The Langevin Approach

The Random Walk and the Langevin equation

The concept of a random walk and its continuum limit – diffusion – introduced in the previous chapter, expresses the time evolution of the probability distribution $p(x,t)$ for a particle's position x by the diffusion equation:

$$\frac{\partial p}{\partial t} = D\frac{\partial^2 p}{\partial x^2},$$

which is a standard example of a so called *Fokker-Planck* equation, which is second-order in space and first-order in time.

In contrast, the *Langevin equation* provides a stochastic differential equation for the particle's trajectory $x(t)$. To understand it, consider the random hopping motion of a particle on a 1d lattice over a small time increment Δt:

$$x(t+\Delta t) = x(t) + \Delta x(t)$$

Here, $\Delta x(t)$ is a random displacement. If the lattice spacing is a, we define the step statistics as:

$$\Delta x(t) = \begin{cases} +a & \text{with probability } \nu\Delta t \\ -a & \text{with probability } \nu\Delta t \\ 0 & \text{with probability } 1-2\nu\Delta t \end{cases}$$

This defines a discrete-time, discrete-space random walk. The average and variance of the step are easily derived using binomial statistics:

- Mean: $\langle \Delta x \rangle = 0$
- Variance: $\langle (\Delta x)^2 \rangle = 2a^2\nu\Delta t = 2D\Delta t$

The steps $\Delta x(t)$ are uncorrelated across time.

To take the continuum limit, we let both $a \to 0$ and $\Delta t \to 0$ in such a way that:

$$a \propto \sqrt{\Delta t}$$

In this limit, we obtain the **Langevin equation**:

$$\dot{x}(t) = \eta(t)$$

where $\eta(t) \equiv \frac{\Delta x(t)}{\Delta t}$ is a stochastic noise satisfying:

$$\langle \eta(t) \rangle = 0$$

$$\langle \eta(t)\eta(t') \rangle = \Gamma \delta(t - t')$$

This $\eta(t)$ is known as white noise — it has zero mean and is uncorrelated at different times. Γ measures its amplitude.

The Langevin equation tells us that the velocity $\dot{x}(t)$ is purely driven by noise. We can formally integrate it:

$$x(t) - x_0 = \int_0^t \eta(t')\, dt'$$

Taking ensemble averages:

- Mean displacement:
$$\langle x(t) - x_0 \rangle = 0$$
- Mean square displacement:
$$\langle [x(t) - x_0]^2 \rangle = \int_0^t \int_0^t \langle \eta(t')\eta(t'') \rangle \, dt' \, dt'' = \Gamma \int_0^t dt' = \Gamma t$$

Comparing this with the diffusion equation result, we identify:

$$\Gamma = 2D$$

Hence, the Langevin description yields the same physical behavior — not just the mean-square displacement but also the full probability distribution $p(x, t)$ — as the diffusion (Fokker-Planck)

equation. This equivalence arises from the fact that the integral of many small, independent random steps leads to a Gaussian distribution, in agreement with the solution of the diffusion equation.

For more details, see: *Stochastic Processes in Physics and Chemistry* by N.G. van Kampen (North Holland, 1981).

Brownian Motion

Let us now examine Brownian motion, originally observed as the erratic motion of colloidal particles suspended in a fluid. These particles undergo constant collisions with surrounding (smaller) fluid molecules, which results in seemingly random movement.

From a coarse-grained perspective — where we do not track each individual collision — this appears as motion under random forces. This statistical treatment introduces irreversibility at the macroscopic level, even though the underlying molecular dynamics are reversible.

The Langevin equation provides a way to model this behavior. For a particle of mass m in one dimension, Langevin proposed the equation:

$$m\ddot{x} = -\gamma\dot{x} + f(t)$$

Here:

- $-\gamma\dot{x}$ is a frictional damping force, where γ is the damping coefficient.
- $f(t)$ is a random force due to molecular collisions.

  Often, the mobility is defined as $\mu = 1/\gamma$ — note that this is unrelated to chemical potential.

Noise Properties

In principle the random forces are correlated in time since the molecular collisions which cause them are correlated and have some definite duration.

Let us assume that there is some correlation time t_c over which $\langle f(t_1)f(t_2)\rangle = g(t_1 - t_2)$ decays rapidly as shown in the sketch below:

Then as long as we consider timescales $\gg t_c$ we can safely replace $g(t_1 - t_2)$ by a delta function. Thus we can make the approximation of white noise

$$\langle f(t)\rangle = 0$$

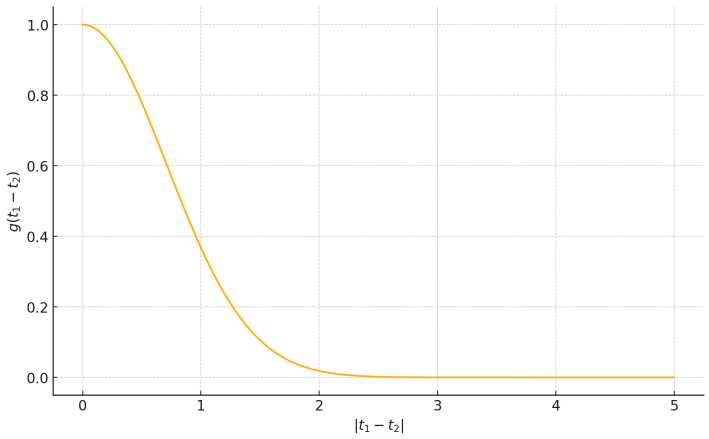

Figure 1: Sketch of $g(t1 - t2)$ against $|t1 - t2|$

$$\langle f(t_1)f(t_2)\rangle = \Gamma\delta(t_1 - t_2)$$

Solving the Langevin Equation (velocity)

Let's set $m = 1$ for simplicity and solve the equation:

$$\dot{v} + \gamma v = f(t)$$

We apply an integrating factor:

$$\frac{d}{dt}\left[ve^{\gamma t}\right] = e^{\gamma t}f(t)$$

Integrating both sides:

$$v(t) = v_0 e^{-\gamma t} + \int_0^t e^{-\gamma(t-t')}f(t')\,dt'$$

Taking the average:

$$\langle v(t)\rangle = v_0 e^{-\gamma t}$$

Thus:

- At short times: $(\gamma t \ll 1)$: $\langle v\rangle \approx v_0$ ie. friction is negligible.
- At long times: $(\gamma t \gg 1)$: $\langle v\rangle \to 0$ ie. the system loses memory of the initial velocity.

Mean-square velocity

We now compute (do it as an exercise):

$$\langle v(t)^2 \rangle = v_0^2 e^{-2\gamma t} + \Gamma \int_0^t e^{-2\gamma(t-t')} dt' = v_0^2 e^{-2\gamma t} + \frac{\Gamma}{2\gamma}\left(1 - e^{-2\gamma t}\right)$$

Solution

From the solution for $v(t)$ (with $m = 1$): $\quad v(t) = v_0 e^{-\gamma t} + \displaystyle\int_0^t e^{-\gamma(t-t')} f(t')\, dt'.$

$$\Rightarrow \; v(t)^2 = v_0^2 e^{-2\gamma t} + 2v_0 e^{-\gamma t} \int_0^t e^{-\gamma(t-t')} f(t')\, dt' + \int_0^t \int_0^t e^{-\gamma(2t-t'-t'')} f(t') f(t'')\, dt'\, dt''.$$

$$\langle v(t)^2 \rangle = v_0^2 e^{-2\gamma t} + 2v_0 e^{-\gamma t} \int_0^t e^{-\gamma(t-t')} \underbrace{\langle f(t') \rangle}_{=0}\, dt'$$

$$+ \int_0^t \int_0^t e^{-\gamma(2t-t'-t'')} \underbrace{\langle f(t') f(t'') \rangle}_{=\Gamma\,\delta(t'-t'')}\, dt'\, dt''$$

$$= v_0^2 e^{-2\gamma t} + \Gamma \int_0^t e^{-2\gamma(t-t')}\, dt'$$

$$= v_0^2 e^{-2\gamma t} + \Gamma \left[-\frac{1}{2\gamma} e^{-2\gamma(t-t')} \right]_{t'=0}^{t'=t}$$

$$= v_0^2 e^{-2\gamma t} + \frac{\Gamma}{2\gamma}(1 - e^{-2\gamma t}).$$

$$\boxed{\langle v(t)^2 \rangle = v_0^2 e^{-2\gamma t} + \frac{\Gamma}{2\gamma}\left(1 - e^{-2\gamma t}\right)}$$

Implying that at

- Short times: $\langle v^2 \rangle \approx v_0^2$
- Long times: $\langle v^2 \rangle \to \Gamma/(2\gamma)$

At equilibrium, the equipartition theorem gives:

$$\frac{1}{2} m \langle v^2 \rangle = \frac{1}{2} k_B T$$

Using this to identify Γ:

$$\Gamma = 2\gamma k_B T$$

This important result relates the noise strength to the damping and temperature — they have the same microscopic origin (molecular collisions).

Mean-square displacement

We now integrate $v(t)$ again to get position $x(t)$ (with $m = 1$):

Using the result above and substituting $\Gamma = 2\gamma k_B T$, we find:

$$\langle [x(t) - x_0]^2 \rangle = \frac{(v_0^2 - k_B T)}{\gamma^2}(1 - e^{-\gamma t})^2 + \frac{2k_B T}{\gamma}\left[t - \frac{1 - e^{-\gamma t}}{\gamma}\right]$$

Limiting behaviours:

- Short times: ($\gamma t \ll 1$):

  $$\langle [x(t) - x_0]^2 \rangle \approx v_0^2 t^2$$

  (correspinding to *ballistic* motion)

- Long time ($\gamma t \gg 1$):

  $$\langle [x(t) - x_0]^2 \rangle \approx \frac{2k_B T}{\gamma} t$$

  (corresponding to *diffusive motion*)

The effective diffusion constant is:

$$D = \frac{k_B T}{\gamma}$$

This is the **Einstein relation**, connecting the rate of diffusion to temperature and damping. It is useful as it allows an explicit expression for the diffusion constant if one knows γ. A famous example is a sphere: the equation for fluid flow past a moving sphere may be solved and yields $\gamma = 6\pi\eta a$ where a is the radius of the sphere and here η is the fluid viscosity. This gives

$$D = \frac{6\pi\eta a}{kT}$$

which is the **Stokes-Einstein** formula for the diffusion constant of a colloidal particle.

External Forces and Mobility

Now consider a charged particle with charge q under an external electric field E. The Langevin equation becomes:

$$m\dot{v} = -\gamma v + qE$$

At long times, the particle reaches a steady drift velocity:

$$\langle v \rangle = \frac{qE}{\gamma} = \frac{qED}{k_B T}$$

Defining the mobility μ by $\langle v \rangle = \mu q E$, we get the **Nernst-Einstein relation**:

$$\mu = \frac{D}{k_B T}$$

This relation connects the response of a system to an external perturbation (mobility) with its internal fluctuations (diffusivity).

Molecular Dynamics simulation of Brownian motion for a colloid particle in a liquid suspension

Movies/brownian_colloid.mp4

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from numba import njit

# Parameters
n_fluid = 300
box_size = 20.0
n_steps = 10000
sigma_f = 1.0
sigma_c = 10.0
```

```
epsilon = 0.05
mass_f = 1.0
mass_c = 2.0
dt = 1e-5
dt_e = 1e-5

# Echo the parameter values
print("Simulation Parameters:")
print(f"n_fluid = {n_fluid}")
print(f"box_size = {box_size}")
print(f"n_steps = {n_steps}")
print(f"sigma_f = {sigma_f}")
print(f"sigma_c = {sigma_c}")
print(f"epsilon = {epsilon}")
print(f"mass_f = {mass_f}")
print(f"mass_c = {mass_c}")
print(f"dt = {dt}")

 # Derived quantities

sigma_f6 = sigma_f ** 6
sigma_f12 = sigma_f **12
sigma_cf = 0.5 * (sigma_f + sigma_c)
sigma_cf6 = sigma_cf ** 6
sigma_cf12 = sigma_cf ** 12

np.random.seed(42)

# Safe initialization to avoid overlaps with colloid and other fluid particles
def initialize_fluid_positions(n_fluid, box_size, sigma_f, sigma_c, colloid_pos, min_dist_fa
    min_dist_ff = min_dist_factor * sigma_f
    min_dist_cf = min_dist_factor * 0.5 * (sigma_f + sigma_c)
    positions = []
    max_attempts = 20000

    for _ in range(n_fluid):
        for attempt in range(max_attempts):
            trial = np.random.rand(2) * box_size
            too_close = False

            # Check distance to colloid center
            if np.linalg.norm(trial - colloid_pos[0]) < min_dist_cf:
```

```
                too_close = True

            # Check distances to already placed fluid particles
            for existing in positions:
                if np.linalg.norm(trial - existing) < min_dist_ff:
                    too_close = True
                    break

            if not too_close:
                positions.append(trial)
                break
        else:
            raise RuntimeError("Failed to place a fluid particle without overlap after many a

    return np.array(positions)

colloid_pos = np.array([[box_size / 2, box_size / 2]])
fluid_pos = initialize_fluid_positions(n_fluid, box_size, sigma_f, sigma_c, colloid_pos)
fluid_vel = (np.random.rand(n_fluid, 2) - 0.5)
colloid_vel = np.zeros((1, 2))

@njit
def compute_forces_numba(fluid_pos, colloid_pos, sigma_cf6, sigma_cf12, sigma_f6, sigma_f12,
    forces_f = np.zeros_like(fluid_pos)
    force_c = np.zeros_like(colloid_pos)

    for i in range(n_fluid):
        # Fluid-colloid interaction
        rij = fluid_pos[i] - colloid_pos[0]
        rij -= box_size * np.round(rij / box_size)
        dist2 = np.dot(rij, rij)
        if dist2 < (2.5 ** 2) * ((sigma_cf6 ** (1/6)) ** 2) and dist2 > 1e-10:
            r2 = dist2
            r6 = r2 ** 3
            r12 = r6 ** 2
            fmag = 48 * epsilon * ((sigma_cf12 / r12) - 0.5 * (sigma_cf6 / r6)) / r2
            fvec = fmag * rij
            forces_f[i] += fvec
            force_c[0] -= fvec

        for j in range(i + 1, n_fluid):
            rij = fluid_pos[i] - fluid_pos[j]
```

```
            rij -= box_size * np.round(rij / box_size)
            dist2 = np.dot(rij, rij)
            if dist2 < (2.5 ** 2) * ((sigma_f6 ** (1/6)) ** 2) and dist2 > 1e-10:
                r2 = dist2
                r6 = r2 ** 3
                r12 = r6 ** 2
                fmag = 48 * epsilon * ((sigma_f12 / r12) - 0.5 * (sigma_f6 / r6)) / r2
                fvec = fmag * rij
                forces_f[i] += fvec
                forces_f[j] -= fvec

    return forces_f, force_c

    fluid_history = []
colloid_history = []
forces_f, force_c = compute_forces_numba(fluid_pos, colloid_pos, sigma_cf6, sigma_cf12, sigma

n_equilibration = 2000  # Number of steps to equilibrate before tracking

# Equilibration phase (no history recorded)

for step in range(n_equilibration):
    fluid_pos += fluid_vel * dt_e + 0.5 * forces_f / mass_f * dt**2
    colloid_pos += colloid_vel * dt_e + 0.5 * force_c / mass_c * dt**2
    fluid_pos %= box_size
    colloid_pos %= box_size
    new_forces_f, new_force_c = compute_forces_numba(fluid_pos, colloid_pos, sigma_cf6, sigma
    fluid_vel += 0.5 * (forces_f + new_forces_f) / mass_f * dt_e
    colloid_vel += 0.5 * (force_c + new_force_c) / mass_c * dt_e
    forces_f = new_forces_f
    force_c = new_force_c
    # if step < 10:  # Log only the first few steps
    #     force_mag = np.linalg.norm(force_c[0])
    #     print(f"Step {step:3d} | Colloid Pos: {colloid_pos[0]} | Vel: {colloid_vel[0]} | |

# Production phase (history recorded)

for step in range(n_steps):
    fluid_pos += fluid_vel * dt + 0.5 * forces_f / mass_f * dt**2
    colloid_pos += colloid_vel * dt + 0.5 * force_c / mass_c * dt**2
    fluid_pos %= box_size
    colloid_pos %= box_size
```

```
    new_forces_f, new_force_c = compute_forces_numba(fluid_pos, colloid_pos, sigma_cf6, sigma
    fluid_vel += 0.5 * (forces_f + new_forces_f) / mass_f * dt
    colloid_vel += 0.5 * (force_c + new_force_c) / mass_c * dt
    forces_f = new_forces_f
    force_c = new_force_c
    if step % 10 == 0:
        fluid_history.append(fluid_pos.copy())
        colloid_history.append(colloid_pos.copy())

fig, ax = plt.subplots()
# Calculate figure and plot scale parameters
fig_width_inch = fig.get_size_inches()[0]
dpi = fig.dpi
axis_length_pt = fig_width_inch * dpi
marker_scale = 0.1  # Scale factor for visibility
fluid_marker_size = (marker_scale * axis_length_pt / box_size) ** 2
colloid_marker_size = 0.7*(marker_scale * sigma_c / sigma_f * axis_length_pt / box_size) ** 2
fluid_scatter = ax.scatter([], [], s=fluid_marker_size, c='blue')
colloid_scatter = ax.scatter([], [], s=colloid_marker_size, c='red')
trajectory, = ax.plot([], [], 'r--', linewidth=1, alpha=0.5)
ax.set_xlim(0, box_size)
ax.set_ylim(0, box_size)
ax.set_xticks([])
ax.set_yticks([])
ax.set_xticklabels([])
ax.set_yticklabels([])
ax.set_aspect('equal')
colloid_traj = []

def init():
    empty_offsets = np.empty((0, 2))
    fluid_scatter.set_offsets(empty_offsets)
    colloid_scatter.set_offsets(empty_offsets)
    trajectory.set_data([], [])
    return fluid_scatter, colloid_scatter, trajectory

def update(frame):
    fluid_scatter.set_offsets(fluid_history[frame])
    colloid_scatter.set_offsets(colloid_history[frame])
    colloid_traj.append(colloid_history[frame][0])
    traj_array = np.array(colloid_traj)
    trajectory.set_data(traj_array[:, 0], traj_array[:, 1])
```

```
    return fluid_scatter, colloid_scatter, trajectory

ani = animation.FuncAnimation(fig, update, frames=len(fluid_history), init_func=init, blit=T
ani.save("brownian_colloid.mp4", writer="ffmpeg", fps=30)
print("Simulation complete. Video saved as 'brownian_colloid.mp4'.")
```

More about the scientists mentioned in this chapter:

Paul Langevin

Robert Brown

Albert Einstein

Walther Nernst

George Stokes