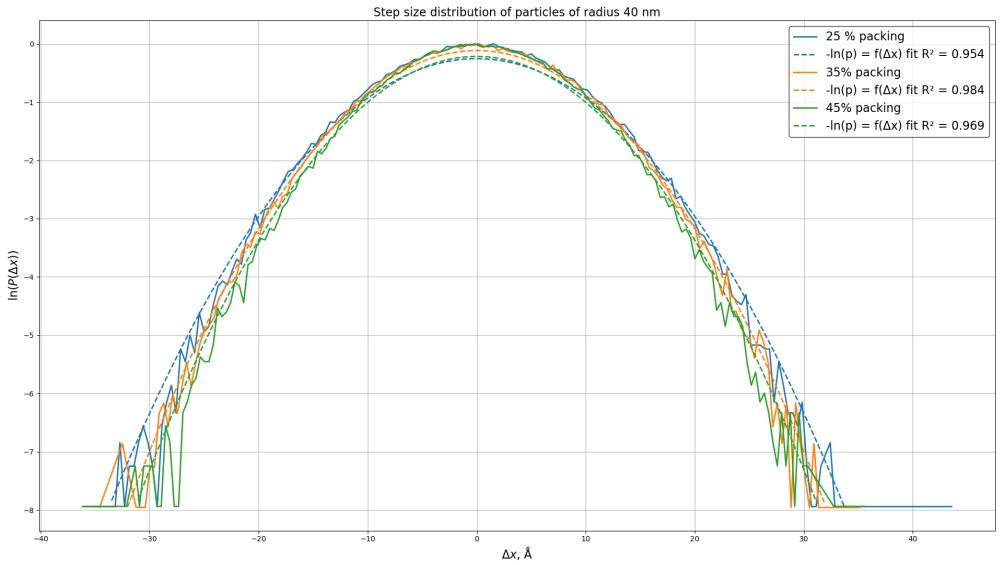
This analysis is done by considering both the effects of crowding (by increasing packing density), and the possible effects of the Hamaker constant A. The results are presented and discussed below.

4.5.1 Effect of packing density

Figure 16 shows results obtained for the 2 nm and 40 nm particles in a polydisperse environment across 25%, 35%, and 45% packing densities. Hamaker constant $A = 1 \cdot 10^{-5}\, eV$ is applied in order to mimic hard-sphere-like behavior. In the case of the 2 nm particle, all 3 obtained distributions appear to be a result of an underlying Gaussian distribution, as indicated by the $R^2 > 0.99$ polynomial fit coefficients (see Section 2.3.3). This is a strong indicator that the motion of these particles in such an environment is Brownian in the $0.5\, ms$ timescale. 40 nm particles appear to exhibit a slight deviation from a perfect Gaussian distribution of the displacements, reflected by the slight deviation from the $ln(p(\Delta x)) \propto -(\Delta x)^2$ relation. This most likely indicates that the largest particles also exhibit Brownian motion over such timescales. However, a question still remains whether further increasing packing density may somehow induce heterogeneity. Furthermore, even longer timescales might be necessary to analyze in order to confirm this. Therefore, achieving longer time simulations ($t > 0.5\, ms$) still remains a goal, and a challenge after this work. This is particularly made difficult because a) increasing packing density while keeping all other settings constant will increase both the number of particles and the number of interactions (due to higher proximity of particles on average) and thus increasing computational cost and b) the necessity to use a small simulation timestep needed to faithfully simulate hard-sphere-like interactions.



(a)



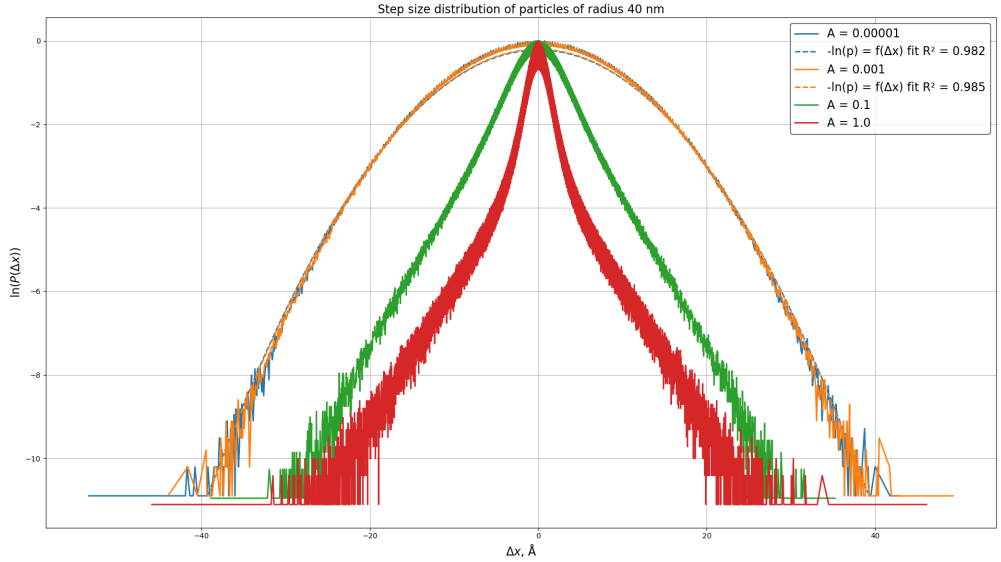
(b)

Figure 16: Step-size distributions of the 2 nm (17a) and 40 nm (17b) radius particles in polydisperse environment.

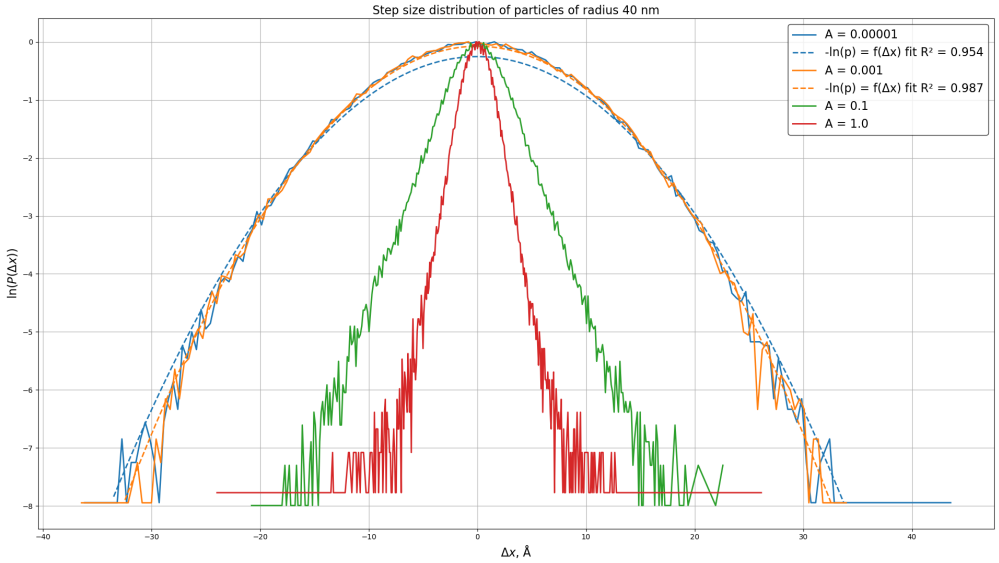
4.5.2 Effect of Hamaker constant

The possible effect of A on heterogeneity is also examined. Simulations are ran for mono- and polydisperse systems with a 25% packing density, with values of $A = 1 \cdot 10^{-}5, 0.001, 0.1,$ and 1.0. Results obtained for the 40 nm particles in the mono- and polydisperse environments are collected and shown in Figure 17. Considering first the monodisperse 40 nm particle, it can be seen that for the two of the lowest A values, step-size distributions are Gaussian. This confirms to so far seen picture - particles with a hard-sphere-like interactions with 25% packing in a monodisperse system follow Brownian dynamics. These two curves also align with the expectations from the sanity checks. However, increasing

A to 0.1 and 1.0 eV induces significant deviation from the Gaussian distribution. This is reflected in the shape of the $ln(p(\Delta x))$ distribution resembling more a $-|\Delta x|$ dependency, rather than the $-(\Delta x)^2$ one. This is most likely a direct consequence of the attractive forces at play, taking over the Langevin stochastic forces and being the dominant factor in the particle movement. The thicker appearance of the monodisperse distribution curves compared to the polydisperse one is due to the difference in number of particles - both simulations contained 1000 particles, which in a lognormal distributed radii polydisperse system results in 13 particles of radius 40. Taking a look at the 40 nm particle in the polydisperse environment, it is again evident that for $A < 0.1$, particles act as hard spheres, and the motion is Brownian, despite the polydispersity and high packing. However, once strong short-range attractions and weak long-range attractions are introduced by increasing the Hamaker constant to $A > 0.1$ eV, particle movement becomes heavily affected by the interactions with its surroundings. Both the $A = 0.1$ and $A = 1.0$ distributions show a general $-|\Delta x|$ shape, thus confirming existence of exponential tails and a presence of non heterogeneity in the simulated dynamics. This implies that the polydispersity and packing density might after all contribute to the heterogeneity of the cytoplasm. However, in order to truly confirm this hypothesis, it is evident that great attention must be paid to the modeling of the interactions.



(a)



(b)

Figure 17: Step-size distributions of the 40 nm radius particles in mono- (17a) and polydisperse (17b) environments with 25% packing.

5 Conclusion

This master thesis was set out with an ambitious goal: to build a series of steps toward a physics-based and biologically sound, long-timescale molecular dynamics simulation of a small but representative portion of cytoplasm, and to ask whether the macromolecular polydispersity alone could help explain the non-heterogeneity in the colloidal dynamics as reported by the experimentalists. Although the road to this goal proved steeper than expected, and key biological questions remain open, the work has nonetheless produced a solid computational groundwork for reaching simulation times in the range of milliseconds, revealed some methodological pitfalls regarding the colloid potential and simulations of highly polydisperse systems, and pointed to clear priorities for the next phase of the study.

The central piece of work of this thesis is a fully automated workflow, combining Python scripting with LAMMPS, that can a) generate discrete log-normal radius distributions spanning arbitrary ranges, b) pack a large number of colloids in user-defined volume fractions without overlaps and up to 45% packing densities, c) dynamically calculate energy-based cutoff distances for given choices of the A and σ parameters in the colloid potential function, d) assign physically based Langevin thermostat damping times at a desired temperature and viscosity to all individual particle types, and e) automatically adjust and prepare a ready-to-run input file for any choice of system settings. This allowed going from a once-manual and error-prone setup to a fully automated way to generate dozens of reproducible systems, which should enable quicker progress for the future simulations in this study. For example, this script allows quick benchmarking of the neighbor list parameters - skin distance and delay value, which can be system-specific and therefore can require individual parameterization.

Using that infrastructure, more than sixty production runs were successfully completed (as a result of many more trial and error or parametrization runs), ranging from dilute to 45% packing, and from hard-sphere-like $A = 1 \cdot 10^{-6}\, eV$ to strongly attractive $A = 3.0\, eV$ interactions, with the calculations reaching 0.5 ms of simulation time within 20-40 hours of real time. From these simulations, several sanity checks have provided technical reassurance on the quality of the simulation framework. For $A = 0.00001\, eV$, which is supposed to model a hard-sphere-like behavior, the diffusivity of each monodisperse system scales inversely with particle radius, thus confirming that the simulations are able to reproduce behavior as expected by the Stokes-Einstein equation. Furthermore, the diffusion coefficients agree with the first-order approximation to diffusivity dependence on the packing fraction, to within $\pm 10\%$ (where the error becomes larger the higher the packing and the larger the particles - i.e. when hydrodynamic effects cannot be neglected). These confirmed that the choice of the Langevin thermostat and NVE time-integration do not introduce any anomalies.

Beyond these baseline tests, two qualitative insights stand out as a result of the remaining simulations. First, macromolecular size by itself is not enough to generate cytoplasm-like exponential tails, at least on the sub 1 ms scale reached here. Neither dense packing (up to 45%), nor the high variety of lognormal distributed radii in the range from 2 nm to 40 nm could alone distort the Gaussian step-size distributions. Second, the colloid potential function is tricky and requires careful parametrization on a per-interaction basis. This is because of the colloid potential function dependency on the $a_i a_j$ terms, where a_i and a_j are the radii of two interacting particles i and j. For simplicity, and because this work was primarily focused on achieving millisecond scale simulations, the same values of A and σ were applied for all particle interactions, regardless of their size. This in turn produces asymmetric behavior, where for the same value of A, two smaller particles may interact under a soft-repulsion mode, while two large particles may interact under a strong attraction mode. This clearly necessitates

carefully tailoring and optimizing the A and σ parameters for each individual interaction, until truly biological behavior can be replicated. This is indeed confirmed by the findings that the inclusion of a mild attraction with $A = 0.1\ eV$ can reproduce the exponential tails seen in the experimental step-size distributions.

Although the biological questions remain unanswered, this thesis has made solid progress into bringing the colloid potential equations into a reproducible and reliable pair style for modeling cytoplasmic crowding. Furthermore, it revealed that with appropriate parametrization, milliseconds-long simulations are within reach. The immediate next steps that are going to be taken in the further development of this project will involve a) careful parametrization, and possible modification of the colloid potential function until the results can be fit to the data obtained by the experimentalists in the group, b) going beyond simple Langevin thermostat and NVE integration, and attempting to introduce hydrodynamic effects which are non-negligible for large particles and large densities, and c) making further effort to reach 10-50 μs simulation times that are needed to truly capture all the subtle effects at play in a complex system such as the cytoplasm.

References

(1) Zeng, H. What Is a Cell Type and How To Define It? *Cell* **2022**, *185*, 2739–2755.

(2) Kim, H.; Delarue, M. Dynamic Structure of the Cytoplasm. *Curr. Opin. Cell Biol.* **2025**, *94*, 102507.

(3) Sakai, K.; Kondo, Y.; Goto, Y.; Aoki, K. Cytoplasmic Fluidization Contributes to Breaking Spore Dormancy in Fission Yeast. *Proc. Natl. Acad. Sci. U.S.A.* **2024**, *121*, e2405553121.

(4) Valverde-Méndez, D.; Sunol, A. M.; Bratton, B. P.; Delarue, M.; Hofmann, J. L.; Sheehan, J. P.; Gitai, Z.; Holt, L. J.; Shaevitz, J. W.; Zia, R. N. Macromolecular Interactions and Geometrical Confinement Determine the 3D Diffusion of Ribosome-Sized Particles in Live *Escherichia coli* Cells. *bioRxiv* **2024**, Preprint posted March 28, 2024, DOI: 10.1101/2024.03.27.587083.

(5) Alric, B.; Formosa-Dague, C.; Dague, E.; Holt, L. J.; Delarue, M. Macromolecular Crowding Limits Growth under Pressure. *Nat. Phys.* **2022**, *18*, 411–416.

(6) Meriem, Z. B.; Mateo, T.; Faccini, J.; Denais, C.; Dusfour-Castan, R.; Guynet, C.; Merle, T.; Suzanne, M.; Di-Luoffo, M.; Guillermet-Guibert, J. A Microfluidic Mechano-Chemostat for Tissues and Organisms Reveals That Confined Growth Is Accompanied with Increased Macromolecular Crowding. *Lab Chip* **2023**, *23*, 4445–4455.

(7) Holt, L. J.; Delarue, M. Macromolecular Crowding: Sensing without a Sensor. *Curr. Opin. Cell Biol.* **2023**, *85*, 102269.

(8) Ellis, R. J. Macromolecular Crowding: Obvious but Underappreciated. *Trends Biochem. Sci.* **2001**, *26*, 597–604.

(9) Neurohr, G. E.; Amon, A. Relevance and Regulation of Cell Density. *Trends Cell Biol.* **2020**, *30*, 213–225.

(10) Trovato, F.; Tozzini, V. Diffusion within the Cytoplasm: A Mesoscale Model of Interacting Macromolecules. *Biophys. J.* **2014**, *107*, 2579–2591.

(11) Rapaport, D. C., *The Art of Molecular Dynamics Simulation*, 2nd; Cambridge University Press: Cambridge, U.K., 2004.

(12) Frenkel, D.; Smit, B., *Understanding Molecular Simulation: From Algorithms to Applications*, 2nd; Computational Science Series; Academic Press: San Diego, CA, 2002.

(13) Verlet, L. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.* **1967**, *159*, 98–103.

(14) Wahnström, G. Molecular Dynamics: Lecture Notes, University of Gothenburg [Online lecture notes], accessed May 17, 2025, Gothenburg, Sweden, 2018.

(15) González, M. A. Force Fields and Molecular Dynamics Simulations. *Collection SFN* **2011**, *12*, 169–200.

(16) Lenhard, J.; Stephan, S.; Hasse, H. On the History of the Lennard-Jones Potential. *Ann. Phys.* **2024**, *536*, 2400115.

(17) Wang, X.; Ramírez-Hinestrosa, S.; Dobnikar, J.; Frenkel, D. The Lennard-Jones potential: when (not) to use it. *Phys. Chem. Chem. Phys.* **2019**, *21*, 18958–18969.

(18) Everaers, R.; Ejtehadi, M. R. Interaction Potentials for Soft and Hard Ellipsoids. *Phys. Rev. E* **2003**, *67*, 041710.

(19) Plimpton, S. J.; Developers, L. pair$_s$*tylecolloidcommand* accessed June 1, 2025, https://docs.lammps.org/pair_colloid.html.

(20) Dhont, J. K. G., *An Introduction to Dynamics of Colloids*; Studies in Interface Science, Vol. 2; Elsevier: Amsterdam, 1996.

(21) Toxvaerd, S.; Heilmann, O. J.; Dyre, J. C. Energy Conservation in Molecular Dynamics Simulations of Classical Systems. *J. Chem. Phys.* **2012**, *136*, 224106.

(22) *Theoretical and Computational Chemistry: Foundations, Methods and Techniques*; Andrés, J., Bertran, J., Eds.; Ciències Experimentals, Vol. 11; Publicacions de la Universitat Jaume I: Castelló de la Plana, Spain, 2007, 448 pp.

(23) Jensen, F., *Introduction to Computational Chemistry*, 2nd; John Wiley Sons: Chichester, West Sussex, England, 2007.

(24) Plimpton, S. J.; Developers, L. fix langevin command, LAMMPS Molecular Dynamics Simulator [Online], accessed May 20, 2025, 2025.

(25) Nosé, S. A Molecular Dynamics Method for Simulations in the Canonical Ensemble. *Mol. Phys.* **1984**, *52*, 255–268.

(26) Hoover, W. G. Canonical Dynamics: Equilibrium Phase-Space Distributions. *Phys. Rev. A* **1985**, *31*, 1695–1697.

(27) Martyna, G. J.; Klein, M. L.; Tuckerman, M. Nosé–Hoover Chains: The Canonical Ensemble via Continuous Dynamics. *J. Chem. Phys.* **1992**, *97*, 2635–2643.

(28) Allen, M. P.; Tildesley, D. J., *Computer Simulation of Liquids*; Oxford University Press: Oxford, U.K., 1987.

(29) Plimpton, S. J.; Developers, L. fix viscous command, LAMMPS Molecular Dynamics Simulator [Online], accessed May 20, 2025, 2025.

(30) Reichert, M. Hydrodynamic Interactions in Colloidal and Biological Systems, Ph.D. Thesis, Konstanz, Germany: University of Konstanz, 2006.

(31) Długosz, M.; Trylska, J. Diffusion in Crowded Biological Environments: Applications of Brownian Dynamics. *BMC Biophys.* **2011**, *4*, 3.

(32) Löwen, H.; Szamel, G. Long-Time Self-Diffusion Coefficient in Colloidal Suspensions: Theory versus Simulation. *J. Phys.: Condens. Matter* **1993**, *5*, 2295–2306.

(33) Brady, J. F.; Bossis, G. Stokesian Dynamics. *Annu. Rev. Fluid Mech.* **1988**, *20*, 111–157.

(34) Mark, P.; Nilsson, L. Structure and Dynamics of the TIP3P, SPC, and SPC/E Water Models at 298 K. *J. Phys. Chem. A* **2001**, *105*, 9954–9960.

(35) Karplus, M.; McCammon, J. A. Molecular Dynamics Simulations of Biomolecules. *Nat. Struct. Biol.* **2002**, *9*, 646–652.

(36) Fogolari, F.; Brigo, A.; Molinari, H. The Poisson–Boltzmann Equation for Biomolecular Electrostatics: A Tool for Structural Biology. *J. Mol. Recognit.* **2005**, *18*, 267–282.

(37) Honig, B.; Nicholls, A. Classical Electrostatics in Biology and Chemistry. *Science* **1995**, *268*, 1144–1149.

(38) Ermak, D. L.; McCammon, J. A. Brownian Dynamics with Hydrodynamic Interactions. *J. Chem. Phys.* **1978**, *69*, 1352–1360.

(39) Einstein, A. On the Movement of Small Particles Suspended in a Stationary Liquid Demanded by the Molecular-Kinetic Theory of Heat. *Ann. Phys.* **1905**, *322*, 549–560.

(40) Dhont, J. K. G. In *Lecture Notes of the 43rd IFF Spring School 2012,* Schriften des Forschungszentrums Jülich / Reihe Schlüsseltechnologien / Key Technologies*, Vol. 33*, Angst, M., Brückel, T., Richter, D., Zorn, R., Eds.; Forschungszentrum Jülich GmbH, JCNS, PGI, ICS, IAS: Jülich, 2012.

(41) Rouaud, M., *Probability, Statistics and Estimation: Propagation of Uncertainties in Experimental Measurement (Short Edition)*; Self-published under CC BY-NC 4.0 License: 2017.

(42) Plimpton, S. J. Fast Parallel Algorithms for Short-Range Molecular Dynamics, 1995.

(43) Thompson, A. P.; Aktulga, H. M.; Berger, R.; Bolintineanu, D. S.; Brown, W. M.; Crozier, P. S.; in 't Veld, P. J.; Kohlmeyer, A.; Moore, S. G.; Nguyen, T. D., et al. LAMMPS – A Flexible Simulation Tool for Particle-Based Materials Modeling at the Atomic, Meso, and Continuum Scales. *Comput. Phys. Commun.* **2022**, *271*, 108171.

(44) Plimpton, S. J.; Thompson, A. P.; Developers, L. units command, LAMMPS Molecular Dynamics Simulator [Online], accessed May 13, 2025, 2025.

(45) Plimpton, S. J.; Thompson, A. P.; Developers, L. atom$_s$*tylecommand*, LAMMPS Molecular Dynamics Simulator [Online], accessed May 13, 2025, 2025.

(46) Plimpton, S. J.; Thompson, A. P.; Developers, L. boundary command, LAMMPS Molecular Dynamics Simulator [Online], accessed May 13, 2025, 2025.

(47) Plimpton, S. J.; Developers, L. LAMMPS Developer Guide: Neighbor Lists, LAMMPS Molecular Dynamics Simulator [Online], accessed June 8, 2025, 2025.

(48) Fincham, D. Choice of Timestep in Molecular Dynamics Simulation. *Comput. Phys. Commun.* **1986**, *40*, 263–269.

(49) Ando, T.; Skolnick, J. Crowding and Hydrodynamic Interactions Likely Dominate in vivo Macromolecular Motion. *Proc. Natl. Acad. Sci. U.S.A.* **2010**, *107*, 18457–18462.

(50) Hwang, J.; Kim, J.; Sung, B. J. Dynamics of Highly Polydisperse Colloidal Suspensions as a Model System for Bacterial Cytoplasm. *Phys. Rev. E* **2016**, *94*, 022614.

(51) Freedman, D.; Diaconis, P. On the histogram as a density estimator: L2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* **1981**, *57*, 453–476.

A Appendix A

The following is the default input script for the colloid example provided by LAMMPS. Throughout this work, several modifications have been made to this input in order to adapt it to the objectives of the project. Including the original version here allows the reader to clearly identify and compare the changes implemented in the customized inputs discussed in the main text.

Default input for colloid

```
units lj
atom_style sphere
dimension 2

lattice sq 0.01
region box block 0 30 0 30 -0.5 0.5
create_box 2 box
create_atoms 1 box

set group all type/fraction 2 0.96 23984

set type 1 mass 9
set type 2 mass 1

velocity all create 1.44 87287 loop geom

neighbor 1 multi
neigh_modify delay 0
comm_modify mode multi

pair_style colloid 12.5
pair_coeff 1 1 1.0 1.0 5.0 5.0 12.5
pair_coeff 1 2 5.0 1.0 5.0 0.0 7.0
pair_coeff 2 2 10.0 1.0 0.0 0.0 2.5

fix 1 all npt temp 2.0 2.0 1.0 iso 0.0 1.0 10.0 drag 1.0 &
mtk no pchain 0 tchain 1
fix 2 all enforce2d


thermo_style custom step temp epair etotal press vol
thermo 1000

timestep 0.005

run 50000
```

B Appendix B

1) Number of H_2O particles, approximately:

- $V_t = 5000^3\,\text{Å}^3$
- packing $= 0.25 \Rightarrow V_w = 5000^3 - 0.25 \cdot 5000^3 = 0.25 \cdot 5000^3\,\text{Å}^3$

$$V_w = 9.28 \cdot 10^{-20}\,\text{m}^3$$

$$\rho_{H_2O} \approx 1000\,\text{kg/m}^3$$

$$\Rightarrow m_w \approx 9.375 \cdot 10^{-17}\,\text{kg} = 9.375 \cdot 10^{-14}\,\text{g}$$

$$\Rightarrow n_{H_2O} = \frac{m}{M} = \frac{9.375 \cdot 10^{-14}\,\text{g}}{18\,\text{g/mol}} \approx 5.21 \cdot 10^{-15}\,\text{mol}$$

$$\Rightarrow N_{H_2O} = n \cdot N_A \approx 3.14 \cdot 10^9 \quad \text{water molecules}$$