Chemistry Modeling

Henry Thrasher

July 13, 2026

Introduction

We are motivated to model the eliptical motion of a nano particles ($< 10^{-9}$ m) using a Brownian motion simulator.

Discrete Randon Walk

Brownian motion is built up from fundamental stochastic principles. We begin from the discrete random walk.

We have a basic object (a particle) in a random position at time t:

$$x(t)$$

The particle, X, is moving on a 1D line. At eaxch time step, it moves left or right with equal probability.

$$X_{n+1} = X_n + \Delta X_n, \quad \text{where} \quad \Delta X_n = \begin{cases} +\ell, & p = 0.5 \\ -\ell, & p = 0.5 \end{cases}$$

After N steps:

$$X_N = \Delta X_1 + \Delta X_2 + ... + \Delta X_N$$

Naturally, we can conclude that the expected displacement is equal to the theoretical expected value, $\mathbb{E}[X_N]$, and the variance is the theoretical variance, σ^2:

$$\mathbb{E}[X_N] = 0, \quad \text{where} \quad \mathbb{E}[X_N] = \sum_{x=1}^{N} xP(X = x) = 0$$

$$\sigma^2 = N\ell^2, \quad \text{where} \quad \sigma^2 = \sum_{x=1}^{N} xP(X = x)(x - \mathbb{E}[X])^2 = \mathbb{E}[X_N^2] - \mathbb{E}[X_N]^2 = \mathbb{E}[X_N^2] = Nl^2$$

We can see that expected value is 0 and variance grows linearly with steps.

0.1 Time

Looking now in time, let each discrete step require a fixed amount of time τ. After N steps the elapsed time is t:

$$t = N\tau$$

So

$$N = \frac{t}{\tau}$$

Substituting into variance:

$$Var(X(t)) = Nl^2 = \frac{l^2}{\tau}t$$

Let the one dimension diffusion constant be:

$$D = \frac{l^2}{2\tau}$$

Thus:

$$Var(X(t)) = 2Dt$$

Program Simulation

Simulating

In the jupyter notebook, the program simulates the same one dimension random walk described above. The main parameters are:

$$N, \quad M, \quad \ell, \quad \text{seed}$$

where N is the number of steps, M is the number of walkers, ℓ is the step length, and the seed makes the same random walk reproducible.

`validate_parameters` checks that the number of steps, number of walkers, and step length are positive. This prevents the simulation from running with impossible values.

`simulate_random_walk` creates an array of random steps:

$$\Delta X_n \in \{-\ell, +\ell\}$$

Then it uses the cumulative sum to build the full position history for every walker:

$$X_N = \sum_{n=1}^{N} \Delta X_n$$

The initial position $X_0 = 0$ is added as the first column. Therefore the output is a matrix with M rows and $N + 1$ columns. For example:

$$(M, N + 1)$$

`analyze_random_walk` computes the simulated mean, variance, standard deviation, and mean squared displacement at every step. The theoretical values are also computed as:

$$Var(X_N) = N\ell^2$$

and

$$\sigma = \sqrt{N\ell^2}$$

The method proves that the program is working by comparing the simulated values to the theoretical values. Since the expected mean is 0, the simulated final mean should be close to 0.