INTERNSHIP REPORT

naoufal elaisati

2024

Contents

1 Internship description

I had the good fortune to work under the supervision of Sophie Marbach in the laboratory PHENIX in Sorbonne University in Paris, on the subject of diffusion of a particular type of Junas colloid, with theory and data analysis. During my internship, I also had the opportunity to discuss with Federico Paratore (ETH Zürich), who provided data from an experiment he is currently conducting.

Dr. Sophie Marbach is currently serving as a CNRS researcher at Sorbonne University. Her work focuses on non-equilibrium transport processes in soft matter, studying micro to nanoscale systems where unusual phenomena come to light. She builds theoretical models in close contact with experiments.

Federico Paratore, currently a Senior Postdoctoral Researcher at ETH Zürich, Switzerland, completed his Ph.D. at the Technion - Israel Institute of Technology. He is interested in hydrodynamics, transport phenomena, surface science, and electrokinetics at the micro/nanoscale. His research encompasses various applications including flow and particle control, energy storage, (bio)molecular analysis, and collective motion.

The PHENIX laboratory, an acronym for Physicochemistry of Electrolytes and Interfacial Nanosystems laboratory, is housed at Sorbonne University. Reimagined with a new name since 2014, the laboratory now hosts a team of 87 professionals at the crossroades between physics and chemistry. The lab's work is characterized by its three areas of expertise: established proficiency in experimentation and chemical synthesis, the ability to execute numerical simulations over an extensive range of lengths and time scales, and a significant commitment to the creation of suitable theoretical models. Furthermore, the laboratory is organized into three research teams: Inorganic Colloids, Modeling and Multiscale Experiments, and Electrochemistry and Ionic Liquids.

As an intern working on colloids, I was part of the Inorganic Colloids team. I have attended monthly basis meetings with the team, where individual team members would share their current work and findings with the rest of the group, offering a platform for discussion, feedback, and idea exchange. The team meetings and the seminars that the laboratory regularly holds taught me a great variety of things about the flow of particles, non-equilibrium transport systems, molecular biophysics, computational modeling techniques, and various other topics. My involvement wasn't limited to these professional interactions, I was also an active participant in the lab's social life. Attending birthday celebrations and indulging, perhaps a bit excessively, in the treats during coffee breaks. I also took part in more formal social events such as team-building day ("journée de cohésion") and casual lab gatherings for drinks.

2 Introduction

Microorganisms have always captivated scientists due to their impact on health and on the environment [1]. One key feature of numerous microorganisms is their ability to move, swim or propel, which necessarily affects all the processes that they take a part in. Consequently, it is important to probe and quantify their motion to develop accurate models that can predict their behavior.

One of the primary objectives of studying the motion of a particle (or a microorganism) is to gain a deeper understanding of how it navigates within a medium [2, 3]. In particular, understanding how far a particles diffuses over long times is relevant to quantify the efficiency of mixing, transporting, and distributing particles. Although diffusion is a complex concept, we can define it in simple terms as describing how particles spread from areas of high concentration to areas of lower concentration. Random, spontaneous movement of particles due to thermal agitation is called passive diffusion [4]. In contrast, active diffusion involves some

form of propulsion or energy consumption [5].

Since microorganisms are micron-sized, they operate at scales where it is natural to expect that they diffuse, and perform a random walk often modeled with Brownian motion [4]. Yet, Brownian motion is often not sufficient to describe a microorganism's motion; but more advanced ingredients are required that extend or deviate significantly from Brownian motion. This is due to the self-propulsion of microorganisms via rotating or pulsating flagellae [6], which means their motion is akin to active diffusion, and inherently complex.

This complexity is evident for example in the 'run-and-tumble'-like motion exhibited by certain microorganisms, Such as motile E. Coli bacteria [7]. The so-called 'run-and-tumble' involves alternating sequences of runs and tumbles. During a run, the active particle propels itself in a constant direction, but during a tumble, it stays stationary while reorienting itself, preparing for the subsequent run.

To make progress, it is often useful to reduce complexity, and in that regard, lab-made micro-scale particles that mimic these kinds of motion are especially useful. For example, they, unlike microorganisms, do not reproduce, hence simplifying studies. These so-called *active* particles, consume energy in the fluid, for example in the form of chemical compound, and propel beyond Brownian motion [5].

Investigating *active* particle motion has allowed us to classify different types of motion. For example, the 'run-and-tumble' particles (RTPs) are considered as a class of active particles [5]. Another class of motion for such active particles are active Brownian Particles (ABPs). ABPs exhibit a random walk behavior only over long times, due to a combination of their self-propulsion and a random reorientation of their direction of motion. This behavior is modeled with a Brownian walker with an added self-propulsion term.

Specific differences arise in these various classes of motion. The active Brownian particles (ABPs) and run-and-tumble particles (RTPs) differ in their angular relaxation (continuous versus discrete) [8]. Specifically, the time taken to change direction is different, with RTPs exhibiting more persistence in their runs compared to ABPs that are less persistent as their direction continuously changes due to rotational diffusion.

Models of active particles, (ABP, RTP and beyond), are an important tool to understand the active diffusion. Not only do they help to understand further microorganism motion, but they also provide insights into non-equilibrium physics, as the active particles are in a constant state of energy consumption. Probing active diffuse may thereby enrich our understanding of a wide array of natural and artificial systems that operate far from equilibrium [9].

To analyze particle motion it is common to construct particle trajectories by linking subsequent images [10]. From trajectories one can compute the mean squared displacement of particles, which allows one to extract dynamic properties, such as a particle's diffusion coefficient. The diffusion coefficient characterizes typically how far a particle might diffuse. Such linking methods require to distinguish particle trajectories accurately, and therefore become limited in dense systems where particles are too close to one another to link them without errors. Yet, high particle densities are especially interesting since exotic collective effects emerge in these conditions [11, 12].

The recently introduced "Countoscope" method is an alternative technique to investigate particle dynamics, that does not require particle linking and that can thus operate without errors at high densities. By analyzing the statistical properties of the number of particles in sub-volumes, it allows one to extract various dynamic properties, including a particle's self-diffusion coefficient [13] (also unpublished results). Yet, its ability to probe more complex motion such as active diffusion remains to be established.

During this internship my goal was to analyze a particular type of colloid using both the mean square displacement method and the countoscope method, with the aim of gaining a deeper understanding of how these colloids diffuse within a medium. Our approach will begin with the examination of basic cases of random walks (Sec. 3). We will then explore passive diffusion using both methods (Sec. 4), ultimately progressing to the investigation of active diffusion (Sec. 5).

3 Stochastic simulations

A *stochastic simulation* is a simulation of a system that has variables that can change stochastically (or randomly) with individual probabilities. Such simulations are integral to modern chemical and physical research and are well-suited for such applications because of the inherent random nature of underlying mechanisms at these scales including processes such as Brownian motion, but also reaction kinetics.

In a Brownian motion simulation, which is a specific kind of stochastic simulation, the goal is to simulate the movement of a single molecule, with a focus on the trajectory of this molecule. In contrast, in a simulation where we would track the concentration of particles in space and time, the focus shifts from the trajectory of individual molecules to the final positions of a multitude of molecules. [14]

3.1 Random walk with discrete steps

As a preliminary step to understand Brownian motion, I simulated a random walk. A *random walk* is a random process that describes a path that consists of a succession of random steps in space occuring at discrete times.

To start, I considered the case of a particle moving in one dimension, with an equal probability of moving to the right or to the left. Its initial position is $x_0 = 0$, and it performs a step of finite length a (left or right) at each interval of time Δt. After n iterations its position is x_n.

To investigate the particle's trajectory it is interesting to investigate the expected value of the particle position $\langle x_n \rangle$ or of higher moments such as $\langle x_n^2 \rangle$. Here the symbol $\langle \cdot \rangle$ corresponds to multiple independent realizations of the random walk or can be thought of the behavior of the particle at sufficiently long times. To investigate position statistics, it is important to realize that the simulation is *Markovian*, meaning that it has no memory because the future state depends only on the present state, not on past states. I thus could determine that the expected value of the particle's position is zero, $\langle x_n \rangle = 0$.

I then calculated the expected value of the squared position. I found with mathematical steps that this value is proportional to time, as shown in Equation 1 below:

$$\langle x_n^2 \rangle = n \cdot a^2 = \frac{a^2}{\Delta t} t \tag{1}$$

where $t = n\Delta t$ is time. The random walk in itself is not Gaussian (the steps are not Gaussian-distributed) but the distribution of the final position after a large number of steps can be approximated by a Gaussian distribution.

Then I simulated the random walk of particles in higher dimensions, both in 2 dimensions and 3 dimensions, using a custom-made Python script, for the 2 dimension case, I used $np.random.choise([-1, 1], size = numberSteps)$, which generates the position of x and y, to be random either -1 or 1, for all the number steps, thus, it can go either (up/down) and in the same time, either (back or forward), then I plotted the sequence, the same logic was applied for the 3 dimension case. I report typical trajectories in Figures 1 and 2.

3.2 Brownian motion and the overdamped Langevin equation

I then progressed to a more complex model, where the random moves are continuous in space, using the so-called *overdamped Langevin equation.*

The *Langevin equation* (non-overdamped), presented in Equation 2 , is a stochastic differential equation describing how a system evolves when subjected to a combination of deterministic and random forces. It is based on an extension of Newton's second law of motion, with the inclusion of stochastic forces.

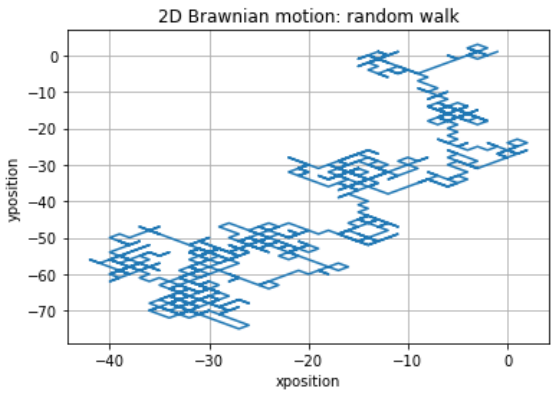

Figure 1: Random walk in 2D

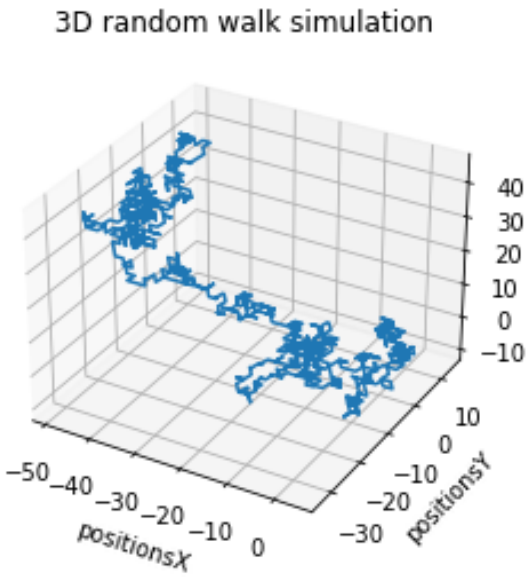

Figure 2: Random walk in 3D

In one dimension, a particle's position x satisfies

$$m\frac{dv}{dt} = -\gamma v + F(x) + \sqrt{2k_B T\gamma}\eta(t) \tag{2}$$

where

- m is the mass of the particle,
- $v = \frac{dx}{dt}$ is the velocity of the particle,
- t is time,
- γ is a friction coefficient,
- $F(x)$ is an external force on the particle,
- $\eta(t)$ is a Gaussian white noise term, where $\langle\eta(t)\rangle = 0$ and $\langle\eta(t)\eta(t')\rangle = \delta(t-t')$, where $\langle\cdot\rangle$ here means an average over realizations of the noise.
- k_B is Boltzmann's constant and T is temperature.

In the above Equation 1, the left-hand side corresponds to the inertia term. On the right hand side, we enumerated the different forces on the particle. The friction force $-\gamma v$ corresponds to the hydrodynamic friction exerted by molecules surrounding the particle, while $+\sqrt{2k_B T\gamma}\eta(t)$ is our stochastic force due to surrounding molecules impinging on the particle's surface. Both of these terms are related *via* the so-called fluctuation-dissipation balance, which determines the amplitude of one relative to the other. Finally, $F(x)$ corresponds to potential external forces on the particle which we will here take to be 0.

The inertia term $m\frac{dv}{dt}$ in Eq. 1 is neglected in the *overdamped Langevin equation*. This approximation holds true when the friction is very high, or, alternatively, when the system moves so slowly that it is constantly in a state of quasi-equilibrium with the heat bath around it, which is the case of micron-sized particles [15]. The resulting *overdamped Langevin equation* is

$$\frac{dx}{dt} = \sqrt{2\frac{k_B T}{\gamma}}\eta(t). \tag{3}$$

I implemented the above random walk with now steps of arbitrary length using a custom made Python routine where

$$x_n = x_{n-1} + \sqrt{2\frac{k_B T}{\gamma}} W_n \Delta t \tag{4}$$

where $W_n \in \mathcal{N}(0,1)$ is a random number sampled from a Gaussian distribution with mean 0 and standard deviation 1. This implementation corresponds to *Brownian dynamics*.

The resulting trajectories, as illustrated in Figures 3 and 4, appear smoother and more continuous compared to those seen in Figures 1 and 2 of the discrete random walk, as the trajectories in the random walk are comprised of a sequence of discrete steps.

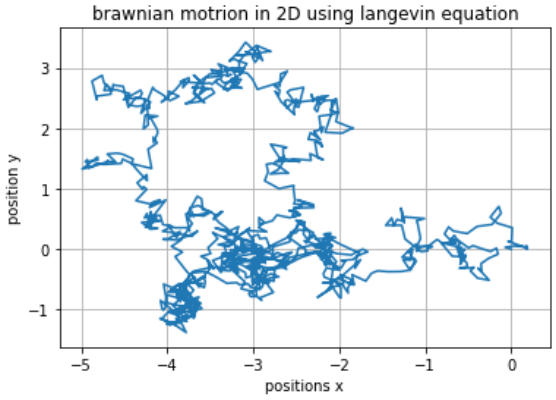

Figure 3: Random walk using overdamped Langevin equation in 2D

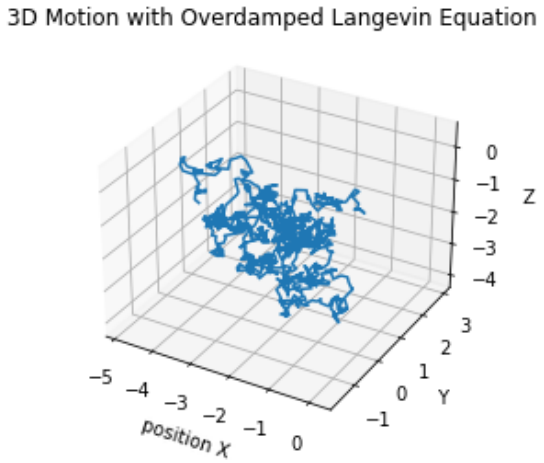

Figure 4: Random walk using overdamped Langevin equation in 3D

The expected value $\langle x(t) \rangle = 0$ (the average after a large number of trials) in this case will also be zero due to the equal probabilities to go on either side (if the particle starts at the origin).

Using elementary math, one can show that the probability distribution $p(x,t)$ to find the particle at some position x after some time t obeys equation 5:

$$\frac{\partial p}{\partial t} = D \frac{\partial^2 p}{\partial x^2} \tag{5}$$

Equation 5 is part of a generic class of equations called Fokker-Planck equations. Here D is the *diffusion coefficient* of the particle, and verifies Einstein's relation [4]

$$D = \frac{k_B T}{\gamma}. \tag{6}$$

I checked that the following expression for $p(x,t)$ is indeed a solution of Equation 5,

$$p(x,t) = \sqrt{\frac{1}{2\pi D t}} \exp\left(-\frac{x^2}{2Dt}\right). \tag{7}$$

I then calculated the expected value of the squared position $x(t)^2\rangle = \int x^2 p(x,t) dx$ using this expression, and I obtained

$$\langle x(t)^2 \rangle = 2Dt. \tag{8}$$

Compared to Equation 1, we can make sense of the diffusion coefficient $D = a^2/2\Delta t$, which characterizes the ratio between the typical squared step sizes and the timestep.

Next, I worked on building a counter that can count particles that are contained within and observation box, as they move in and out of that observation field. To do so, I divided simulation images into sub-spaces, and counted the number of particles in each of these observation boxes with a custom made Python routine. This counter counts how many particles are in and out of a box, and give us these fluctuations through time.

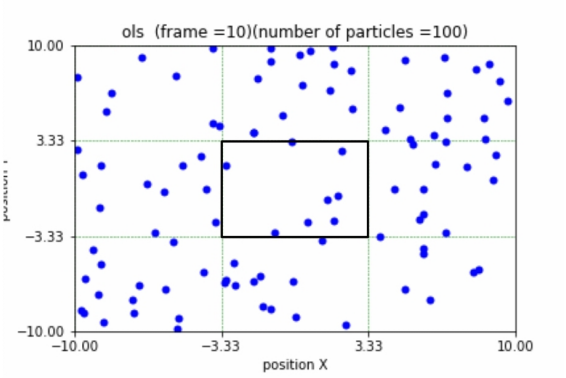

Figure 5: square subspace in an observed space.

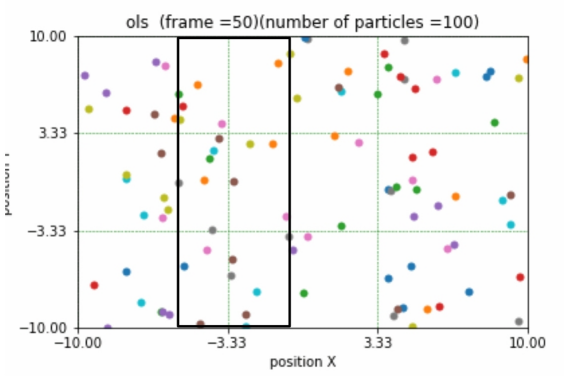

Figure 6: rectangle subspace in an observed space.

As we can see in Figure 7, a snippet of the code used to count particles from randomly generated data, the function "fluctuations", calculates and return the number of particles in a subspace at every time step. It first takes particle position data x and y. Then it divides the observed space into sub-spaces, characterized by a section length that can be different on the x or y axis, *i.e.* for rectangles of arbitrary shape. This arbitrary shape is required to produce rectangles that optimize the analysis of experimental data, depending on whether the camera to aquire images was rectangular or not. Then, for each particle at each time step, the function calculates the coordinates of the *box* in which the particle is located by dividing the particle's x and y coordinates by section length in x, L_x and section length in y, L_y, respectively. These values are then converted to integers, effectively rounding down to which subspace they are in at which time.

```
#counting particles in each sub_space
def fluctuations(x, y, xmin, ymin, section_lengthx,section_lengthy, num_steps, number_particles, number_lines):
    num_particles_insection = np.zeros((number_lines**2, num_steps))
    for i in range(num_steps):
        for j in range(number_particles):
            xposition = int((x[j, i]) / section_lengthx)
            yposition = int((y[j, i]) / section_lengthy)
            num_particles_insection[xposition * number_lines + yposition, i] += 1
    return num_particles_insection
num_particles_insection =fluctuations (x, y, xmin, ymin, section_lengthx,section_lengthy, num_steps, number_particles, number_lines)

```

Figure 7: Snippet of the Python code I implemented to count particles in a box from generated Brownian dynamics trajectories.

To highlight the typical properties of the observed counts, I tested the code on randomly generated trajectories from Brownian dynamics in 2D. Note here, that I coded periodic boundary conditions on the particle positions so that particles could stay in a finite region of space. Periodic boundary conditions mean that if a particle exits on one side of the total simulation box, it is put back on the other side. I apply the counting particles code on an observed total simulation space where I have 99 particles in total. I divided the simulation space into 9 observation boxes, and I calculated the number of particles in these subspaces with time, every 10-time steps for a few time steps, see Figure 8.

Figure 8: Number of particles in a subspace with time for all the different subspaces of a finite simulation space. The titles of each subplot indicate the calculated average of particles in the box for this short time series.

4 Experimental setup

The experiment we aim to understand, currently conducted by Federico Paratore, investigates the behavior of a metallo-dielectric Janus particle in a microfluidic device, under an applied electrical field.

When an alternating electrical field is applied, see Fig. 11, we observe self-propulsion: the colloid becomes active. The propulsion is due to a phenomenon called *induced-charge electro-osmosis* (ICEO) [16]. Briefly, the applied electric field generates a net surface charge near polarizable surfaces. In our case, this charge is predominantly on the Gold (Au) surface, because Gold is more polarizable than Chromium (Cr) [17]. The resulting surface charge means the particle is now a charge dipole, where one side of the particle becomes more negatively charged than the other side. This induced charge on the surface of the particle also affects the surrounding fluid, causing counter-ions in the fluid to accumulate near the particle's charged surface. This creates a thin layer of charged fluid surrounding the particle, known as the Debye layer. The alternating

Janus particles are special types of nano- or micro-particles whose surfaces have two or more distinct physical properties. In our case, as we can see in Figure 9, we have a Janus particle with two hemispherical surfaces A and B, with surface A coated with Chromium (Cr) and surface B coated with Gold (Au), the particle has a diameter of 2 µm.

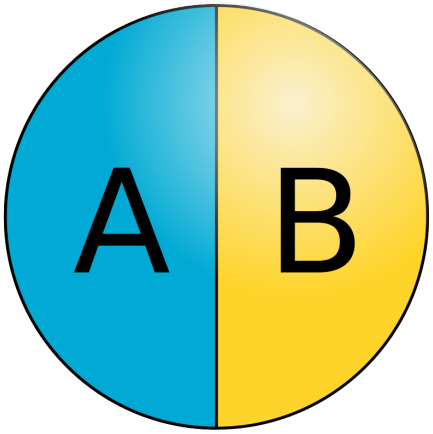

Figure 9: Schematic of a basic Janus particle with two different sides A and B (credits: Wikipedia).

electric field also acts on these charges within the Debye layer, causing the fluid to move. As the fluid moves, it also pushes the particle forwards.

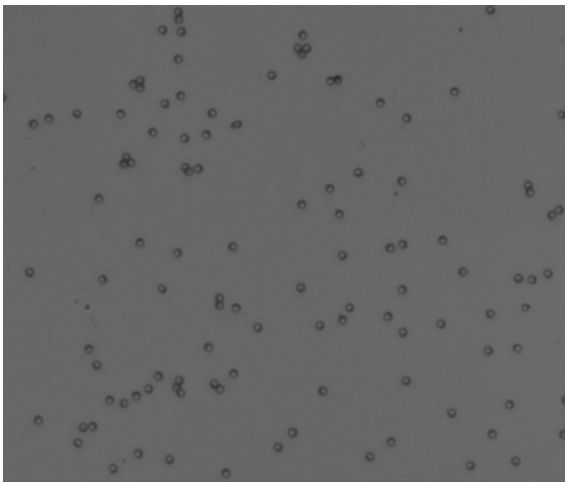

Figure 10: Bright field image of 2 µm particles partially coated with gold, from our collaborator Federico Paratore. The faces of the particles are distinguishable as being more bright or dark.

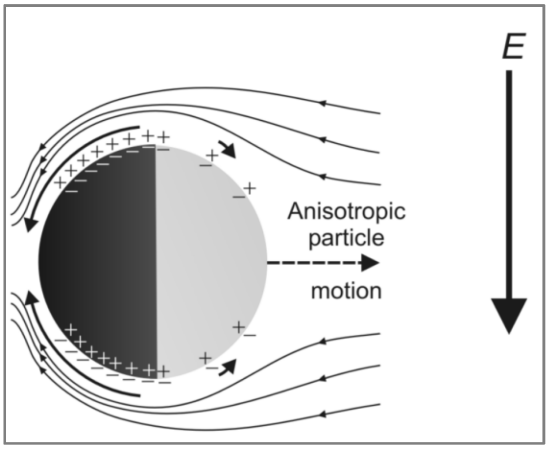

Figure 11: Schematic of a Janus particle in AC electric field.

The key advantage of utilizing electric fields lies in the degree of external control over the colloid driving mechanism, as we can fine-tune the propulsion by changing the field properties: voltage, frequency, and waveform. This permits the evaluation of the relative roles of short-range particle-particle interactions and long-range hydrodynamics underlying collective motion. Thus, a single experimental setup can probe the transition between modes and phases, including active and passive, random walk to collective motion, and assembly [18].

To obtain data from the experiments, it is sufficient to use high-speed cameras on a standard bright field microscope. As we can see in Figure 10, the resolution is sufficient to distinguish individual particles, since

there are many pixels per particle on the image. Performing standard image analysis [10], our collaborator identified on every image individual particles, their positions (x, y) and linked them over time. The total observed space in this experiment was 2560x2160 pixels. Since a pixel corresponds to 0.11 μm, this means the total observed space was 66908.16 μm^2. Note, that in this benchmark scenario, the density of the particles is sufficiently low that linking particle trajectories is relatively error-free.

The experiment yielded two sets of data corresponding to two different conditions: one where an electric field was applied with a peak-to-peak voltage (Vpp) of 10 volts, and another where no electric field was applied (Vpp=0 volts).

Each set of data included the following information for each particle:

- Particle ID: Each particle was assigned a unique identifier number.
- Time steps: The points in time at which each measurement was taken, expressed in milliseconds (ms).
- Position X: The horizontal position of each particle in the image, expressed in pixels.
- Position Y: The vertical position of each particle in the image, expressed in pixels.

These data sets, which were distinct and raw in form, required me to code some preliminary processing before the applying the different analysis methods.

5 Passive Diffusion

In the case where the peak-to-peak voltage (Vpp) is 0 volts, there is no electric field present, which means that the electric dipole is not induced. As a result, the self-propulsion behavior does not appear in the colloids. However, the colloids still exhibit motion due to passive, thermal, diffusion.

The data for the particles under Vpp=0 obtained from this experiment was provided in a MATLAB file, which I converted to a CSV file for processing in Python. I observed that the difference between time steps, denoted as dt, had a margin of difference of ± 2 ms, meaning that the time steps where not exactly uniform. The data was ordered by particle ID, for each particle ID, there were different time steps, and for each time step, the location of that particle in the observed space was provided. These factors characterizing the structure of the data needed to be taken into consideration as they played a significant role in the structure of my code and determined the specific operations I needed to perform on the data. After some data reformatting, I was then able to analyze particle trajectories with the two methods.

5.1 Mean square Displacement method

The *mean squared displacement* (MSD) is a measure of the deviation of the position of a particle with respect to a reference position over time, as shown in equation 5 :

$$\mathrm{MSD}(t) = \langle \Delta x^2(t) \rangle = \langle (x(t) - x(0))^2 \rangle \tag{9}$$

To extract as much information as possible from the data using the Mean Squared Displacement (MSD), we varied the initial time (t_0) and adjusted the code to sweep for different time intervals (t). This process is illustrated with Equation 10 more clearly in an example sketch with 3 different time intervals t in Figure12.

We then took the average over both the time steps and the number of particles to calculate our MSD, the code used to perform these calculations is shown in Figure13.

$$c \tag{10}$$

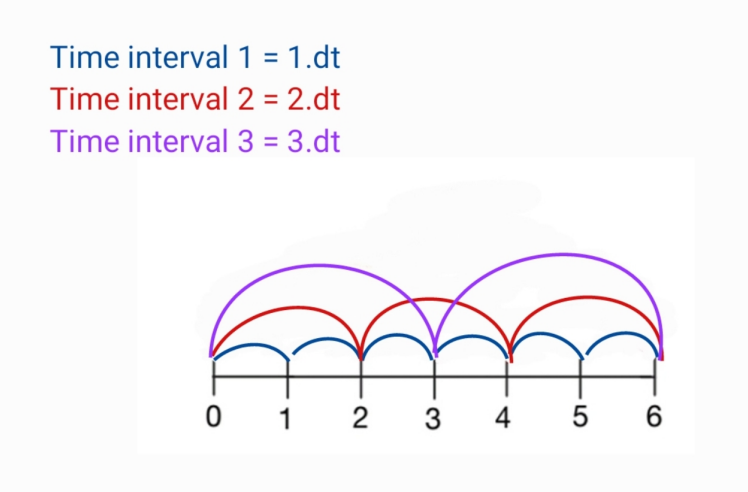

Figure 12: Sketch of 3 different time intervals.

The code I developed calls 2 functions. First, the `compute_r` function that initializes, calculates and finally stores the squared displacement of each particle, according to equation10. It then averages these squared displacements over initial times, in the same manner as we represented for 3 different time intervals in Figure12. Second, `computational_msd` takes the mean squared displacements calculated by `compute_r` and averages them over all particles to obtain the overall mean squared displacement for each time interval.

The MSD is directly proportional to the diffusion coefficient, extending to the 2D case compared to Equation 1,

$$\text{MSD}(t) = 4Dt. \tag{11}$$

My goal was to infer the diffusion coefficient D from the data. To do so, I plotted the calculated MSD in log-log scale so I could distinguish between different modes of diffusion, that will be characterized by their slopes in log-log scale.

I then fitted Equation 11 to my data. This was not an obvious task. I used a least squares fitting technique (Python package lmfit) which minimizes the following "distance" to the data

$$S = \sum_{i=1}^{N} (y_i - f(x_i))^2 \tag{12}$$

where:

- y_i are the observed data points,
- $f(x_i)$ are the predicted data points from the model function $f(x)$, where $f(x) = 4Dx$,
- N is the number of data points.

The least squares technique approximates the value of D, by minimizing the sum of the squares of the residuals S. The technique's accuracy and results can be affected by numerous factors. One of them is the noise on our data that becomes more apparent at longer time intervals, since for longer time intervals we have fewer statistics to average over. In the log scale, however, there are many more data points at long times, biasing the fit to accurately reproduce these long timescales. However, the least squares technique attempts to fit these points, regardless of those factors.

```python
#computing
def compute_r(x, y, num_particles, num_steps):
    rsquared = np.zeros((num_particles, maxDuration))
    #locating the data
    for ipart in range(1, num_particles):

        # figure out which indices correspond to particle ipart
        indices = np.where(data['particle_id'] == ipart)[0]
        xpart= x.iloc[indices].values
        ypart = y.iloc[indices].values
        indices = []
        for delta in range(1, maxDuration):
            nt0 = 0
            for t0 in range(0, maxDuration - delta ):
                #print(t0)
                displacement_x0 = xpart[t0]
                displacement_x = xpart[t0 + delta]
                displacement_y0 = ypart[t0]
                displacement_y = ypart[t0 + delta]
                rsquared[ipart,delta] += (((displacement_x - displacement_x0) ** 2)+((
displacement_y- displacement_y0) ** 2))

                nt0 += 1
            rsquared[ipart, delta]  /= (nt0)
    return rsquared
rseq = compute_r(x, y, num_particles, num_steps)
def computational_msd(rseq, maxDuration,num_particles):
    computational_displacement = np.zeros(maxDuration)
    for t in range(1,maxDuration):
        computational_displacement[t] = np.mean(rseq[:, t])
    return computational_displacement
msd = computational_msd(rseq, maxDuration,num_particles)
```

Figure 13: Snippet of the Python code I implemented to analyze the main square displacement in the case of passive diffusion.

Therefore, I tried different strategies to minimize the apparent error and get the best result possible. One of the strategies is to apply a set of normalized weights w_i for each data point i. In descending order, each weight is inversely proportional to its index in the range:

$$w_i = \frac{\text{(maxduration-i)}}{\text{the total number of information}}. \tag{13}$$

The total number of information refers to the number of data available, and we can write

$$\text{the total number of information} = \text{maxDuration}.\frac{\text{maxDuration} + 1}{2} \tag{14}$$

The residual is then modified to

$$S = \sum_{i=1}^{N} w_i(y_i - f(x_i))^2. \tag{15}$$

I present the results of the MSD obtained from the data and corresponding fit in Figure 14. Notice that in Figure 14, the MSD clearly increases with time as t^1. I obtained the diffusion coefficient from the fitting report. Since 1 pixel = 0.11 μm, and as $dt = 34.6\,ms = 0.346\,s$, the diffusion coefficient I obtained is $D = 0.12688\ \mu\text{m}^2/\text{s}$.

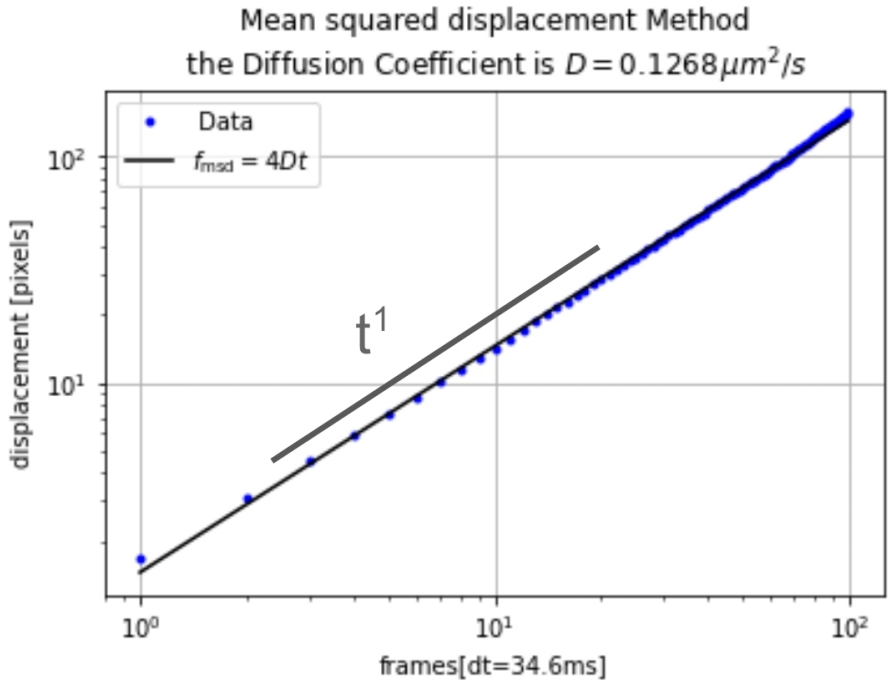

Figure 14: Mean square displacement with time obtained from passive diffusion experimental data.

According to Stokes law [4], the friction coefficient γ is represented by equation 16 :

$$\gamma = 6\pi\eta R, \tag{16}$$

where, in the experiment, η is the viscosity of water at $20°C$, and R is the radius of the colloid and is equal to $R = 1$ μm .

From equation 6 and equation 16, we obtain the prediction for the diffusion coefficient as

$$D_0 = \frac{K_B T}{6\pi\eta R} \tag{17}$$

After numerical calculation, we obtain the value of the diffusion coefficient value is $D_0 = 0.214$ μm^2/s.

The values of diffusion obtained from the mean square displacement method ($D = 0.12688$ μm^2/s) is smaller than the one we obtain from the theoretical prediction ($D_0 = 0.214$ μm^2/s). This difference is approximately equal to a factor 2, as $D_0 \approx 2D$, and is due to friction effects. Indeed, the friction coefficient γ is different depending on where the particle is with respect to the wall. When it is closer to the bottom wall, the hydrodynamic friction γ increases and slows down the particle [19].

The height h above the bottom wall[20] determines the value of the diffusion coefficient parallel to the wall and can be approximated for sufficiently small wall-particle separations by Equation 18

$$D_{\parallel}(h) = \frac{D_0}{Y(h/a)} \tag{18}$$

where

$$Y(h/R) \simeq 0.95 - \frac{8}{15}\log(h/R) \tag{19}$$

and $R = 1\mu m$ is the particle radius and $D_0 = \frac{k_B T}{6\pi\eta R} = 0.21\mu m^2/s$ Therefore we obtain the height to be :

$$h = R\exp\left[\frac{15(Y - 0.95)}{-8}\right] = 165nm \tag{20}$$

5.2 Countoscope method

The Countoscope method is a method that uses the particle number fluctuation modulation in finite spaces or subspaces and fits it with a mathematical model. This method divides the observed space into subspaces and counts how many particles are in each subspace at different time steps $N(t)$. Then, it analyzes the statistical properties of this number in time.

In the coding process, similar to the logic I applied in the mean square displacement method as shown in figure 12, I will vary initial times t_0 and average over the total number of initial times.

$$\langle \Delta N^2(t) \rangle = \langle (N_i(t+t_0) - N_i(t_0))^2 \rangle_{i,t_0} \tag{21}$$

where $\langle \rangle_{i,t_0}$ denotes the average over the subspace ID i and the initial time t_0.

The code I developed calls 2 functions. First, the `fluctuation_diff` function calculates the mean squared fluctuation of the number of particles in different subspaces for increasing time steps and averages on initial times. Second, the `computational_msd` function averages over all subspace IDs, as we can see in Figure 14.

```
16
17
18 def fluctuation_diff(x, y, section_lengthx,section_lengthy, num_steps, number_lignes,
   number_sections,num_particles_insection_data):
19     fluctuation_diff = np.zeros((number_sections, num_steps))
20     for delta in range(1,num_steps):
21         nt0 = 0
22         for t0 in range (0, num_steps-delta):
23             fluctuation_t0 = num_particles_insection_data[:,t0]
24             fluctuation_t = num_particles_insection_data[:,t0 + delta]
25             fluctuation_diff[:,delta] += (fluctuation_t - fluctuation_t0)**2
26             nt0 += 1
27         fluctuation_diff[:,delta] /= nt0
28     return fluctuation_diff
29
30 def computational_msd(diff, num_steps):
31     computational_displacement = np.zeros(num_steps)
32     for t in range(num_steps):
33         computational_displacement[t] = np.mean(diff[:, t])
34     return computational_displacement
```

Figure 15: Snippet of the Python code I implemented to analyze number particle fluctuations in all subspaces

To obtain the diffusion coefficient from investigating number fluctuations, I then used a mathematical framework based on describing fluctuating density fields, that examines the particle number correlations $C_N(t) = \langle N(t)N(0) \rangle - \langle N \rangle^2$. The correlation quantifies the statistical relationship between the number of particles in each region of a system at different points in time. The equation for $C_N(t)$ [13] for passive particles in a cubic box of size L_{obs} is given by

$$C_N(t) = \langle N \rangle \left[f_N \left(\frac{4Dt}{L_{\text{obs}}^2} \right) \right]^2 \tag{22}$$

where $f_N(x)$ is defined as

$$f_N(x) = \frac{x}{\pi} e^{-1/x} + \text{erf}\left(\sqrt{\frac{1}{x}} \right).$$

For the experiments investigated, I have rectangular box observations instead of squares or cubes, with

dimension $L_x \times L_y$. In that case, the particle number correlation is:

$$C_N(t) = \langle N \rangle \left[f_N \left(\frac{4Dt}{L_x^2} \right) \right] \left[f_N \left(\frac{4Dt}{L_y^2} \right) \right]$$

Finally, I obtain the number fluctuations in time noticing that 23 :

$$\Delta N^2(t) = 2\langle N \rangle - 2C_N(t) \tag{23}$$

I fitted Equation 23 to the fluctuating particle number $\langle \Delta N^2(t) \rangle$ obtained from experimental data with the least squares technique. The goal is to find the diffusion coefficient D. For similar reasons mentioned while performing the fit of the mean squared displacement, I needed to apply strategies to make the fitting more accurate. Notice that, unlike the MSD which increases continuously in time, here the number fluctuations increase in the log-log space but then eventually plateau, as fluctuations of particle number are bounded. This means the fitting procedure is likely influenced by slightly different factors. Applying the weights I discussed in the mean squared displacement method didn't lead to much change as the fitting technique tends to focus on fitting the later part of the plot. I thus tried an alternative strategy, where weights take into account the fact that most variations are seen in log-log space and used Equation 24 for the weights

$$w_i = \log(x_{i+1}) - \log(x_i) \tag{24}$$

This weight tells the code to treat the aspect of the curve itself (in log log) and not so much individual points, as it accounts for the fact that the data points are not linearly spaced in log scale. The weight for each point is determined by the spacing between the particular point and the next one. Another strategy I used was to fit the data for $\log(x)$ and $\log(y)$, instead of x and y, which did help immensely make the fit more accurate.

I represent the data in log-log scale in Figure 16.

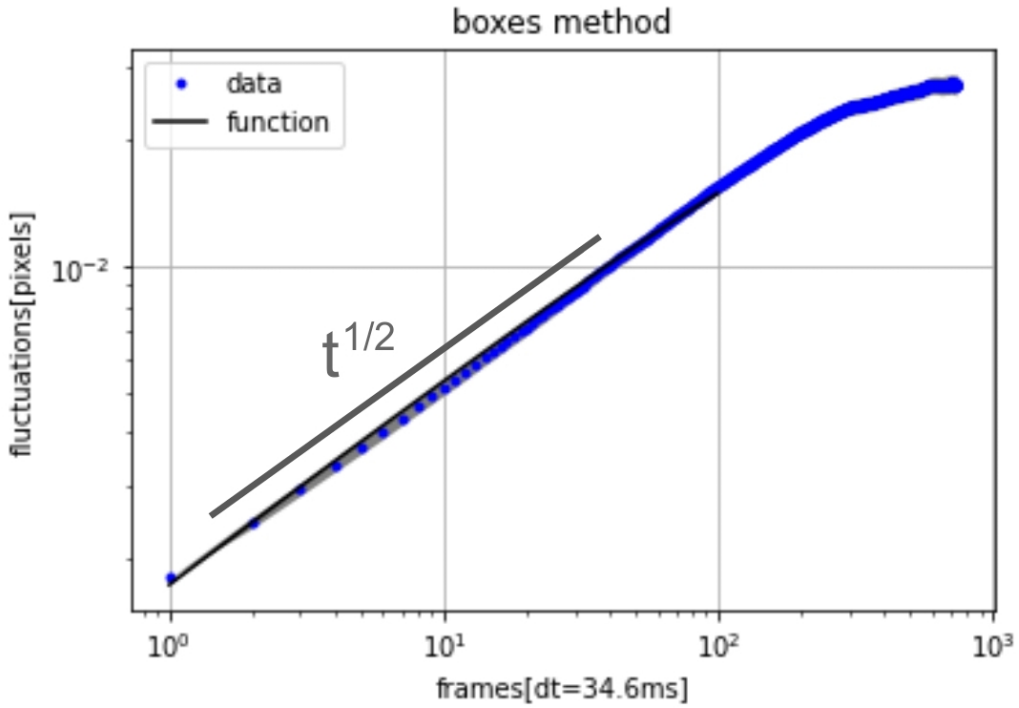

Figure 16: Plot of particle number fluctuations with time using the countoscope method and a box of size of $138.24\mu m^2$.

I plotted also the standard deviation (STD) between different boxes for $\Delta N^2(t)$ that I present in Figure16 in light gray. The standard deviation shows how much the particle number fluctuation for every subspace is spread out from the average value.

I observed that the particle number fluctuations grow at early times as $t^{1/2}$, and keep increasing until eventually they plateau. This means that the number of particles in the box has reached a steady state where the number of particles entering the subspace is approximately equal to the number of particles leaving the subspace.

The Diffusion coefficient from the fitting process I obtained is $D = 0.13\ \mu\text{m}^2/\text{s}$. It is approximately the same value I obtained using the mean square displacement method. We can see that clearly see it in Figure 5.2, which shows both a plot of the particle number fluctuations with the best-fitted value of $D = 0.13\ \mu\text{m}^2/\text{s}$ and another plot of the particle number fluctuation represented with the value obtained from the MSD fitting $D = 0.12\ \mu\text{m}^2/\text{s}$.

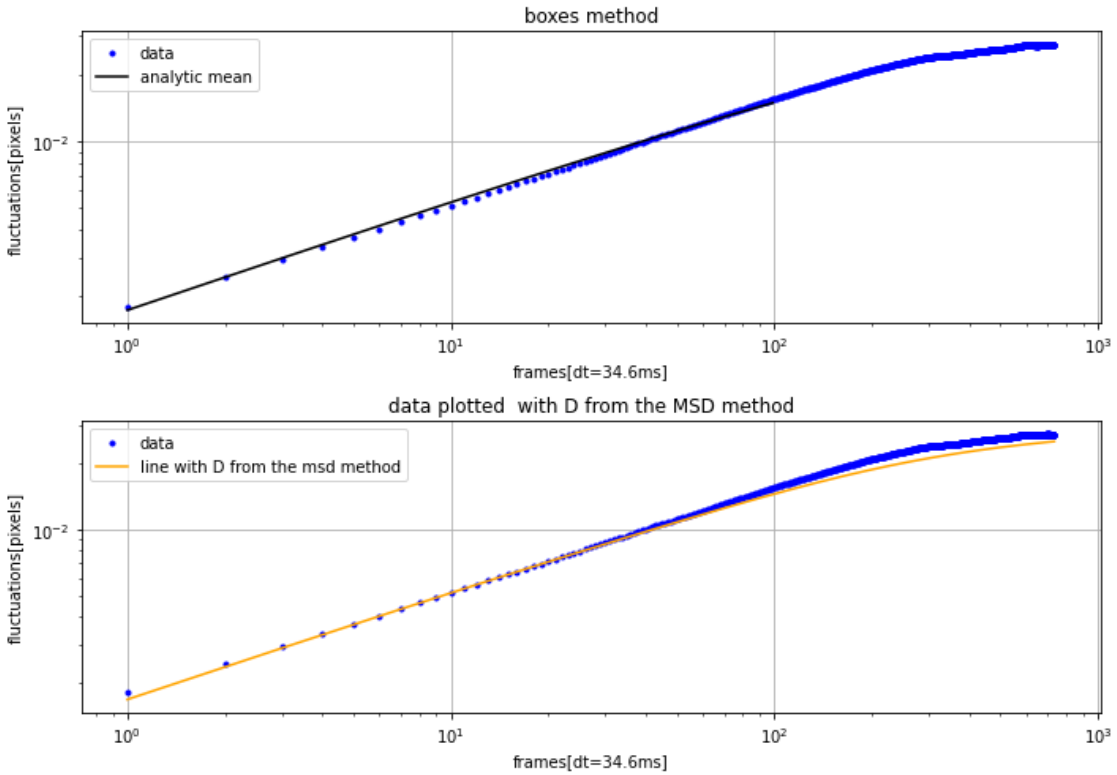

Figure 17: Two plots of the particle number fluctuation. (Top) Fitted Using a varying Diffusion Coefficient (D) and best fit presented and (bottom) with a Diffusion Coefficient obtained by fit to the Mean Squared Displacement (MSD)

We finally attempt to understand a bit better the behavior of the particle number fluctuations. By varying the size of the subspaces, I obtained varying behavior, see Fig 18 (inset). To make sense of this varying behavior, the theory Equation 23 suggests rescaling the amplitude of the fluctuations by the average number of particles in the box $\langle N \rangle$ and time by the time to diffuse across a box $\tau_{\text{Diff}} = L^2/D$. $L^2 = L_x L_y$ as L_x and L_y are not the same length. As presented in Figure 18, this rescaling yields a remarkable collapse of the experimental data, suggesting that the theory robustly describes these different situations. The important timescale is therefore the time to diffuse across the box.

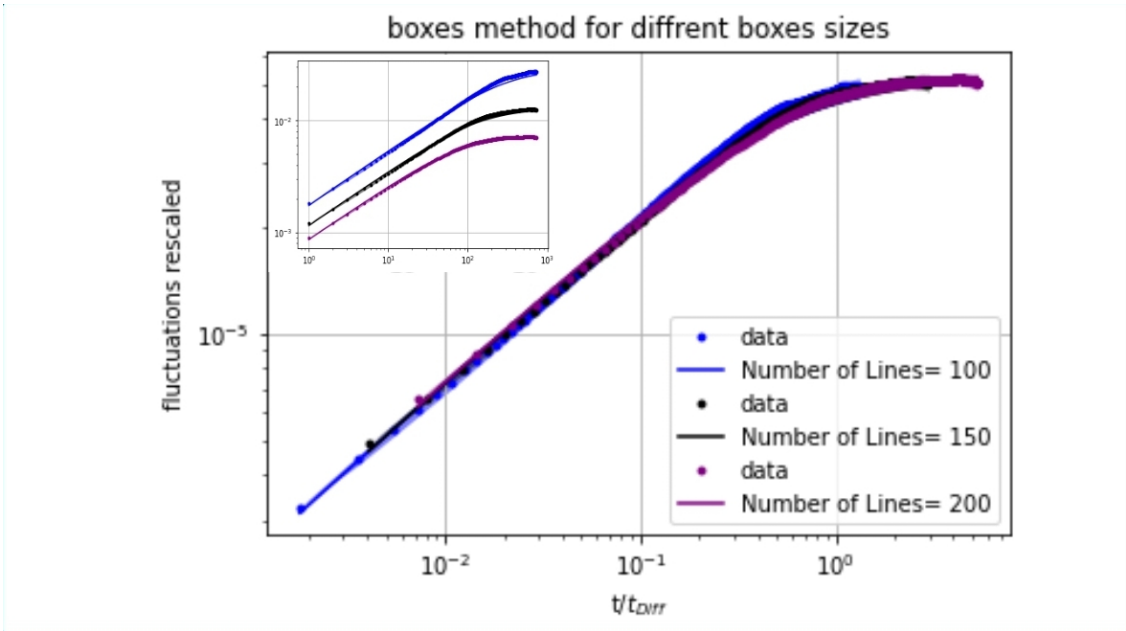

Figure 18: Plot of particle number fluctuations for different subspace sizes in re-scaled units and (inset) before rescaling.

6 Active Diffusion

In the case where the peak-to-peak voltage (Vpp) is 10 volts, an electric field is present, which induces an electric dipole. As a result, the self-propulsion behavior appears in the colloids, making the colloids behave like active particles.

The experimental data I analyzed was largely similar to the one with Vpp= 0 volts, with time step intervals dt around 34.6 ± 2 ms. However, there was a "glitch" in the data where dt suddenly went from about 35 ms to 500 ms in a small section of the data. This discrepancy forced me to remove approximately 30% of the corrupted data to maintain data validity. Nonetheless, it was possible to obtain results from this limited amount of data.

6.1 Mean squared displacement method

In the process of plotting the mean square displacement for the case of active diffusion, I noticed a strange behavior in the slope plotted, the first intuition I got was that maybe there's some sort of a systematic displacement or movement of particles or drift, due to an experimental factor like a small tilt in the measurement apparatus, which made the particle drift to this direction or the other. Therefore, I applied a drift correction in my code. the approach taken was by calculating the average drift velocities in the x and y directions and then correcting the term in the main code.

the velocities are $v_x = \langle \frac{dx}{dt} \rangle$ and $v_y = \langle \frac{dy}{dt} \rangle$, the corrected mean square displacement became $\langle (x(t+t_0) - x(t_0) - v_x \cdot dt)^2 \rangle + \langle (y(t+t_0) - y(t_0) - v_y \cdot dt)^2 \rangle$. Upon analyzing the results, I found that the drift velocities v_x and v_y were very small, suggesting that their influence on the overall motion is negligible.

the strange behavior of the slope of data plotted may be explained if we try to fit the theoretical equation of the mean square displacement equation in the case of active Diffusion, as we can in Equation 25.

$$\mathrm{MSD}(t) = 2Dt + {v_0}^2 t^2 \tag{25}$$

where v_0 stands for the self-propulsion speed of active particles. We plotted the data and fitted it with equation 25. As can be seen in Figure 19, the slope of the mean square displacement increases quadratically with time (t^2). Fitting the equation yields a self-propulsion speed $v_0 \approx 3.58 \mu m/s$. This high velocity suggests that under the given conditions ($V_{pp} = 10$), the particles are moving so rapidly that the mean square displacement is primarily dominated by the ballistic phase.

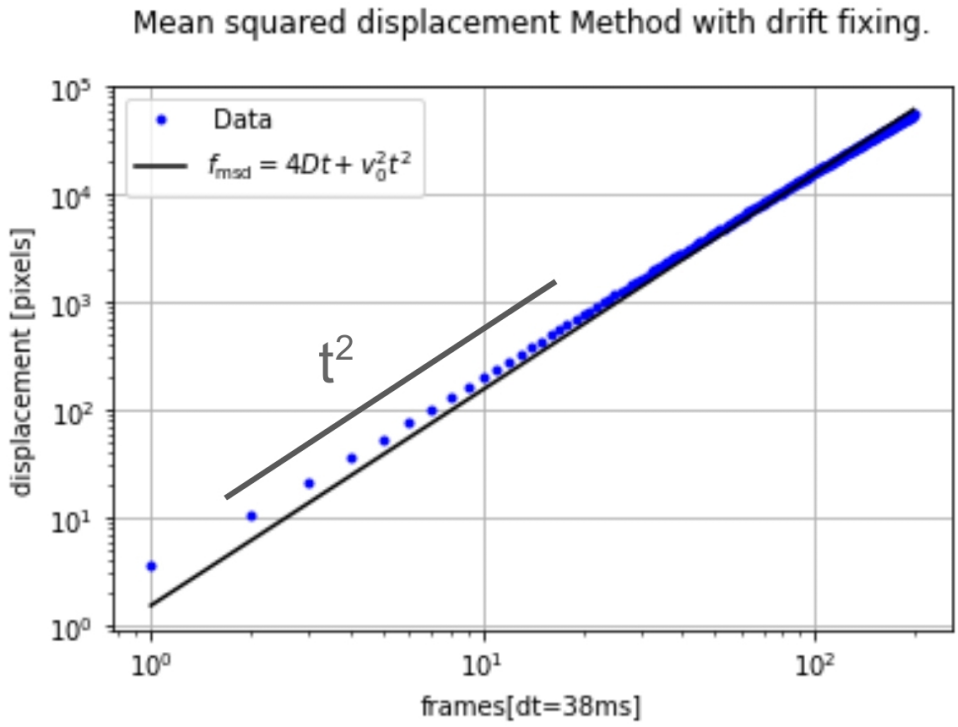

Figure 19: Mean square displacement with time obtained from active diffusion experimental data with drift fixing

6.2 countscope Method

I plotted the particle number fluctuation $\Delta N^2(t)$ that I present in Figure 20, and as we can observe the slope grows with time (t^1).

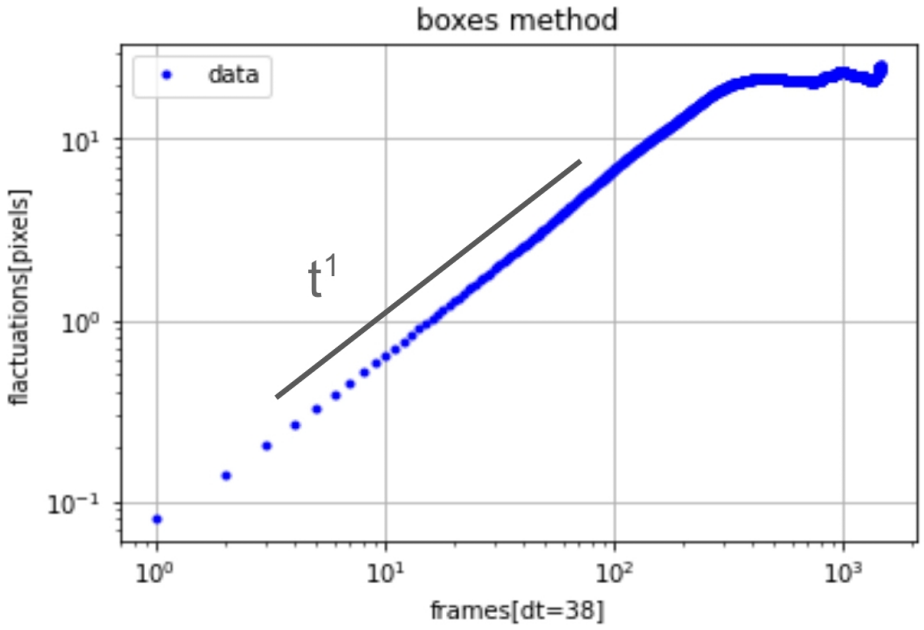

Figure 20: particle number fluctuation $\Delta N^2(t)$ with time obtained from active diffusion experimental data.

7 discussion and conclusion

During this Internship, I have investigated diffusion properties of micron-scaled Janus particles, in the absence of an electrical field ($V_{pp} = 0\ V$), and in its presence ($V_{pp} = 10\ V$). In the absence of an electrical field, the particles diffuse passively, or thermally, and In the presence of the field, under (($V_{pp} = 10\ V$) particles show a ballistic regime.

Component	MSD method	Countoscope method
Passive case		
Value of the diffusion coefficient	$D = 0.12\mu m^2/s$	$D = 0.13\mu m^2/s$
Behavior of the slope at short times	t^1	$t^{1/2}$
Active case		
Value of the drift	$v_0 = 3.57\mu m/s$	
Behavior of the slope at short times	t^2	t^1

Table 1: results of the study

The countscope method has proven to be as effective as the traditional mean square displacement analysis, not only has it successfully performed tasks that the MSD method does, but it also transcends many of the limitations associated with MSD, the countscope method is largely uncharted territory in the realm of investigating particle dynamics with a lot of capabilities are yet to be fully tested and understood.

There are still several intriguing questions to answer in this project. How does the system evolve from a predominantly diffusive to a ballistic regime? What characteristics define these transitions, and can we predict them based on the voltage? The immediate next step would be to explore the intermediate regimes of voltage, Subsequent goals include extending the mathematical formalism of the countoscope method to accommodate active diffusion, and applying this method on other types of particles, such as swimming bacteria, or in much more dense systems.

References

[1] de Jesús Astacio, L. M., Prabhakara, K. H., Li, Z., Mickalide, H. & Kuehn, S. Closed microbial communities self-organize to persistently cycle carbon. *Proceedings of the National Academy of Sciences* **118** (2021).

[2] Bhattacharjee, T. & Datta, S. S. Bacterial hopping and trapping in porous media. *Nature communications* **10**, 2075 (2019).

[3] Dentz, M., Creppy, A., Douarche, C., Clément, E. & Auradou, H. Dispersion of motile bacteria in a porous medium. *Journal of Fluid Mechanics* **946**, A33 (2022).

[4] Bian, X., Kim, C. & Karniadakis, G. E. 111 years of brownian motion. *Soft Matter* **12**, 6331–6346 (2016).

[5] Bechinger, C. *et al.* Active particles in complex and crowded environments. *Reviews of Modern Physics* **88**, 045006 (2016).

[6] Nirody, J. A., Sun, Y.-R. & Lo, C.-J. The biophysicist's guide to the bacterial flagellar motor. *Advances in Physics: X* **2**, 324–343 (2017).

[7] Berg, H. C. *E. coli in Motion* (Springer, 2004).

[8] Solon, A. P., Cates, M. E. & Tailleur, J. Active brownian particles and run-and-tumble particles: A comparative study. *The European Physical Journal Special Topics* **224**, 1231–1262 (2015).

[9] Berthier, L. & Kurchan, J. Lectures on non-equilibrium active systems. *arXiv preprint arXiv:1906.04039* (2019).

[10] Crocker, J. C. & Grier, D. G. Methods of digital video microscopy for colloidal studies. *Journal of colloid and interface science* **179**, 298–310 (1996).

[11] Bricard, A., Caussin, J.-B., Desreumaux, N., Dauchot, O. & Bartolo, D. Emergence of macroscopic directed motion in populations of motile colloids. *Nature* **503**, 95–98 (2013).

[12] Wioland, H., Lushi, E. & Goldstein, R. E. Directed collective motion of bacteria under channel confinement. *New Journal of Physics* **18**, 075002 (2016).

[13] Minh, T. H. N., Rotenberg, B., Marbach, S. *et al.* Ionic fluctuations in finite volumes: fractional noise and hyperuniformity. *Faraday Discussions* (2023).

[14] Weiss, C. J. Introduction to stochastic simulations for chemical and physical processes: Principles and applications. *Journal of Chemical Education* **94**, 1904–1910 (2017).

[15] Marbach, S. & Holmes-Cerfon, M. Mass changes the diffusion coefficient of particles with ligand-receptor contacts in the overdamped limit. *Physical Review Letters* **129**, 048003 (2022).

[16] Gangwal, S., Cayre, O. J., Bazant, M. Z. & Velev, O. D. Induced-charge electrophoresis of metallodielectric particles. *Physical review letters* **100**, 058302 (2008).

[17] Schwerdtfeger, P. & Nagle, J. K. 2018 table of static dipole polarizabilities of the neutral elements in the periodic table. *Molecular Physics* **117**, 1200–1225 (2019).

[18] Mauleon-Amieva, A. *et al.* Competing active and passive interactions drive amoebalike crystallites and ordered bands in active colloids. *Physical Review E* **102**, 032609 (2020).

[19] Goldman, A. J., Cox, R. G. & Brenner, H. Slow viscous motion of a sphere parallel to a plane wall—i motion through a quiescent fluid. *Chemical engineering science* **22**, 637–651 (1967).

[20] Sprinkle, B., Van Der Wee, E. B., Luo, Y., Driscoll, M. M. & Donev, A. Driven dynamics in dense suspensions of microrollers. *Soft Matter* **16**, 7982–8001 (2020).