Contents

1. Introduction

The mesoscopic scale is, perhaps, one of the most difficult frontiers in physics. At least, conceptually it is for me. It lies in that ambiguous territory where neither the mathematical clarity of microscopic laws or the comforting simplicity of macroscopic approximations can be fully trusted. It is a domain of negotiation, where intuition must constantly be rebuilt, and where the language of isolated equations gives way to that of collective behaviour.

On the microscopic side, mathematics often retains its authority. The systems are small enough, the variables are explicit and — at least in principle — the governing equations can be written down exactly. Quantum mechanics stands as the clearest example of this triumph: an austere yet complete framework capable of capturing atoms, molecules and the very first signs of disorder, provided one accepts the mental gymnastics it demands.

At the opposite extreme, the macroscopic world feels reassuringly coarse. Details fade into averages and fluctuations are smoothed out by numbers. We no longer track the motion of each component, but rather treat the entire ensemble as if it were a few single beings. We speak of global laws — thermodynamics, hydrodynamics or even social dynamics. It is the scale of our daily experience, the one our senses and instincts have evolved to understand. We are creatures of the macroscopic world.

I know this is a personal take, but the mesoscopic scale seems to be where our confidence as physicists firstly fades. Our intuition, shaped by the macroscopic world we inhabit, keeps trying to tell us how things should behave. But at this scale, it is often wrong. The individuals — the particles, the degrees of freedom — still exist, still move, still collide. They have not yet dissolved into the smooth averages, and their presence makes every description painfully complex. Who would dare to write 10^{24} coupled equations to describe a gas? And yet, that is what the mesoscopic world silently demands of us: to think of the collective without ever forgetting the individual.

And yet, our predecessors have not faced this complexity empty-handed. Over time, we have learned to think differently — to accept that not every variable deserves to be followed, that meaning can emerge from noise. The mesoscopic world has taught us to reason from the bottom up, to let the collective arise from the individual, and to translate collisions into probabilities. From these attempts were born the stochastic formalisms, the language of open systems, and the statistical methods that still guide us today — as fragile, as approximated, it can be but still remarkably alive.

The work presented here follows the same lineage. We will begin with a chain in contact with a solvent — or, to borrow the physicist's jargon, with a reservoir. The question is how this environment influences the chain, how it stirs, folds, and makes it vibrate. It is, in essence, a continuation of the ideas first explored by Langevin, Einstein and Smoluchowski, extended here to the case where several Brownian particles are harmonically coupled.

The formalism itself dates back to the work of Paul E. Rouse, who first proposed a simple yet powerful model for the dynamics of polymer chains. The role of the computer in this story is 3–fold: to generate the noise that drives the dynamics, to integrate the resulting equations of motion and finally to offer us the chance to observe, to interpret and perhaps to glimpse a bit of order within the fluctuations.

For this, I chose to work in Common Lisp. That is a language whose core is as old as the Rouse model itself, and one that encourages the same kind of dialogue between abstraction and concreteness. It feels appropriate that model born of simplicity should be explored through a language built for expressing ideas. The rest of this report is an attempt to retrace that journey — from equations to motion, from noise to structure.

2. Theoretical aspects

The goal here is not to recount the entire history of Brownian theory, but to offer a few insights into it. Bertrand Duplantier gives a far more exhaustive and elegant account in his *Séminaire Poincaré* lecture (2005) [1] — one can only recommend reading the transcript.

It all began, quite innocently, in 1827 when the botanist Robert Brown observed that pollen grains suspended in water exhibited an erratic and never-ending motion. The phenomenon was puzzling: it persisted even when the grains were no longer alive, ruling out any biological cause. Its physical origin would remain debated for nearly a century.

In 1905, Albert Einstein [2] proposed a quantitative explanation. He showed that this irregular motion could be understood as the visible consequence of molecular collisions. Instead of following the particle itself, he described the probability of finding it at a certain position in time, using the diffusion equation.

A few months later, the physicist Marian Smoluchowski [3] offered an alternative viewpoint. Where Einstein's approach was statistical, Smoluchowski's was kinetic and mechanical. He pictured the suspended particle as being constantly bombarded by the molecules of the fluid, each collision imparting a tiny impulse. From this more intuitive picture, he recovered the same diffusion law, but with an emphasis on the discrete nature of the random kicks.

Finally, in 1908, Paul Langevin [4] brought synthesis and clarity. He wrote down directly the equation of motion for a Brownian particle — developed in the next section.

Since then, the formalization of these stochastic equations has continued, notably through Itô calculus. The fact that such equations now stand at the core of modern financial models was, one suspects, motivation enough for mathematicians to polish their theory. Physicists, meanwhile, took another path. They introduced external potentials, coupled multiple particles and explored how collective fluctuations emerge from these stochastic laws. These efforts gave rise to techniques such as electrophoresis and, of course, to the Rouse model [5], which remains until this day an insightful description of polymer dynamics. The next section recalls the essence of Langevin's idea, an equation simple enough to fit in a single line, yet rich enough to describe a century of stochastic motion.

2.1. Langevin equation

Let us start with what is, perhaps, the master equation of all terrestrial beings: the time evolution of a body's acceleration is governed by the forces acting upon it,

$$m\partial_t \boldsymbol{v} = \sum_i \boldsymbol{f}_i$$

Let this body be a small particle immersed in a viscous fluid. The surrounding medium exerts a resistance proportional to its velocity — a simple friction term,

$$m\partial_t \boldsymbol{v} = -\gamma \boldsymbol{v}$$

where γ is the viscous drag coefficient. This viscous coefficient is given by the Stokes' law, in the form $\gamma = 6\pi\eta R$ where η is the dynamic viscosity of the fluid and R the typical dimension of our particle. The solution is straightforwardly,

$$\boldsymbol{v}(t) = \boldsymbol{v}_0 e^{-\gamma t/m}$$

The velocity decays exponentially and the motion dies away. The particle eventually comes to rest, yet under the microscope, that is not what we see. The motion never ceases, it persists. There must be another force at play. Langevin proposed to introduce such a random force ξ to the equations of motion, leading to

$$m\partial_t \boldsymbol{v} = -\gamma \boldsymbol{v} + \boldsymbol{\xi}(t) \tag{1}$$

with the requirement that

$$\langle \boldsymbol{\xi}(t) \rangle = 0 \tag{2}$$

since the particle, though constantly shaken, achieves no net motion over time. But what is the strength of this force? Mathematically, since the average of $\boldsymbol{\xi}$ is zero, we must instead look at its variance. This can be understood in the light of the fluctuation-dissipation theorem, which states that the amplitude of the random fluctuations is directly tied to the strength of the dissipative term, such that,

$$\langle \xi_\alpha(t)\xi_\beta(t') \rangle = 2\gamma k_B T \cdot \delta_{\alpha\beta} \cdot \delta(t - t') \tag{3}$$

where Greek letters denote coordinates of $\boldsymbol{\xi}$ and where k_B is the Boltzmann constant and T the temperature. With these properties, $\boldsymbol{\xi}$ is known as a Gaussian white noise. These considerations are modern[1] — Langevin himself had no knowledge of them. For instance, the appearance of the $\delta(t - t')$ implicitly enforces what we now call the Markov property: the random force acting on the particles at a given instant t is independent of the forces that acted in the past, and likewise uncorrelated with those that will act in the future. Moreover, the mathematical nature of this noise, while perfectly acceptable from a physicist's point of view, deserves a closer look. We will return to this question when discussing the numerical resolution of the Langevin equation.

2.2. Rouse model

To move on to the Rouse model, we must now consider a collection of N identical particles, called beads, whose dynamics each follow the Equation 1 of Langevin. This time, however, we introduce an additional coupling term — a harmonic interaction U that links neighboring particles together. Thus, for the ith particle, the equation of motion takes the general form,

$$m\partial_t \boldsymbol{v}_i = -\boldsymbol{\nabla}_{\boldsymbol{x}_i} U - \gamma \boldsymbol{v}_i + \boldsymbol{\xi}_i$$

[1] I would like to apologize for the lack of derivation here. I am familiar with the derivation of Equation 3 using the integrating factor — I simply chose not to include it. None of the demonstrations I have come across seem to *physically* justify the path they take, and perhaps out of a lack of technical skill on my part, it feels to me more like a matter of mathematics than of physical intuition. To the reader, sorry, again.

In practice, the bead mass m is tiny (typically $m \sim 10^{-25}$ kg) compared with the viscous scale set by γ. The momentum-relaxation time $\tau_m = m/\gamma$ is therefore extremely short, much smaller than the spring and diffusive times that governs the dynamics. Since the thermal noise amplitude scales as $\sqrt{2\gamma k_B T}$ (fluctuation-dissipation), and the harmonic coupling captures bond forces, the inertial term $m_i \partial_t \boldsymbol{v}_i$ can be safely neglected and the bead enters the overdamped regime,

$$\begin{aligned} \gamma \boldsymbol{v}_i &= -\boldsymbol{\nabla}_{\boldsymbol{x}_i} U + \boldsymbol{\xi}_i \\ \Leftrightarrow \gamma \partial_t \boldsymbol{x}_i &= -\boldsymbol{\nabla}_{\boldsymbol{x}_i} U + \boldsymbol{\xi}_i \end{aligned}$$

once rewritten using position instead of velocity. Then, for a chain of N beads (indices $i = 0, ..., N-1$) with the harmonic potential,

$$U = \frac{k}{2} \sum_{i=0}^{N-2} \| x_{i+1} - x_i \|^2$$

where k is the spring constant, it follows that the system

$$\begin{aligned} \gamma \partial_t \boldsymbol{x}_0 &= k(x_1 - x_0) + \boldsymbol{\xi}_0 \\ \gamma \partial_t \boldsymbol{x}_i &= k(x_{i+1} - 2x_i + x_{i-1}) + \boldsymbol{\xi}_i \\ \gamma \partial_t \boldsymbol{x}_{N-1} &= k(x_{N-2} - x_{N-1}) + \boldsymbol{\xi}_{N-1} \\ \langle \boldsymbol{\xi}_i \rangle &= 0 \\ \langle \xi_{i_\alpha}(t) \xi_{j_\beta}(t') \rangle &= 2\gamma k_B T \cdot \delta_{ij} \cdot \delta_{\alpha\beta} \cdot \delta(t - t') \end{aligned} \tag{4}$$

forms our complete set of equations to solve for the Rouse model in the overdamped limit. To solve this coupled system, Rouse proposed to introduce a set of orthogonal coordinates — the Rouse modes — that diagonalize the harmonic coupling between beads. For sake of clarity, we will follow the derivation proposed in [6]. These solutions were of the form,

$$\boldsymbol{X}_p(t) = \frac{1}{N} \sum_{i=0}^{N-1} \boldsymbol{x}_i(t) \cos\left[\frac{p\pi}{N} \left(i + \frac{1}{2} \right) \right] \tag{5}$$

Each of these modes describes a collective motion of the chain, analogous to the normal modes of a vibrating string, damped and driven by noise. The lowest modes (small p) govern the large-scale motions of the chain, while higher modes describe faster, local fluctuations. The mode 0 corresponds to the diffusion of the center of mass,

$$\boldsymbol{X}_0(t) = \frac{1}{N} \sum_{i=0}^{N-1} \boldsymbol{x}_i(t)$$

We can now reverse Equation 5, giving

$$\boldsymbol{x}_i(t) = \boldsymbol{X}_0(t) + 2 \sum_{p=1}^{N-1} \boldsymbol{X}_p(t) \cos\left[\frac{p\pi}{N} \left(i + \frac{1}{2} \right) \right] \tag{6}$$

which can be injected into Equation 4. We want to emphasize that, by doing so, the equations now decouple and we are left with something that goes as,

$$\gamma \partial_t \boldsymbol{X}_p = -k_p \boldsymbol{X}_p + \boldsymbol{\Xi}_p \tag{7}$$

where $k_p = 4k \sin^2[p\pi/2N]$ represents an effective spring constant for the mode p and $\boldsymbol{\Xi}$ the corresponding white noise. The formal solution can be written as

$$\boldsymbol{X}_p(t) = \boldsymbol{X}_p(0) e^{-t/\tau_P} + \mathcal{F}\left[\boldsymbol{\Xi}_p\right]$$

with $\tau_p = \gamma / k_p$ being the characteristic relaxation time of mode p and $\mathcal{F}$ a functional of $\boldsymbol{\Xi}_p$ representing the fluctuations. The Rouse modes do not oscillate; their amplitude decay exponentially, independently, with a finite lifetime. We will attempt to measure these later on.

2.3. Observables

Before turning to simulations, let us briefly recall which quantities can be observed. Most of the observables discussed below are not specific to the Rouse model. They are rather standard quantities used to characterize the structure and dynamics of polymer chains — both in theory and simulation. Most of the quantities of interest are derived from [7].

The story begins with a simple observation: if the chain vibrates, stirs, and folds, its center of mass must itself diffuse within the volume. But how much does it move, and how does it move? The answer lies in Section 2.2 where we mentioned that the mode $p = 0$ corresponds to the diffusion of the center of mass of the chain. Then, according to Equation 7,

$$\partial_t \boldsymbol{X}_0 = \frac{1}{\gamma} \boldsymbol{\Xi}_0 \quad \text{with} \quad \boldsymbol{\Xi}_0 = \frac{1}{N} \sum_{i=0}^{N-1} \boldsymbol{\xi}_i$$

Since the individual random forces $\boldsymbol{\xi}_i$ are uncorrelated, their sum produces an effective noise of reduced strength,

$$\left\langle \Xi_{0_\alpha}(t) \Xi_{0_\beta}(t') \right\rangle = \frac{2\gamma k_B T}{N} \cdot \delta_{\alpha\beta} \cdot \delta(t - t')$$

The formal solution is,

$$\boldsymbol{X}_0(t) = \boldsymbol{X}_0(0) + \frac{1}{\gamma} \int_0^t \boldsymbol{\Xi}_0(s) \, \mathrm{d}s$$

We can derive a quantitative feature from there, in the name of mean-squared displacement (*abbrev*. MSD), by rewriting the previous relation and square it on both sides,

$$\left(\boldsymbol{X}_\mathbf{0}(t) - \boldsymbol{X}_0(0)\right)^2 = \frac{1}{\gamma^2} \int_0^t \int_0^t \boldsymbol{\Xi}_0(s) \boldsymbol{\Xi}_0(s') \, \mathrm{d}s \, \mathrm{d}s'$$

Averaging quantities out,

$$\begin{aligned} \left\langle \left(\boldsymbol{X}_\mathbf{0}(t) - \boldsymbol{X}_0(0)\right)^2 \right\rangle &= \frac{1}{\gamma^2} \int_0^t \int_0^t \langle \boldsymbol{\Xi}_0(s) \boldsymbol{\Xi}_0(s) \rangle \, \mathrm{d}s \, \mathrm{d}s' \\ &= \frac{1}{\gamma^2} \int_0^t \int_0^t \frac{6\gamma k_B T}{N} \cdot \delta(s - s') \, \mathrm{d}s \, \mathrm{d}s' \end{aligned}$$

which simplifies into

$$\left\langle (\boldsymbol{X_0}(t) - \boldsymbol{X}_0(0))^2 \right\rangle = 6\frac{k_B T}{\gamma N} t \tag{8}$$

This linear dependence of the mean-squared displacement on time is the signature of Brownian motion. By comparison with the Einstein relation $\langle (\Delta R)^2 \rangle = 6Dt$ we can identify the diffusion coefficient of the center of mass as $D = k_B T / \gamma N$. It follows that the center of mass of the polymer diffuses N times slower than for a single bead, as if the entire chain behaved as a single particle with an effective friction γN.

Regarding the individual beads, the situation is more complex. Because each bead is coupled to its neighbours through the harmonic potential, its motion cannot be derived as straightforwardly as for the center of mass. In the literature, two dynamical regimes are usually distinguished when analyzing the MSD: in addition to the diffusive regime — governed by the interactions with the solvent, and scaling as t^1 — there is the subdiffusive regime that is mainly constrained by the harmonic potential, scaling as $t^{1/2}$.

Between these two limits, one may expect intermediate regimes where both harmonic constraints and thermal fluctuations act on comparable timescales. Moreover, at very short times, a ballistic regime may be observed: before the beads have thermalized and start to feel the effects of the coupling potential or the solvent, their motion is nearly free, resulting in an MSD scaling as t^2. This inertial regime is generally too short-lived to be captured in overdamped simulations.

After discussing global motions, we can now turn to the linear structure of the system. One of the first quantities of interest is the linear dimension of the chain. At equilibrium, the total length ℓ can be roughly estimated as $\ell = Nb$ where b is the typical bond length (or spring equilibrium length) between two consecutive beads.

However, this estimate is not particularly informative, since the chain tends to fold and coil like a ball of yarn — making it difficult to relate ℓ to the actual space occupied by the chain. A more meaningful is the end-to-end distance, R_e^2, defined as the quadratic distance between the two ends of the chain. As shown in [7], its mean value for a Rouse-like chain scales as,

$$\langle \boldsymbol{R}_e^2 \rangle = Nb^2 \tag{9}$$

Another closely related measure is the radius of gyration R_g^2 which represents the squared distance between the beads and the center of mass. Again, for a Rouse-like chain, these two quantities are related through,

$$\langle \boldsymbol{R}_e^2 \rangle = 6\langle R_g^2 \rangle$$

Together, R_e and R_g provides complementary information: the first characterizes the extension of the chain, while the second captures how the mass is distributed within the space. The contribution of modes to R_e can be computed,

$$\boldsymbol{R}_e = \boldsymbol{x}_{N-1} - \boldsymbol{x}_0 = 2\sum_{p=1}^{N-1} \boldsymbol{X}_p[(-1)^p - 1]\cos\Big(\frac{p\pi}{2N}\Big)$$

The observation is 2–fold. First, the amplitude of $\boldsymbol{R}_e$ depends on how the different modes $\boldsymbol{X}_p$ are populated. Second, because each mode $\boldsymbol{X}_p$ relaxes exponentially with its own characteristic time τ_p, the end-to-end distance is also expected to decay in time, reflecting the chain's tendancy to retract and fold as internal tensions are relaxed.

Consequently, the population of the Rouse modes becomes a key quantity to investigate. A natural way to characterize it is to look at how each mode contributes to the energy of the chain. As discussed in the previous section, each mode behaves as an independent harmonic oscillator with an effective spring constant k_p. Therefore, applying the equipartition theorem yields a relationship between the mode index p and its average amplitude, since,

$$\begin{aligned} \frac{1}{2}k_p\left\langle|\boldsymbol{X}_p|^2\right\rangle &= \frac{3}{2}k_BT \\ \Rightarrow \left\langle|\boldsymbol{X}_p|^2\right\rangle &= 3\frac{k_B}{k_p}T \end{aligned}$$

Recall that $k_p = 4k\sin^2[p\pi/2N]$. When the chain is long enough, one can go a bit further for the slowest modes $p \ll N$, since $k_p \approx 4k(p\pi/2N)^2$. From there, it follows that,

$$\left\langle|\boldsymbol{X}_p|^2\right\rangle \propto p^{-2} \tag{10}$$

This inverse-square dependence implies that the slowest modes contribute the most to the overall conformational dynamics. My last words are to give a practical way to measure the relaxation times τ_p. Starting from the formal solution of Equation 7,

$$\boldsymbol{X}_p(t) = \boldsymbol{X}_p(0)e^{-t/\tau_P} + \mathcal{F}\left[\boldsymbol{\Xi}_p\right]$$

one can compute the time autocorrelation function,

$$\begin{aligned} \langle\boldsymbol{X}_p(t)\boldsymbol{X}_p(0)\rangle &= \langle|\boldsymbol{X}_p(0)|^2\rangle e^{-t/\tau_P} + \langle\left(\boldsymbol{X}_p(0) \times \mathcal{F}\left[\boldsymbol{\Xi}_p\right]\right)\rangle \\ &= \langle|\boldsymbol{X}_p(0)|^2\rangle e^{-t/\tau_P} + \mathcal{F}'\left[\langle\boldsymbol{X}_p(0)\boldsymbol{\Xi}_p\rangle\right] \end{aligned}$$

Physically, this cross-term vanishes because the random force $\boldsymbol{\Xi}_p$ represents new thermal kicks acting at time t, which have no memory — due to the Markov property aforementioned — of the configuration at time $t = 0$. The initial $\boldsymbol{X}_p(0)$ results from past noise realizations, while $\boldsymbol{\Xi}_p$ reflects independent, instantaneous fluctuations of the solvent. As a result, their correlation averages to zero. Finally,

$$\frac{\langle\boldsymbol{X}_p(t)\boldsymbol{X}_p(0)\rangle}{\langle|\boldsymbol{X}_p(0)|^2\rangle} = e^{-t/\tau_P} \tag{11}$$

This relation provides a direct way to extract the relaxation times τ_p from simulations, by fitting the decay of this autocorrelation function. This closes our theoretical overview of the Rouse model.

3. Implementation

The idea of Lisp was born with John McCarthy in 1958 [8], five years after the Rouse model, and it has survived ever since through many dialects — Emacs Lisp, Racket, Clojure and Common

Lisp among the best known today. At its core lay a remarkably elegant insight: that a program could be nothing more than a structured collection of symbols, and that the very syntax of computation could be represented as a list.

And once one accepts that a program is nothing but a list, an inevitable idea follows. McCarthy realized that one could write functions — *macros* — that take lists as input and return new lists as output. In other words, programs capable of writing programs. This reflexive capacity gave Lisp its extraordinary flexibility: the language could adapt to the problem, reshape itself to fit a domain, or even extend its own syntax. It is often summarized by the motto: *code is data*.

Beyond its syntax, Lisp introduced another quiet revolution: the idea of interaction. The READ-EVAL-PRINT-LOOP (REPL) turned programming from a static act of compilation into a genuine dialogue. One can feed expressions to the machine, observe their behavior, modify them and start again — all in real time. This immediate feedback transforms the way models are built. Everything emerges iteratively, as the programmer is feeding thoughts into the process.

In that sense, modern environments such as Jupyter notebooks or interactive shells in Python, Julia or JavaScripts are only distant echoes of this original idea. Likewise, the macros proposed in C++ or Rust are little more than preprocessor tricks. They lack reflection. They lack semantic coherence. In Lisp, macros are not text substitutions; they are meta-programs, capable of reshaping the language itself from within. Deprived of Lisp's unified syntax and symbolism, they borrow some of its features while remaining bound to the rigidity of C-like grammars, condemning themselves to be forever less expressive.

Here, I would like to give a glimpse of what a molecular dynamics tool written in Lisp could look like. Those who have used some of the popular solutions (GROMACS, LAMMPS, …) might well regret that things did not take such a path. One could perhaps mention OpenMM, which indeed is on the right track but, alas, has fallen in love with Python. Before giving a taste of it, however, we must return for a moment to the equations.

3.1. Euler-Maruyama

We must now address this white noise term, the random force. As mentioned earlier, for physicists this term feels natural; Langevin himself introduced it explicitly in his model. It has the dimension of a force and, one imagines it as representing, as each instant, the effect of random molecular kicks, with the statistical properties discussed in Section 2.1.

Yet, rigor demands that we understand this noise in a probabilistic sense. Mathematicians have formalized it through the notion of Wiener processes, which provide a convenient way (among others) to encode randomness in continuous time. Formally, one writes,

$$\begin{aligned} \boldsymbol{\xi} &= \sqrt{2\gamma k_B T} \cdot \frac{\mathrm{d}\boldsymbol{W}_t}{\mathrm{d}t} \\ \langle \boldsymbol{W}_t \rangle &= 0 \\ \langle W_{t_\alpha}^2 \rangle &= t \end{aligned} \tag{12}$$

From there, one can simply rewrites the Langevin equation in its differential form,

$$\mathrm{d}\boldsymbol{x} = -\frac{1}{\gamma}\boldsymbol{\nabla} U \,\mathrm{d}t + \frac{1}{\gamma}\boldsymbol{\xi}\,\mathrm{d}t$$

which, by virtue of Equation 12, can be stated as

$$\mathrm{d}\boldsymbol{x} = -\frac{1}{\gamma}\boldsymbol{\nabla} U \,\mathrm{d}t + \sqrt{\frac{2k_B T}{\gamma}}\,\mathrm{d}\boldsymbol{W}_t$$

The term $\mathrm{d}\boldsymbol{W}_t$ expresses a small increment in the Wiener process. To implement this Wiener process numerically, one can simply draw a random number η_α from a standard normal distribution $\mathcal{N}(0,1)$, and write that the Wiener increment is given by $\mathrm{d}W_{t_\alpha} = a\eta_\alpha$ with a a proportionality constant. We can check that this construction satisfies the expected statistical properties.

For the mean, the result is immediate, since $\langle \eta_\alpha \rangle = 0$, hence $\langle \, \mathrm{d}W_{t_\alpha} \rangle = 0$. For the mean-squared displacement, one has,

$$\langle \, \mathrm{d}W_{t_\alpha}^2 \rangle = a^2 \langle \eta_\alpha^2 \rangle = a^2$$

Comparing with the theoretical definition immediately lands $a = \sqrt{\mathrm{d}t}$. Thus, in practice, the discrete form of the Wiener process reads,

$$\Delta W_{t_\alpha} = \eta_\alpha \sqrt{\Delta t}, \quad \eta_\alpha \sim \mathcal{N}(0,1) \tag{13}$$

By discretizing the Langevin equation,

$$\Delta \boldsymbol{x} \approx \boldsymbol{x}(t+\Delta t) - \boldsymbol{x}(t) = -\frac{1}{\gamma}\boldsymbol{\nabla} U \Delta t + \sqrt{\frac{2k_B T}{\gamma}} \cdot \boldsymbol{\eta} \cdot \sqrt{\Delta t}$$

we obtain the **Euler-Maruyama** scheme,

$$\boldsymbol{x}(t+\Delta t) = \boldsymbol{x}(t) - \frac{1}{\gamma}\boldsymbol{\nabla} U \Delta t + \sqrt{\frac{2k_B T}{\gamma}} \cdot \boldsymbol{\eta} \cdot \sqrt{\Delta t} + O(\Delta t) \tag{14}$$

3.2. Common Lisp & Architecture

When one starts implementing a molecular dynamics engine, the first practical question is not only the algorithm, but the environment in which it will live. Most existing tools provide remarkable numerical engines, yet they remain monolithic or even opaque. They offer efficiency, but they lack intimacy. They are computationally powerful but rigid. Launching a simulation can feel almost as exhausting as doing labwork.

But to me, a molecular dynamics engine should be seen as a tool for *in silico* experimentation. Let us recall what we truly do: we integrate equations derived from models, and we test what our assumptions imply. What we need, then, are tools suited to the pace of theoretical reasoning, *i.e.* to the act of erasing, rewriting and simplifying as one does on a blackboard.

This project was born precisely from that absence: a lack of flexibility, of agility, of an environment that feels alive. Common Lisp, with its symbolic nature and living runtime, offered exactly

that. It allows a style of development that is both exploratory and precise, where one can express physical ideas directly in code and see them evolve interactively.

From this starting point, the implementation was designed to remain modular and transparent. Each main component could be tested or replaced independently. The codebase is divided into six main packages: topology, control, test, viewer, simulation & scripts. Each component is relatively independent and could be rewritten or extended without difficulty — a design choice meant to favor modularity and long-term maintainability. Let us briefly walk through some of them,

- The **topology** module is in charge of building the chain. It relies on two simple classes: `chain`, which contains a list of beads, and `bead` which holds the physical information of each Brownian particle — its position, mass and possibly velocity. Around these structures live a few methods to access positions, extract properties or compute derived quantities (center of mass, radius of gyration, …). But perhaps more importantly, a small set of macros lets us write concise and expressive topology definitions. For instance, the following expression,

  ```
  (top:make-linear-chain 20 :along :y :spaced-by 1d-9)
  ```

  will generate a chain of 20 beads, aligned along the y–axis, with a spacing of 1nm between each bead at time zero.

- On the other side, the **simulation** module contains the core simulation logic. It defines what a simulation `state` is — a combination of physical parameters ($\gamma, T, \Delta t, \ldots$) and a polymer `chain` at a given time. A simulation is then represented as a class holding a `timeline`, *i.e.* a list of states stored in a LIFO structure, together with a `cursor` pointing to the current state.

  This design is quite distinctive: it allows one to move `backward` or `forward` in time at will, using dedicated methods, to modify either the physical parameters or the chain on the fly, and to recompute trajectories instantly from any previous state. Running replicas at different temperatures is therefore as simple as,

  ```
  (defvar *sim*
    (sim:make-simulation
      :chain (top:make-linear-chain 20 :along :y :spaced-by 1d-9)
      :temperature 273.15 :gamma 1d-11 :k 1.2d-2 :dt 1d-12))

  (sim:propagate *sim* :steps 1000) ;; thermal equilibrium reached after 1ns

  (loop for temperature in '(273.15 280 290 300)
        do (let ((current-state (sim:current-state)))
             ;; changing the temperature
             (setf (sim:state-temperature current-state) temperature)
             ;; computing new solutions
             (sim:propagate *sim* :steps 10000 :save-every 100)
             ;; exporting solutions to a file
             (sim:export-simulation *sim* (format nil "~a" temperature))
             (sim:backward *sim* :steps 1000)))
  ```

  I challenge the reader to do it with such ease in GROMACS.

- Finally, the **viewer** module follows the same philosophy, allowing one to visualize the simulation as it is being shaped by thought. Its implementation remains minimal. Due to time constraints, only a single view has been developed so far: the so-called `ortho-view`, illustrated in Figure 1. This view displays the motion of the chain projected along the cartesian axes $(\pm X, \pm Y, \pm Z)$, while removing the diffusion of the center of mass so that one can focus purely on the internal dynamics. It also provides a few convient interactions; for instance, the $\leftarrow$ and $\rightarrow$ keys allow one to navigate backward and forward in time through the timeline. To launch it,

```
(view:view *sim* :mode :ortho-view)
```

Most of these functions have been verified through integration tests, but more importantly through unit testing. While the code coverage is not exhaustive, the tests target the most critical components of the system and ensure that each module behaves as intended.

A final note about Common Lisp: although born in the 1980s and only slightly updated since, its ecosystem for numerical computation remains surprisingly limited. For instance, implementing the Euler-Maruyama scheme requires drawing samples from a normal distribution, yet Common Lisp — to the best of my knowledge — provides no native generator for $\mathcal{N}(0, 1)$. I therefore reimplemented the Box-Muller transform myself.

Figure 1 — Visual feedback during simulation, as displayed by the `ortho-view` mode.

Finally, the **scripts** package contains a set of Python scripts used for figure generation and physical analysis, bridging the Lisp simulation data with more conventional visualization tools.

3.3. Physical validation

We would not want to spoil the reader too early about the behavior of the Rouse modes! For this reason — and because our integrator conveniently allows it — we shall start in a regime that is

actually poorly captured by the Rouse model: that of a short chain. We will focus mainly on its diffusive properties, which provide a simple yet effective way to test the physical consistency of our simulation engine, and some structural behavior.

We will work at constant N, T and set the temperature to $T = 273.15$K. The chain length is kept short — $N = 20$ — with a bead mass of order $m \sim 10^{-25}$. The viscous coefficient is estimated from Stokes' law,

$$\gamma = 6\pi\eta^\star R$$

with $\eta^\star$ the viscosity of water and R a typical bead radius. At this temperature and for a typical radius $R \sim 10$Å, γ is of order 10^{-11} kg/s. Since bonds are harmonic, equipartition sets the spring constant through

$$\frac{1}{2}kb^2 = \frac{1}{2}k_BT \quad \Rightarrow \quad k = \frac{k_BT}{b^2}$$

With $b = 0.56$ nm, this gives $k \sim 1.2 \times 10^{-2}$ N/m. The momentum-relaxation time is thus $\tau_m = m/\gamma = 10^{-14}$s. With a timestep $\Delta t = 0.1$ ps, we are thus safely in the overdamped regime. It is with these parameters that we launch the following code for 150 ns,

```
(in-package #:rouse)

(defvar *sim-test*
  (sim:make-simulation
   :chain (top:make-linear-chain 20 :along :y :spaced-by 1d-9 :mass 1d-25)
   :temperature 273.15 :gamma 1d-11 :k 1.2d-2 :dt 1d-13))

(sim:propagate *sim-test* :steps 1500000 :save-every 1500)
(sim:export-simulation *sim-test* "150ns")
```

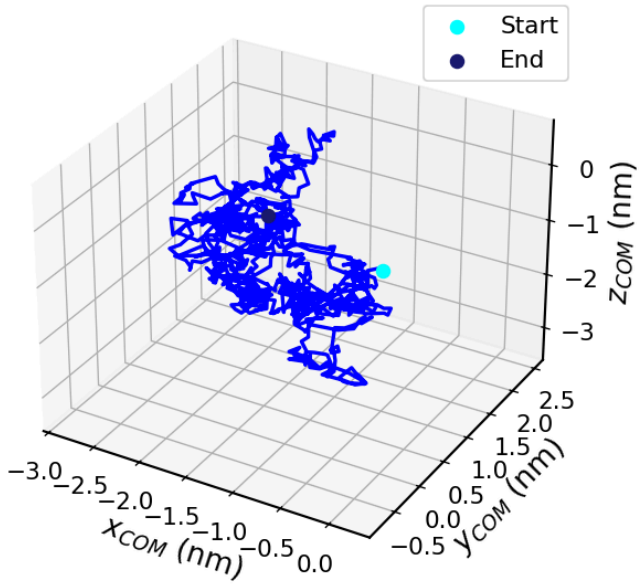

Figure 2 — Trajectory of the polymer center of mass over 150 ns.
The motion is purely diffusive, as expected for a Brownian particle in the overdamped regime.

Focusing now on the physics of the center of mass, we first note that the COM exhibits a clear Brownian motion, diffusing freely in space as shown in Figure 2. To quantify this behavior, we compute its mean-squared displacement and fit the linear region according to Equation 8. The result is shown in Figure 3 against time lags, and the corresponding diffusion constants are,

$$D_{\text{com}}^{\text{fit}} = 1.71 \times 10^{-12}\ \text{m}^2/\text{s} \quad ; \quad D_{\text{com}}^{\text{th}} = 1.88 \times 10^{-12}\ \text{m}^2/\text{s}$$

The integrator conserves the expected diffusive scaling law, within 10% accuracy, which is consistent regarding the short duration of the simulation. At large time lags, the MSD becomes noisier due to the reduced number of samples available for averaging, a purely statistical effect that does not affect the physical interpretation.

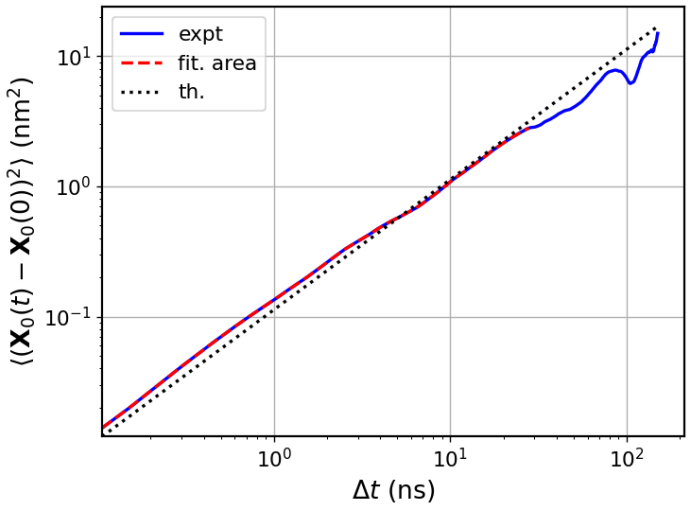

Figure 3 — Mean-squared displacement of the center of mass as function of time lags. The linear dependence confirms diffusive behavior. The fitted slope provides the diffusion coefficient $D_{\text{com}}^{\text{fit}}$.

If we now focus on the diffusion of a single bead, we should be able to observe the two characteristic regimes discussed in Section 2.3: a subdiffusive regime and a diffusive one. Figure 4 (left) shows the MSD of bead 3, and Figure 4 (right) extends this to several beads along the chain to remove any ambiguity that the behavior might be specific to one position.

What we observe is consistent with the literature: at short times, a brief diffusive regime — possibly the traces of a ballistic phase — is followed by a subdiffusive regime where the bead is transiently constrained by its neighbors. At longer times, the dynamics cross over back to diffusion as the bead ultimately follows the global motion of the chain. Figure 4 (right) further shows that beads located near the center of the chain (*e.g.*, the 8th) exhibit slightly smaller amplitude in the long run than those at the ends, which is likely a signature of Rouse-like dynamics — a point that will be further explored in the next section.

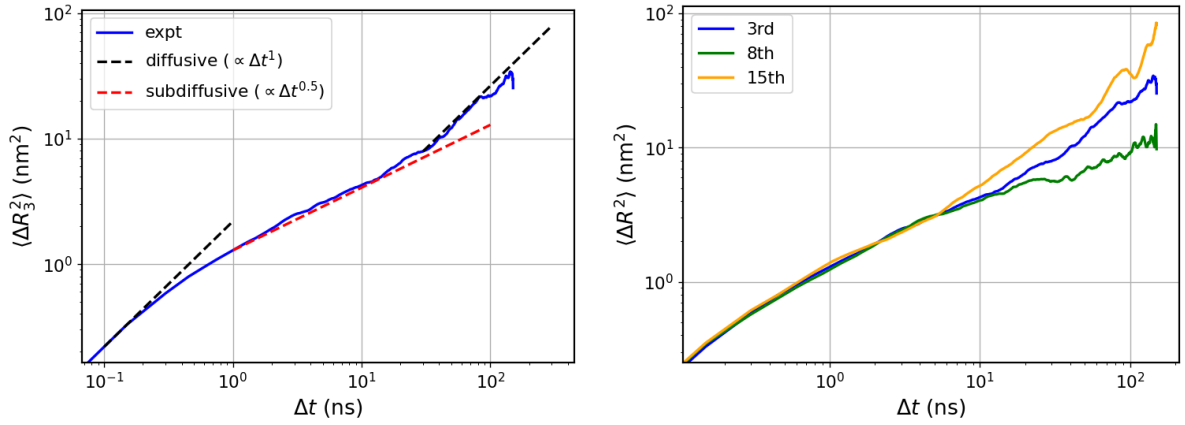

Figure 4 — (**Left**) Mean-squared displacement of the bead 3. At short times, a brief diffusive regime is followed by a subdiffusive one, before recovering diffusion at long times. The dashed lines indicate the expected scaling laws. (**Right**) Mean-squared displacement of some beads (3rd, 8th and 15th) as a function of time lags. All beads exhibit a short-time diffusive regime followed by a subdiffusive one, with end beads showing slightly larger amplitudes due to reduced connectivity.

To isolate discretization effects, we reran the simulations with the same random seed but a time step ten times larger. The MSD of the center of mass and of single beads retained the same type of slopes and fitted diffusion constant remained within statistical uncertainty. This indicates that, in our parameter range, the implementation of the Euler-Maruyama scheme is not introducing a systematic bias, provided that the chosen Δt remains consistent with overdamped timescales.

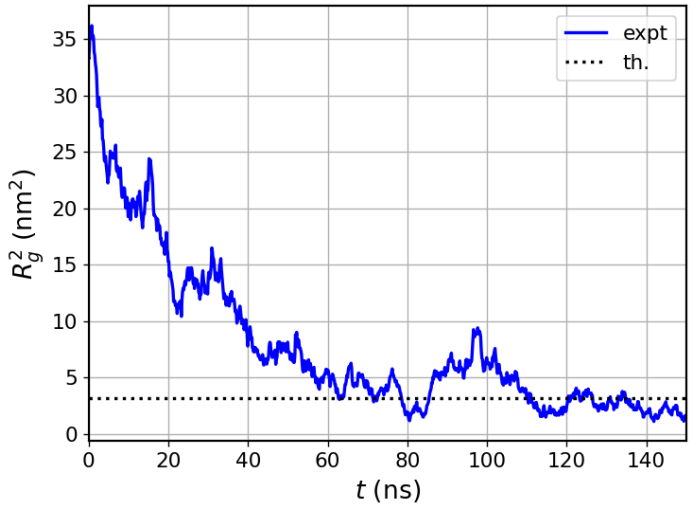

Figure 5 — The radius of gyration R_g^2 (blue) progressively relaxes toward the theoretical ideal-chain value $Nb^2/6$ (dotted line). The convergence confirms that the chain has reached its equilibrium conformational size.

Finally, we verified that the structural properties of the chain were also correctly reproduced. Figure 5 shows the time evolution of the radius of gyration R_g^2. Starting from a stretched initial configuration, the chain rapidly collapses and equilibrates around the theoretical value

$Nb^2/6$, as expected. Taken together with the diffusive results, this agreement indicates that our simulation engine may well capture both the dynamical and structural aspects of the physics.

4. Results & Physical discussion

Perhaps, at first glance, the reader might find this model somewhat toy-like. Yet, it is worth insisting on what it truly represents. As mentioned earlier, the Rouse model is nothing more than the coupling of Brownian particles, which, on paper, translate into the study of the evolution of a linear structure freely moving in a solvent.

Such structures are ubiquitous. They form the very basis of polymer physics — and, more importantly, of biological polymers. They are your proteins. They are your membrane receptors for drugs. They are your DNA and RNA. Taken together, they assemble into larger complexes — your enzymes. What the Rouse model ultimately proposes it is not merely a toy picture of coupled beads, but rather an effective coarse-graining of the flexible chains that make up life itself.

This kind of coarse-graining is standard in molecular simulations. As the theoretical part (Section 2.2) showed, these objects are governed by slow, collective timescales — often far beyond the reach of atomistic simulations. On a typical mesocenter, one may reach hundreds of nanoseconds, or the microsecond scale for the most ambitious runs, in all-atoms. Yet these remain insufficient to capture the long-time dynamics of chains. Hence, one relies on such simplified models — not as a compromise, but as a bridge toward the physical essence of the problem.

In this perspective, the Rouse model can in fact reveal something quite interesting. The whole construction — from the equations to the simulation — should be understood under this enlightenment. To emphasize this, we now place ourselves in a setup closer to a biological environment. We set the temperature to $T = 310\text{K}$, corresponding to the human body temperature, and consider a longer chain with $N = 80$ beads. Following the same procedure as in the previous section, we derive the physical parameters as,

$$\gamma = 10^{-11}\ \text{kg/s}, \quad m = 10^{-25}\ \text{kg},$$
$$k = 1.2 \times 10^{-2}\ \text{N/m}, \quad \Delta t = 1\ \text{ps}$$

The chain spacing is set to $b = 0.56$ nm. We then propagate the system using the Euler-Maruyama integrator for a total simulated time of 20 µs.

4.1. Diffusion, Structure

We start by checking again the global dynamics of the chain by looking at its COM and a few structural observables. The trajectory of the COM, shown in Figure 6, exhibits the same Brownian character as previously observed, confirming that the system remains well in the overdamped regime.

To quantify this behavior, we again computed the MSD of the COM, displayed in Figure 7. Thanks to the longer simulation time and finer sampling, the linear diffusive regime is now clearly identified, allowing for a more reliable fit of the slope. From this fit, we obtain,

$$D_{\text{com}}^{\text{fit}} = 5.70 \times 10^{-12}\ \text{m}^2/\text{s} \quad ; \quad D_{\text{com}}^{\text{th}} = 5.35 \times 10^{-12}\ \text{m}^2/\text{s}$$

The agreement, within roughly 6.5%, confirms once again that the simulation engine correctly reproduces the expected diffusive dynamics, even for longer chains and timescales.

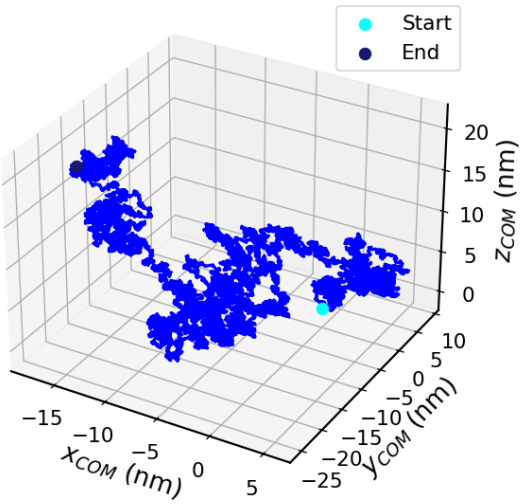

Figure 6 — Trajectory of the polymer center of mass over 20 µs.
The motion is purely Brownian, consistent with the overdamped Langevin regime and confirming the correct implementation of the integrator.

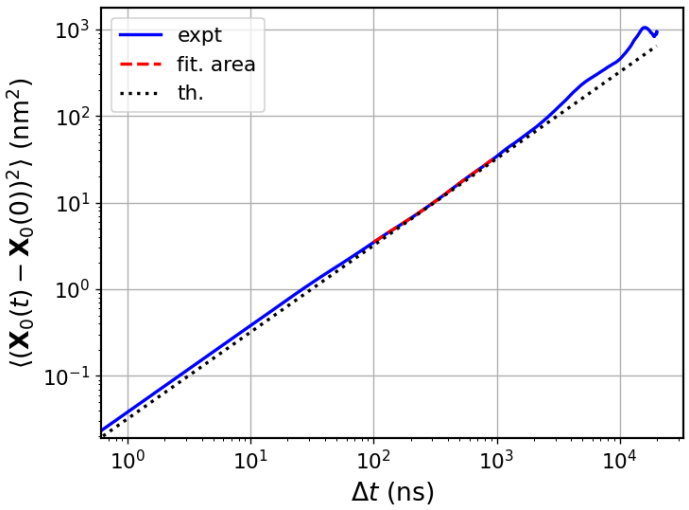

Figure 7 — Mean-squared displacement of the center of mass as a function of time lags.
The MSD grows linearly with time, characteristic of diffusive motion. The fitted slope gives a diffusion coefficient $D_{\text{com}}^{\text{fit}} = 5.70 \times 10^{-12}$ m^2/s, in excellent agreement (6.5 %) with the theoretical value $D_{\text{com}}^{\text{th}} = 5.35 \times 10^{-12}$ m^2/s.

In Figure 8, we report the time evolution of the end-to-end distance and the radius of gyration, respectively. Both quantities show a relaxation from the stretched initial condition and fluctuate around their expected theoretical values. Averaging over the trajectory gives,

$$\langle R_g^2 \rangle = 1.62 \times 10^{-17}\ \text{m}^2 \quad ; \quad \langle R_g^2 \rangle_{\text{th}} = 1.43 \times 10^{-17}\ \text{m}^2$$

$$\langle R_e^2 \rangle = 9.36 \times 10^{-17}\ \text{m}^2 \quad ; \quad \langle R_e^2 \rangle_{\text{th}} = 8.56 \times 10^{-17}\ \text{m}^2$$

These values agree within $\sim 7\%$, confirming that the equilibrium structure is correctly captured. The ratio R_e^2/R_g^2, shown in Figure 9, oscillates around 6, as theoretically predicted.

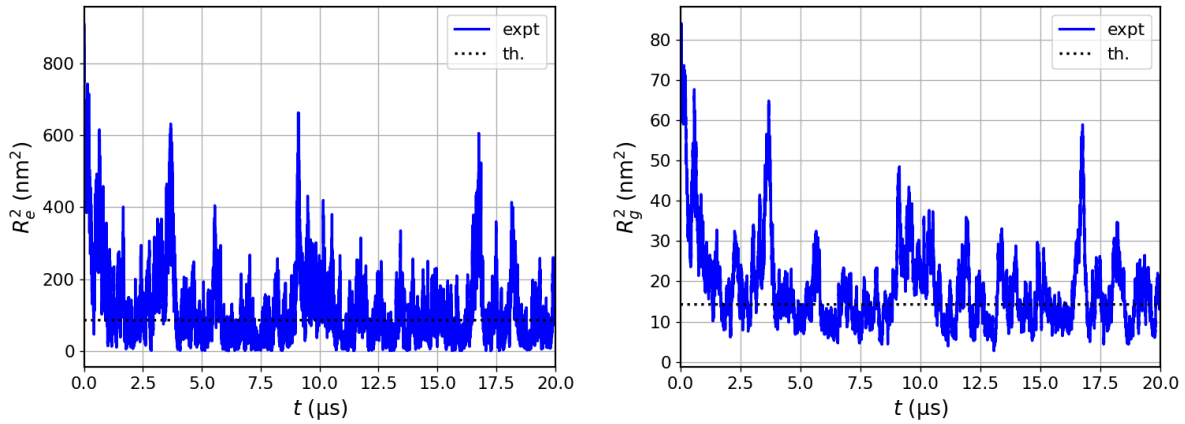

Figure 8 — Time evolution of the (**Left**) end-to-end distance (**Right**) radius of gyration. Starting from an initially stretched configuration, the chain quickly relaxes and fluctuates around its theoretical equilibrium value.
Both observables display comparable dynamical fluctuations, though end-to-end distance exhibits stronger variations due to the larger influence of the end beads.

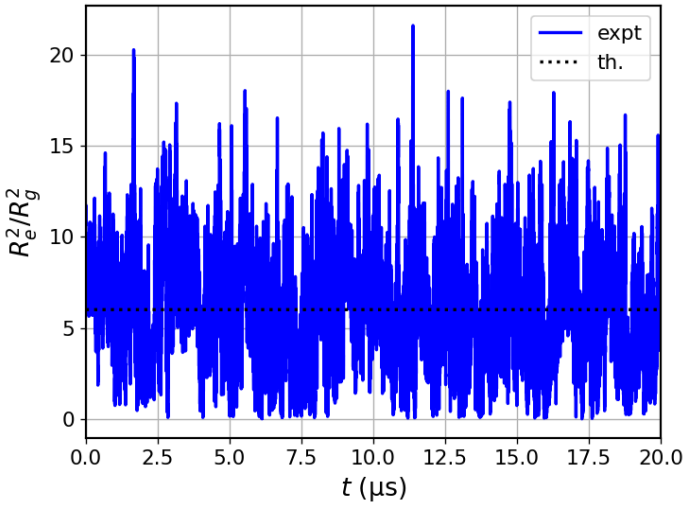

Figure 9 — The ratio R_e^2/R_g^2 as a function of time.

4.2. Rouse dynamics

Now comes the time to watch the theoretical derivations of the Rouse model come to life. A first remark: we should not expect to capture the fastest modes, as our time step may be limiting, and more importantly, the simulation is not recorded at every frame — otherwise, trajectories would quickly fill our disks. But what do Rouse modes actually look like in practice ? Figure 10 shows a few of them projected on the x–coordinate. The mode $p = 0$ (blue) corresponds to the diffusive motion of the center of mass, while higher modes represent internal fluctuations of increasing frequency and decreasing amplitude.

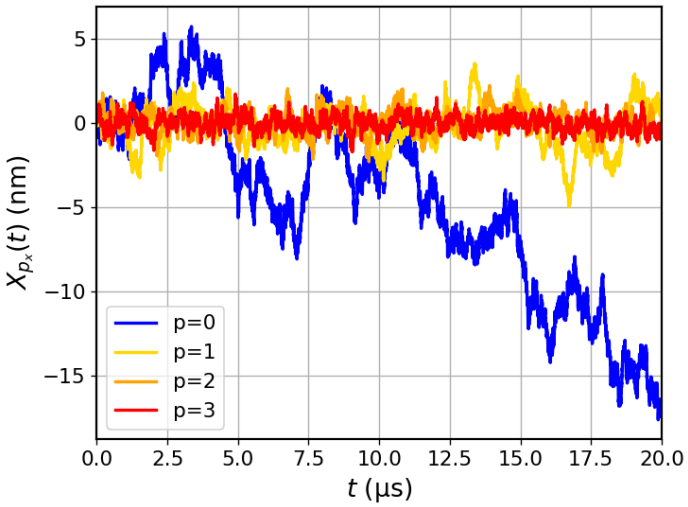

Figure 10 — Time evolution of the first Rouse modes along the x-axis. The slowest modes display large-scale motion of the chain, while higher-order modes correspond to faster, small-scale internal fluctuations that average out over time.

To measure the relaxation times of these modes, we follow the procedure outlined in Section 2.3, by computing the autocorrelation function of each mode after projecting the trajectories onto the corresponding Rouse basis. Figure 11 on left shows several of these normalized autocorrelation functions, while on right, the figure presents a typical exponential fit — here for mode $p = 1$. We chose to plot these correlations only over the first 2 µs, since — as shown in Figure 11 and in Figure 12 — the higher the p, the faster it decays. Given that the mode $p = 1$ relaxes on a timescale of about 0.6 µs, the signal beyond that is just noise.

Physically, these results show that different modes contribute to the chain dynamics on distinct spatial and temporal scales. Low–p modes represent large-scale, coherent motions of the whole chain and therefore relax slowly. Higher–p modes correspond to short-wavelength internal fluctuations, involving only neighboring segments, which decay much faster. The sharp decrease in relaxation times with increasing p thus reflects a sort of hierarchy — a hierarchy that is mirrored in the distribution of mode amplitudes. In fact, the mean population of each mode is expected to decrease as p^{-2}, reflecting the progressively smaller contribution of higher-order modes to the overall conformational motion.

We compute these mode populations from the simulation trajectories and compared them with the theoretical scaling (see Figure 13). While the overall decay follows the expected trend, we observe a clear deviation at large p. We believe this deviation mainly reflects finite-size effects, as well as the limited temporal resolution and frame sampling of our simulation — put differently, high–p modes fluctuate too fast to be fully captured with our parameters.

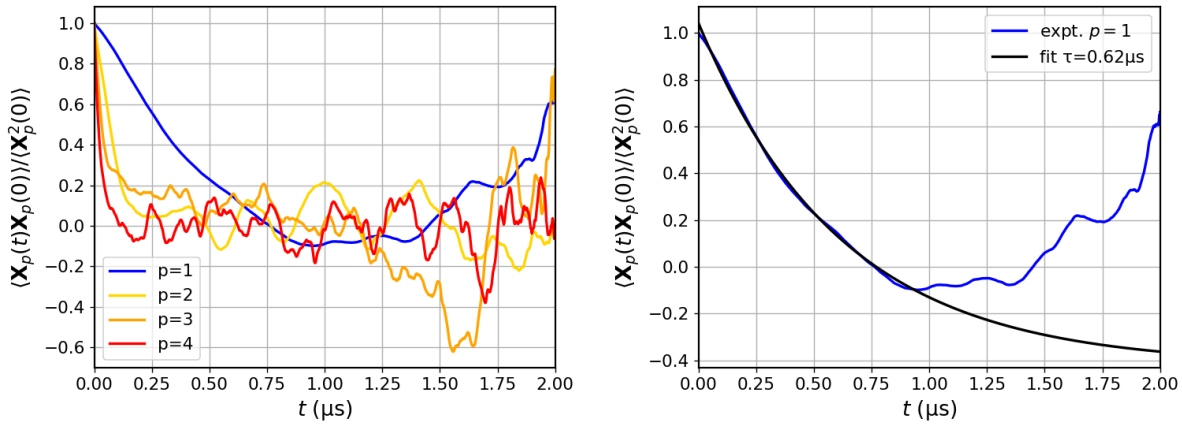

Figure 11 — (**Left**) Normalized autocorrelation functions of the first Rouse modes ($p = 1$–4). Higher-order modes decay faster.
(**Right**) Normalized autocorrelation function of the Rouse mode $p = 1$. The exponential decay (black line) provides the relaxation time $\tau_1 = 0.62$ μs.

Beyond these numerical details, it is worth taking a step back to think about what these mode populations — and their dynamics — actually mean. At first glance, it is clearly the slow modes that dominate the average behavior of the chain: they hold both the largest amplitudes and the longest relaxation times, so they carry most of the energy and the collective motion. The fast modes, by contrast, decay quickly and contribute only weakly to global averages — they are almost higher-order corrections in that sense. Still, it feels wrong to simply dismiss them.

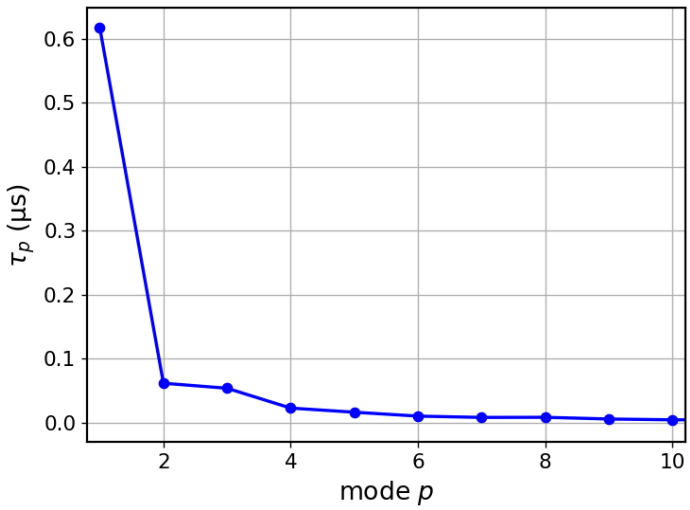

Figure 12 — Relaxation times τ_p of Rouse modes (1-10) as a function of mode index p. The relaxation time decreases rapidly with p, indicating that higher modes relax much faster and contribute mainly at short times.

In a biological setting, say in the motion of membrane receptors, these fast, high-frequency fluctuations might be the ones that matter most for binding dynamics. They are small, subtle, but they are precisely where the system can react — where it is sensitive enough to tiny conformational changes or external cues. Put differently, and if we go back to the theoretical section, we had introduced the idea of an effective spring constant k_p associated with each

Rouse mode. What this really means is that the low–p modes correspond to very stiff effective springs: they strongly couple the motion of distant beads, enforcing large-scale coherence in the chain's dynamics. At the other end of the spectrum, in the weakly coupled regime of large p, the situation is inverted — these modes make the beads much more sensitive to their local environment, allowing them to respond to small perturbations. These observations close our exploration of the Rouse model and set the stage for the concluding remarks.

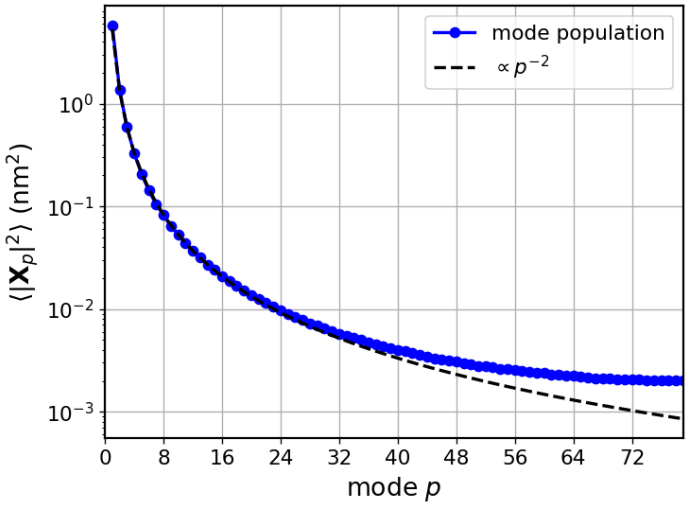

Figure 13 — Average population of the Rouse modes, $\langle |\boldsymbol{X}_p|^2 \rangle$, as a function of mode index p.

5. Conclusions

This work was an attempt to bring the Rouse model to life — not as an abstract theory, but as a tangible object that one can simulate, explore and shape. Starting from Langevin's description of coupled Brownian particles, we derived the chain dynamics, implemented the Euler-Maruyama scheme, and validated the physical consistency of the results through diffusion, structural and relaxation analyses. Altogether, the simulations reproduced the expected scaling behaviors and provided a coherent picture of Rouse dynamics.

Along the way, this project also offered an introduction to the vocabulary of stochastic differential equations and, more broadly, to some of the key concepts of mesoscopic physics. It is a field that fascinates me, even though it is often a demanding one — connecting two cliffs that stand on opposite sides is never easy.

From a physical point of view, the Rouse model has since been extended in many ways — most notably by the Zimm model, which accounts more accurately for hydrodynamic interactions and often provides a closer description of real polymer behavior. Other advances, to my knowledge, have mostly followed this direction. I had initially planned to explore another one: that of constraints. I did not have the time, but the main idea was to constrain a few degrees of freedom — as would naturally occur, for instance, in a membrane receptor whose backbone is partially trapped within a lipid slab, preventing it from moving freely.

Still, I do not see this project as finished. Much of the effort was devoted to building a simulation engine that is both solid and modular — and I hope I have shown that it can serve precisely the purpose of exploring physics with flexibility and precision. What happens when I force

inclusions into a membrane? What happens when I fix or release certain parts of the chain? I have the feeling that this Common Lisp approach is a natural companion to thought itself. And perhaps, through a few of these delicate tools, it will one day offer a few virtuous insights into the physics of living matter.

Bibliography

[1] B. Duplantier, Le mouvement brownien, "divers et ondoyant", Séminaire Poincaré **1**, 155 (2005).

[2] A. Einstein, Investigations on the Theory of Brownian Movement, Annalen Der Physik (n.d.).

[3] M. Smoluchowski, Sur le chemin moyen parcouru par les molécules d'un gaz et sur son rapport avec la théorie de la diffusion, Pisma Mariana Smoluchowskiego **1**, 479 (1924).

[4] P. Langevin, Sur la théorie du mouvement brownien, Comptes-Rendus De L'académie Des Sciences. Séance Du 9 Mars 1908 (1908).

[5] P. E. Rouse, A Theory of the Linear Viscoelastic Properties of Dilute Solutions of Coiling Polymers, The Journal of Chemical Physics **21**, 1272 (1953).

[6] W. J. Briels and J. T. Padding, *Theory of Polymer Dynamics*, in (Han-sur-Lesse Winter-school, 2007).

[7] M. Rubinstein, R. H. Colby, M. Rubinstein, and R. H. Colby, *Polymer Physics* (Oxford University Press, Oxford, New York, 2003).

[8] J. McCarthy, Recursive functions of symbolic expressions and their computation by machine, Part I, Communications of the ACM **3**, 184 (1960).

A. Installation

The reader most likely does not have a Common Lisp implementation readily available on their system. We therefore outline here, in broad terms, a complete installation procedure to get the project running. The project was developed on macOS, although an equivalent procedure should run on any Unix-like environment.

A.1. Common Lisp & Emacs

Common Lisp exists in many implementations. Among them, **Steel Bank Common Lisp** — usually referred to as `sbcl` — is arguably the most mature and performant. It provides a fast compiler, a complete runtime and excellent integration with modern developments tools.

Let us begin by installing `sbcl`. The reader can obtain the latest version directly from the official website, at https://www.sbcl.org/. Once installed, and since our project relies on several dependencies, we will need a package manager. For Common Lisp, the standard choice is **Quicklisp** whose official website provides a concise and well-written installation guide.

In principle, any text editor will do the job. However, it is far more convenient — and fitting — to use **GNU Emacs**, itself written in Lisp. With the addition of the `SLIME` plugin, Emacs becomes a fully interactive Lisp environment, allowing you to evaluate code, inspect data and shape your simulation in real time. Once Emacs, `sbcl` and Quicklisp installed, the reader simply needs to type `M-x list-packages RET`, search for `SLIME` and click to install it. All the necessary tools are now installed.

A.2. `Rouse` installation

The project is available on my GitHub repository at https://github.com/weld-lab/rouse. To get started, clone it using,

```
git clone https://github.com/weld-lab/rouse.git
```

Once this is done, create a configuration file `~/.sbclrc` (if it does not already exist), and add the following lines,

```
(pushnew #P"~/path/to/rouse" asdf:*central-registry*)
```

This ensures that Quicklisp can properly locate the project and its dependencies when you start `sbcl`. Next, start `sbcl`. Then, within the Lisp prompt, load the project with,

```
(ql:quickload "rouse")
```

This should start the Lisp system and load all required dependencies. Now start Emacs. From within Emacs, open the `SLIME` connection by typing,

```
M-x slime-connect RET localhost RET 4005 RET
```

This will start a REPL connected to the Lisp session launched earlier, allowing you to interact directly with the running system. You are now ready to run your first simulation! Try the following thing by typing (or copying/pasting),

```
(in-package #:rouse)

(defvar *sim*
  (sim:make-simulation
   :chain (top:make-linear-chain 20 :along :y :spaced-by 1d-9 :mass 1d-25)
   :temperature 273.15
   :gamma 1d-11
   :k 1.2d-2
   :dt 1d-13))

(view:view *sim* :mode :ortho-view)
```

This should open an `OpenGL` window displaying your simulation in real time. From there, you can either refer to the short user manual available at `path/to/rouse/viewer/README.md` or directly call functions from the REPL as you would in any interactive Lisp session. For example,

```
(sim:propagate *sim* :steps 10000)
```

will run the simulation for 10 nanoseconds.