université de BORDEAUX

université de BORDEAUX

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR

DE L'UNIVERSITÉ DE BORDEAUX

ECOLE DOCTORALE SCIENCES PHYSIQUES ET DE L'INGÉNIEUR

LASERS, MATIÈRE, NANOSCIENCES

Par **Maxime Lavaud**

Confined Brownian Motion

Sous la direction de : **Thomas Salez**
Co-encadrant : **Yacine Amarouchene**

Soutenue le vendredi 26 novembre 2021

Membres du jury :

M. Christophe Ybert	Directeur de Recherche	Université de Lyon	Rapporteur
M. Alois Würger	Professeur	Université de Bordeaux	Examinateur
M. Lionel Bureau	Directeur de Recherche	Université Grenoble Alpes	Rapporteur
M. Frédéric Restagno	Directeur de Recherche	Université Paris Sud	Examinateur
M. Thomas Salez	Chargé de Recherche	Université de Bordeaux	Directeur
M. Yacine Amarouchene	Chargé de Recherche	Université de Bordeaux	Co-encadrement

Abstract

In this manuscript, I present the work done during my PhD on confined Brownian motion. Brownian motion is the erratic movement of microscopic particles when immersed into a fluid. Thanks to Einstein and followers, it is generally possible to describe Brownian motion using simple equations. However, in the last two decades a scientific revolution has taken place with the advent of miniaturization and in particular microfluidics, thanks to which we are now able to create complex networks of pipes at the micrometer scale. Microfluidics makes it possible to sort particles, such as drops, cells or bubbles, but also to distribute drugs in cells and observe their effect on thousands of them. Regarding Brownian motion, it has been observed that once confined near a wall, a particle moves much slower due to non-slip boundary conditions at the wall. The mobility is thus modified by confinement-induced effects.

My thesis work consists in experimentally measuring, analyzing and modeling the movement of micrometric colloids diffusing near a wall. To track the motion of confined Brownian microparticles, I use Lorenz-Mie holography. The Lorenz-Mie framework allows me to record the thermally-induced tridimensional trajectories of individual microparticles, within salty aqueous solutions, in the vicinity of a rigid wall, and in the presence of surface charge with a nanometric resolution. From the recorded trajectory, I construct the time-dependent position and displacement probability density functions, and study the non-Gaussian character of the latter which is a direct signature of the hindered mobility near the wall. Based on these distributions, I implement a novel, robust and self-calibrated multifitting method, allowing for the thermal-noise-limited inference of diffusion coefficients spatially-resolved at the nanoscale, equilibrium potentials, and forces at the femtonewton resolution. Moreover, I use this novel tool in order to infer non-conservative forces and study long-time higher-order statistical properties. Our goal for the future is to use this novel tool for the in-depth study of various problems relevant to nanophysics and microbiology. In chapters 1 and 2, I present a general introduction of the subject (including en French translation). In chapter 3, I present the history of Brownian motion, starting from how it has been discovered to its numerical simulation. In chapter 4, I present state-of-the-art methods that permit tracking single colloids. In particular, I detail how we use the Lorenz-Mie framework. Then, I show the viability of this framework by studying a well-known problem: the sedimentation of a colloid. In chapter 5, I present the confined Brownian motion theory and its experimental study. In particular, I explain the implementation of a novel, robust and self-calibrated multifitting method. In chapter 6, I present other applications the newly developed method. Finally, I conclude the manuscript in chapters 7 and 8 (including en French translation).

Résumé

Dans ce manuscrit, je présente le travail effectué pendant mon doctorat sur le mouvement brownien confiné. Le mouvement brownien est le mouvement erratique de particules microscopiques lorsqu'elles sont immergées dans un fluide. Grâce à Einstein et ses successeurs, il est généralement possible de décrire le mouvement brownien à l'aide d'équations simples. Cependant, au cours des deux dernières décennies, une révolution scientifique a eu lieu avec l'avènement de la miniaturisation et en particulier de la microfluidique, grâce à laquelle nous sommes désormais en mesure de créer des réseaux complexes de tuyaux à l'échelle micrométrique. La microfluidique permet de trier des particules, comme des gouttes, des cellules ou des bulles, mais aussi de distribuer des médicaments dans des cellules et d'observer leurs effets sur des milliers d'entre elles. En ce qui concerne le mouvement brownien, il a été observé qu'une fois confinée près d'une paroi, une particule se déplace beaucoup plus lentement en raison des conditions limites non glissantes à la paroi. La mobilité est donc modifiée par les effets induits par le confinement.

Mon travail de thèse consiste à mesurer, analyser et modéliser expérimentalement le mouvement de colloïdes micrométriques diffusant près d'une paroi. Pour suivre le mouvement de microparticules browniennes confinées, j'utilise l'holographie de Lorenz-Mie. Le cadre de Lorenz-Mie me permet d'enregistrer les trajectoires tridimensionnelles thermiquement induites de microparticules individuelles, dans des solutions aqueuses salées, à proximité d'une paroi rigide, et en présence d'une charge de surface avec une résolution nanométrique. À partir de la trajectoire enregistrée, je construis les fonctions de densité de probabilité de position et de déplacement en fonction du temps, et j'étudie le caractère non-gaussien de ces dernières qui est une signature directe de la modification de la mobilité à proximité de la paroi. Sur la base de ces distributions, j'implémente une nouvelle méthode de multifitting robuste et auto-calibrée, permettant l'inférence limitée par le bruit thermique des coefficients de diffusion résolus spatialement à l'échelle nanométrique, des potentiels d'équilibre et des forces à la résolution du femtonewton. De plus, j'utilise ce nouvel outil pour déduire les forces non-conservatives et étudier les propriétés statistiques d'ordre supérieur à long terme. Notre objectif pour l'avenir est d'utiliser ce nouvel outil pour l'étude approfondie de divers problèmes liés à la nanophysique et à la microbiologie. Dans les chapitres 1 et 2, je présente une introduction générale du sujet (y compris la traduction française). Dans le chapitre 3, je présente l'histoire du mouvement brownien, de sa découverte à sa simulation numérique. Dans le chapitre 4, je présente les méthodes les plus récentes qui permettent de suivre des colloïdes uniques. En particulier, je détaille comment nous utilisons le cadre de Lorenz-Mie. Ensuite, je montre la viabilité de ce cadre en étudiant un problème bien connu : la sédimentation d'un colloïde. Dans le chapitre 5, je présente la théorie du mouvement brownien confiné et son étude expérimentale. En

particulier, j'explique l'implémentation d'une nouvelle méthode de multifitting, robuste et auto-calibrée. Dans le chapitre 6, je présente d'autres applications de la méthode nouvellement développée. Enfin, je conclus le manuscrit dans les chapitres 7 et 8 (y compris la traduction française).

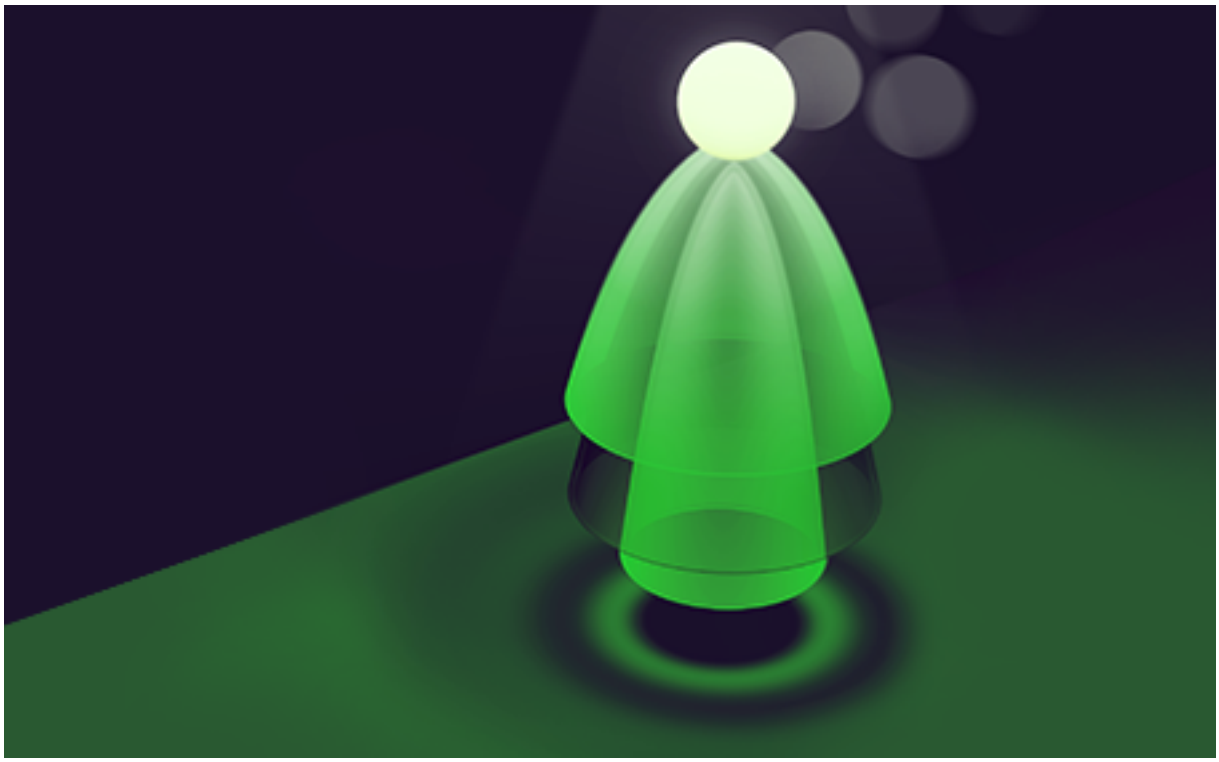


Figure 0.0.1: Artist's view of the Lorenz-Mie method. A polystyrene ball (in white) is illuminated from above. The interference pattern is indicated by the concentric rings (Credit: Pierre Savary).
Vue d'artiste de la méthode de Lorenz-Mie. Une bille de polystyrène (en blanc) est éclairée par le haut. La figure d'interférences est indiquée par les anneaux concentriques (Crédit : Pierre Savary).

Table of Contents

List of Figures

List of Abbreviations

EHD	Elastohydrodynamic
fps	Frames per second
MRSE	Mean Relative Squared Error
MSD	Mean Squared Displacement
PDF	Probability Density Function
PDMS	Polydimethylsiloxane
RICM	Reflection Interference Contrast Microscopy
ROI	Region Of Interest
SDE	Stochastic Differential Equations
SFI	Surface Force Inference

1 Introduction

Brownian motion is a central paradigm in modern science. It has implications in fundamental physics, biology, and even finance, to name a few. By understanding that the apparent erratic motion of colloids is a direct consequence of the thermal motion of surrounding fluid molecules, pioneers like Einstein and Perrin provided decisive evidence for the existence of atoms [**einstein˙uber˙1905**, **perrin˙les˙2014**].

During the past 30 years [**whitesides˙origins˙2006**, **convery˙30˙2019**] microfluidics and the development of *lab-on-a-chip* technologies [**neuzil˙revisiting˙2012**, **ding˙surface˙2013**] became standard in chemistry, biology and medicine. Thus, at a time of miniaturization and interfacial science, and moving beyond the idealized bulk picture, it is relevant to consider the added roles of boundaries to the above context. Indeed, Brownian motion at interfaces and in confinement is a widespread practical situation in microbiology and nanofluidics. In such a case, surface effects become dominant and alter drastically the Brownian statistics, with key implications towards: i) the understanding and smart control of the interfacial dynamics of microscale entities; and ii) high-resolution measurements of surface forces at equilibrium. Interestingly, a confined colloid will exhibit non-Gaussian statistics in displacements, due to the presence of multiplicative noises induced by the hindered mobility near the wall [**felderhof˙effect˙2005**, **wang˙anomalous˙2009**, **chechkin˙brownian˙2017**]. Besides, the particle can be subjected to electrostatic or Van der Waals forces [**bouzigues˙nanofluidics˙2008**] exerted by the interface, and might experience slippage too [**joly˙probing˙2006**, **mo˙brownian˙2017**]. Previous studies have designed novel methods to measure the diffusion coefficient of confined colloids [**faucheux˙confined˙199 dufresne˙brownian˙2001**, **carbajal-tinoco˙asymmetry˙2007**, **eral˙anisotropic˙2010**, **sharma˙high-precision˙2010**, **mo˙broadband˙2015**, **matse˙test˙2017**], or to infer surface forces [**prieve˙measurement˙1999**, **banerjee˙experimental˙2005**, **sainis˙statistics˙2007**, **volpe˙influence˙2010**, **wang˙measurement˙2011**, **li˙subfemtonewton˙2019**]. However, such a statistical inference is still an experimental challenge, and a precise calibration-free method taking simultaneously into account the whole ensemble of relevant properties, over broad spatial and time ranges, is currently lacking.

In my thesis, I aimed at filling the gap identified above by implementing a novel method of statistical inference on a set of trajectories of individual microparticles recorded by holographic microscopy. In the chapter 3, I introduce Brownian motion from its discovery to its mathematical description and numerical simulation. In the chapter 4, I present our state-of-the-art particle-tracking method. In the chapter 5, I further use it experimentally to study buoyant particles that are free to evolve within salty aqueous solutions, near a rigid substrate, and in the presence of surface charges. Besides, I present an optimization

scheme to determine precisely all the physical parameters and the actual distance to the wall, at once. All together, this procedure leads to the robust calibration-free inference of the two central quantities of the problem: i) the space-dependent short-term diffusion coefficients, with a nanoscale spatial resolution; and ii) the total force experienced by the particle, at the thermal-noise limited femtonewton resolution. Finally, in the chapter 6, I use this novel tool in order to infer non-conservative forces and study long-time higher-order statistical properties.

1.1 Open access

A great part of my work depends on resources that are in open-access (OA). OA resources range from articles that can provide experimental data or the source code used for simulation, to entire frameworks to analyze data or build state-of-the-art simulations. I, want to promote open access research as it enables the fastest spread of the latest techniques around the globe by removing the cost barriers. Thus, my manuscript and source codes are uploaded on a free accessible Github repository. Along this manuscript, Github logos () can be found. These are hyperlinks to aforementioned information. The links can be found especially in the figure captions and lead to their source code, or along the text to redirect to any OA resource I am referring to. All the figures, are done using Python and Jupyter notebooks, thus providing an easy way to reproduce or reuse them. The interested reader can follow along the manuscript and change the different variables to see how physics changes. An explanation on how to setup a Python environment and employ the notebooks using Conda can be found in my repository .

Figure 1.1.1: French physicist and Nobel laureate Jean Perrin (1870-1942) with his "mega-spectroscope" at the Institut Curie in 1927.

2 Introduction française

Le mouvement brownien est un paradigme central de la science moderne. Il a des implications en physique fondamentale, en biologie et même en finance, pour n'en citer que quelques-unes. En comprenant que le mouvement erratique apparent des colloïdes est une conséquence directe du mouvement thermique des molécules du fluide environnant, des pionniers comme Einstein et Perrin ont fourni des preuves décisives de l'existence des atomes [**einstein˙uber˙1905**, **perrin˙les˙2014**].

Au cours des 30 dernières années [**whitesides˙origins˙2006**, **convery˙30˙2019**] la microfluidique et le développement des technologies *lab-on-a-chip* [**neuzil˙revisiting˙2012**, **ding˙surface˙2013**] sont devenus des standards en chimie, biologie et médecine. Ainsi, à l'heure de la miniaturisation et de la science interfaciale, et au-delà de l'image idéalisée du volume, il est pertinent de considérer le rôle supplémentaire des bords dans le contexte ci-dessus. En effet, le mouvement Brownien aux interfaces et en confinement est une situation pratique répandue en microbiologie et en nanofluidique. Dans un tel cas, les effets de surface deviennent dominants et modifient radicalement les statistiques browniennes, avec des implications clés pour : i) la compréhension et le contrôle intelligent de la dynamique interfaciale des entités à l'échelle microscopique ; et ii) les mesures à

haute résolution des forces de surface à l'équilibre. Il est intéressant de noter qu'un colloïde confiné présentera des statistiques non gaussiennes dans les déplacements, en raison de la présence de bruits multiplicatifs induits par une mobilité modifiée près de la paroi [**felderhof˙effect˙2005**, **wang˙anomalous˙2009**, **chechkin˙brownian˙2017**]. En outre, la particule peut être soumise à des forces électrostatiques ou de Van der Waals [**bouzigues˙nanofluidics˙2008**] exercées par l'interface, et peut également subir un glissement [**joly˙probing˙2006**, **mo˙brownian˙2017**]. Des études antérieures ont conçu de nouvelles méthodes pour mesurer le coefficient de diffusion des colloïdes confinés [**faucheux˙confined˙1994**, **dufresne˙brownian˙2001**, **carbajal-tinoco˙asymmetry˙2007**, **eral˙anisotropic˙2010**, **sharma˙high-precision˙2010**, **mo˙broadband˙2015**, **matse˙test˙2017**], ou pour déduire les forces de surface [**prieve˙measurement˙1999**, **banerjee˙experimental˙2005**, **sainis˙statistics˙2007**, **volpe˙influence˙2010**, **wang˙measurement˙2011**, **li˙subfemtonewton˙201**
Cependant, une telle inférence statistique reste un défi expérimental, et une méthode précise sans calibration prenant en compte simultanément l'ensemble des propriétés pertinentes, sur de larges plages spatiales et temporelles, fait actuellement défaut.

Dans ma thèse, j'ai cherché à combler le vide identifié ci-dessus en mettant en œuvre une nouvelle méthode d'inférence statistique sur un ensemble de trajectoires de microparticules individuelles enregistrées par microscopie holographique. Dans le chapitre 3, je présente le mouvement brownien depuis sa découverte jusqu'à sa description mathématique et sa simulation numérique. Dans le chapitre 4, je présente notre méthode de pointe pour le suivi des particules. Dans le chapitre 5, je l'utilise expérimentalement pour étudier les particules flottantes qui sont libres d'évoluer dans des solutions aqueuses salées, près d'un substrat rigide, et en présence de charges de surface. En outre, je présente un schéma d'optimisation pour déterminer précisément tous les paramètres physiques et la distance réelle à la paroi, en une seule fois. Dans l'ensemble, cette procédure conduit à l'inférence robuste et sans calibration des deux quantités centrales du problème : i) les coefficients de diffusion à court terme dépendant de l'espace, avec une résolution spatiale à l'échelle nanométrique ; et ii) la force totale subie par la particule, à la résolution femtonewton limitée par le bruit thermique. Enfin, dans le chapitre 6, j'utilise ce nouvel outil pour déduire les forces non-conservatives et étudier les propriétés statistiques d'ordre supérieur à long terme.

2.1 Libre Accès

Une grande partie de mon travail dépend de ressources en libre accès (OA). Les ressources en libre accès vont des articles qui peuvent fournir des données expérimentales ou le code source utilisé pour des simulations, à des *framework* entiers pour analyser les données ou

construire des simulations de pointe. Je souhaite promouvoir la recherche en libre accès, car elle permet une diffusion plus rapide des dernières techniques dans le monde entier en supprimant les obstacles financiers. Ainsi, mon manuscrit et mes codes sources sont téléversée sur un dépôt Github en libre accès. Le long de ce manuscrit, on peut trouver des logos Github (). Il s'agit d'hyperliens vers les informations susmentionnées. Les liens se trouvent notamment dans les légendes des figures et mènent à leur code source, ou le long du texte pour rediriger vers toute ressource OA à laquelle je fais référence. Toutes les figures sont réalisées à l'aide de Python et de Jupyter notebooks, ce qui permet de les reproduire ou de les réutiliser facilement. Le lecteur intéressé peut suivre le manuscrit et changer les différentes variables pour voir comment la physique change. Une explication sur la façon de configurer un environnement Python et d'utiliser les *notebooks* en utilisant Conda peut être trouvée dans mon dépôt de données .

Figure 2.1.1: Le physicien français et lauréat du prix Nobel Jean Perrin (1870-1942) avec son "méga-spectroscope" à l'Institut Curie, en 1927

3 Brownian Motion

3.1 The Brownian motion discovery

In 1827 the Scottish botanist Robert Brown published an article [**robert˙xxvii˙1828**] on his observation on the pollen of *Clarkia pulchella* with a lot of details on his reflection processes. His experiments were made to understand the flower reproduction, but, as he was looking through the microscope he observed some minute particles ejected from the pollen grains. At first, he thought the goal of this agitation was to test the presence of a male organ. To test this theory, he extended his observations to Mosses and *Equiseta*, which were drying for a hundred years. However, the fact that this peculiar motion was still observable made him invalidate his theory. Interestingly, each time that he encountered a material that he could reduce to a fine enough powder to be suspended in water, he observed the same type of motion, although, he never understood its particle's movement.

The difficulty at this time to observe and capture such a movement made the study of what we call contemporarily Brownian motion difficult and the first theoretical work was done by Louis Bachelier in his PhD thesis "The theory of speculation," where he described a stochastic analysis of the stock and option market. Nowadays, the mathematical description of random movement is still used in the modern financial industry.

It is finally in 1905 that Albert Einstein theoretically state that "bodies of microscopically visible size suspended in a liquid will perform movements of such a magnitude that they can be easily observed in a microscope" [**einstein˙uber˙1905**]. A remark to make here is that in 1948 Einstein wrote a letter to one of his friends where he stated having deduced the Brownian motion "from mechanics, without knowing that anyone had already observed anything of the kind" [**peter˙brownian˙nodate**].

It is in 1908 that Jean Perrin published his experimental work on Brownian motion. That way he could measure the Avogadro number and prove the kinetic theory that Einstein developed. I would also cite Chaudesaigues and Dabrowski, who helped Perrin to track the particles manually, half-minutes by half-minutes, for more than 3000 displacements (25 hours) and several particles. This impressive and daunting work is highly detailed in *"Mouvement brownien et molécules"* [**perrin˙mouvement˙1910**]. This is partly due to the results of this work that Perrin received the Nobel award in 1926.

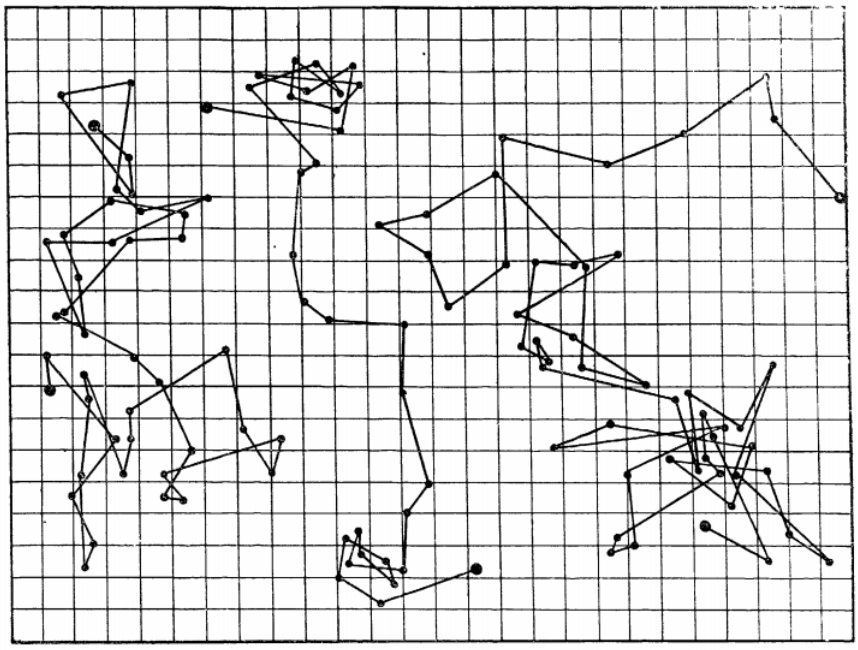


Figure 3.1.1: Brownian motion of 1 μm particles in water tracked manually by Jean Perrin and his colleagues. The points are spaced in time by 30 seconds, and 16 divisions represent 50 μm.

3.2 Einstein's Brownian theory

In this section we derive the main characteristics of bulk Brownian motion in the manner of Einstein in 1905 by summarizing the section 4 of [**einstein˙uber˙1905**]. We then examine the random motion of particles suspended in a liquid and its relation to diffusion, caused by thermal molecular motion. We assume that each particle motion is independent of other particles; also the motions of one particle at different times are assumed to be independent of one another provided that the time interval is not too small. Furthermore, we introduce a time interval τ which is small compared to the observation time but large enough so that the displacements in two consecutive time intervals τ may be taken as independent events.

For simplicity, we will here look only at the Brownian motion of n particles in 1D along the x axis. In a time interval τ the position of each particle will increase by a displacement Δ, positive or negative. The number of particles $\mathrm{d}n$ experiencing a displacement lying between Δ and $\Delta + \mathrm{d}\Delta$ in a time interval τ is written as:

$$\mathrm{d}n = n\varphi(\Delta)\mathrm{d}\Delta \ , \tag{3.2.1}$$

where:

$$\int_{-\infty}^{\infty} \varphi(\Delta) \mathrm{d}\Delta = 1 \ , \tag{3.2.2}$$

and φ is the probability distribution of displacement. We assume for now, that φ is a Gaussian distribution, with a variance scaling linearly with τ. Additionally, since such a distribution is even, it satisfies: $\varphi(\Delta) = \varphi(-\Delta)$.

Let $f(x,t)$ be the number of particles per unit volume. From the definition of the function $\varphi(\Delta)$ we can obtain the distribution of particles found at time $t+\tau$ from their distribution at a time t, through:

$$f(x, t+\tau) = \int_{-\infty}^{+\infty} f(x-\Delta, t)\varphi(\Delta)\mathrm{d}\Delta \ . \tag{3.2.3}$$

Since we suppose that τ is very small with respect to t, we have at first order in time:

$$f(x, t+\tau) \simeq f(x,t) + \tau \frac{\partial f}{\partial t} \ . \tag{3.2.4}$$

Besides, we can Taylor expand $f(x+\Delta, t)$ in powers of Δ since only small values of Δ contribute. We obtain:

$$f(x-\Delta, t) = f(x,t) - \Delta \frac{\partial f(x,t)}{\partial x} + \frac{\Delta^2}{2!} \frac{\partial^2 f(x,t)}{\partial x^2} \ . \tag{3.2.5}$$

Combining Eqs.3.2.4, 3.2.5 and 3.2.3 we obtain:

$$f + \frac{\partial f}{\partial t}\tau = f \int_{-\infty}^{+\infty} \varphi(\Delta)\mathrm{d}\Delta + \frac{\partial f}{\partial x} \int_{-\infty}^{+\infty} \Delta\varphi(\Delta)\mathrm{d}\Delta + \frac{\partial^2 f}{\partial x^2} \int_{-\infty}^{+\infty} \frac{\Delta^2}{2}\varphi(\Delta)\mathrm{d}\Delta \ . \tag{3.2.6}$$

On the right-hand side, since $\varphi(x)$ is an even function, the second term vanishes. Considering Eq. (3.2.2) and invoking the following definition:

$$\frac{1}{\tau} \int_{-\infty}^{+\infty} \frac{\Delta^2}{2} \varphi(\Delta)\mathrm{d}\Delta = D \ , \tag{3.2.7}$$

Eq. (3.2.6) finally becomes:

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2} \ . \tag{3.2.8}$$

We can here recognize a partial equation of diffusion with D the diffusion coefficient. We will now initiate the same position $x = 0$ for all the particles at $t = 0$ as in Fig.3.2.1. $f(x,t)\mathrm{d}x$ denotes the number of particles whose positions have increased between the times 0 and t by a quantity lying between x and $x + dx$ such that we must have:

$$f(x \neq 0, t = 0) = 0 \text{ and } \int_{-\infty}^{+\infty} f(x,t)dx = n \ . \tag{3.2.9}$$

The solution Eq. (3.2.8) is then the Green's function of the heat equation in the bulk:

$$f(x,t) = \frac{1}{\sqrt{4\pi D}} \frac{\exp\left(\frac{-x^2}{4Dt}\right)}{\sqrt{t}} \ . \tag{3.2.10}$$

From this solution we can see that the mean value of the displacement along the x axis is equal to 0 and the square root of the arithmetic mean of the squares of displacements (that we commonly call the Root Mean Square Displacement (RMSD)) is given by:

$$\lambda_x = \sqrt{\langle \Delta^2 \rangle} = \sqrt{2Dt}. \tag{3.2.11}$$

The mean displacement is thus proportional to the square root of time. This result is generally the first behavior that we check when we study Brownian motion. In 3D, the square root of the MSD is given by $\lambda_x\sqrt{3}$.

Previously in his article [**einstein˙uber˙1905**], Einstein had found by writing the thermodynamic equilibrium of a suspension of particles that the diffusion coefficient of a particle should read:

$$D = \frac{RT}{N_\mathrm{A}} \frac{1}{6\pi\eta a} = \frac{k_\mathrm{B}T}{6\pi\eta a} \ , \tag{3.2.12}$$

with R the gas constant, T the temperature, N_A the Avogadro number, η the fluid viscosity and k_B the Boltzmann constant. Thus, an experimental measurement of D lead

to a measurement of the Avogadro number since:

$$N_{\mathrm{A}} = \frac{t}{\lambda_x^2} \frac{RT}{3\pi\eta a} \ . \tag{3.2.13}$$

Furthermore, measuring N_{A} also gives us the mass of atoms and molecules since the mass of a mole is known; as an example the mass of an oxygen atom will be given by $\left(\frac{16}{N_{\mathrm{A}}}\right)$ and the mass of a water molecule by $\frac{18}{N_{\mathrm{A}}}$. Finally, Einstein ends up is article [**einstein˙uber˙1905**] by writing, *"Let us hope that a researcher will soon succeed in solving the problem posed here, which is of such importance in the theory of heat!"* I would like here to emphasize the importance of solving this problem at the very beginning of the 20th century. At this time two hypotheses about the fundamental matter components existed, one involving energy and a continuum description in terms of field, and the other one, discrete atoms, especially supported by Boltzmann and his kinetic theory of gases, used by Einstein. Due to a lot of conceptual misunderstandings and experimental error scientist such as Svedberg or Henri thought that Einstein's theory was false [**genthon˙concept˙2020**] by even suggesting that the statistical properties of Brownian motion were changing with the pH of the solution. It is finally in 1908 that Chaudesaigues and Perrin published all the evidence to prove Einstein's theory mainly by their ability to create particle emulsions of well-controlled radii.

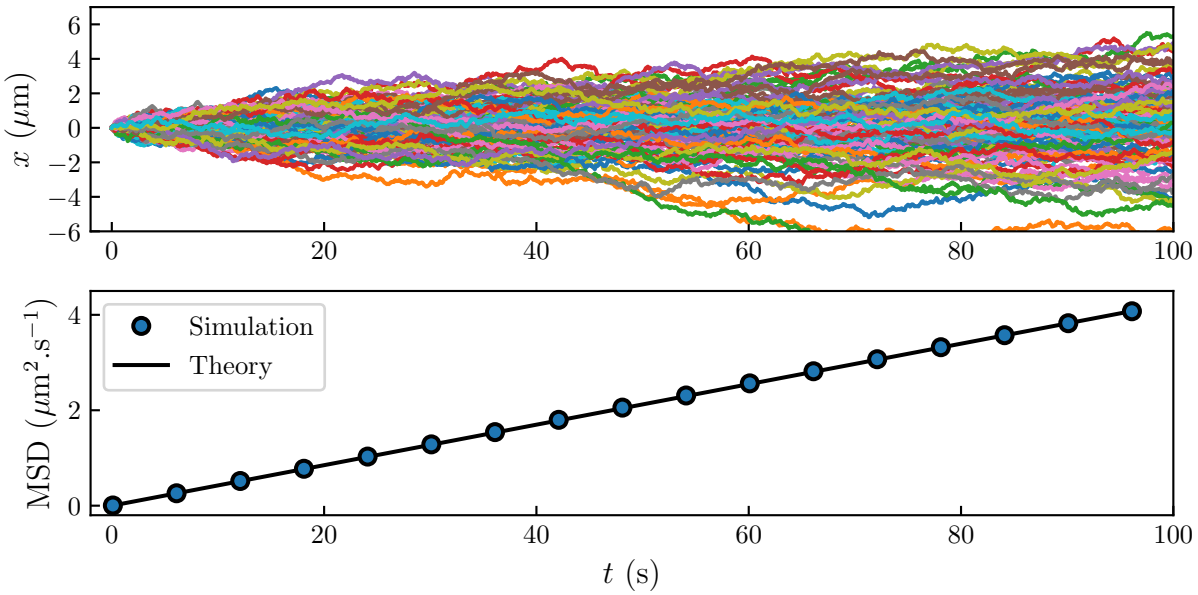


Figure 3.2.1: Simulation of over-damped Brownian motion in the bulk (see Eq. (3.4.9)) of 1 μm particles in water. On the top each line represents the trajectory of a Brownian particle over 100 seconds. A total of 100 trajectories are shown. On the bottom, bullets represent the Mean Square Displacement (MSD) computed from the simulated trajectories. The plain black line represents Einstein's theory, which is computed from the square of Eq. (3.2.11).

3.3 The Langevin Equation

in physics we generally describe Brownian motion through a particular Stochastic Differential Equations (SDE). This model was introduced in 1908 by Langevin [**langevin˙sur˙1908**], this model is now used by the major part of physicists working on random processes. The Langevin equation for a free colloid reads:

$$m\mathrm{d}V_t = -\gamma V_t \mathrm{d}t + g\mathrm{d}B_t \ , \tag{3.3.1}$$

with m the mass and V_t the velocity of the particle. This SDE is the Newton's second law, relating the particle momentum change on the left-hand side of the equation to forces on the right-hand side. We see that the total force applied on the particle is given by two terms: a friction term, with a Stokes-like fluid friction coefficient γ, a random force with g that we will detail for a spherical particle, dBt a random noise which has a Gaussian distribution of zero mean thus:

$$\langle \mathrm{d}B_t \rangle = 0 \ , \tag{3.3.2}$$

and variance equal to:

$$\langle \mathrm{d}B_t^2 \rangle = \mathrm{d}t \ . \tag{3.3.3}$$

For a spherical particle, the friction term is given by the Stoke's formula: $\gamma = 6\pi\eta a$ with η the fluid viscosity and g the particle radius. Thus, we can derive the mean value of the particle velocity as:

$$\left\langle \frac{\mathrm{d}V_t}{\mathrm{d}t} \right\rangle = -\frac{\gamma}{m}\langle V_t \rangle \mathrm{d}t + \frac{g}{m}\langle \mathrm{d}B_t \rangle \ , \tag{3.3.4}$$

with the properties of dB_t given by Eq. (3.3.2), it becomes:

$$\langle \mathrm{d}V_t \rangle = -\frac{\gamma}{m}\langle V_t \rangle \mathrm{d}t \ . \tag{3.3.5}$$

Moreover, without a loss of generality, the average of a variable x, $\langle x \rangle$, is done over a set

of N observations $\{x_i\}$ such as:

$$\langle x \rangle = \frac{1}{N} \sum_{i=1}^{N} x_i \ , \tag{3.3.6}$$

one can then show that:

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle x \rangle = \frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{1}{N}\sum_{i=1}^{N} x_i\right] = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathrm{d}}{\mathrm{d}t}x_i = \langle \frac{\mathrm{d}}{\mathrm{d}t}x \rangle \ . \tag{3.3.7}$$

The latter thus shows that it is possible to invert average value $\langle \cdot \rangle$ and a derivative. Therefore, Eq. (3.3.5) becomes:

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle V_t \rangle = -\frac{\gamma}{m}\langle V_t \rangle \ , \tag{3.3.8}$$

which has a familiar solution:

$$\langle V_t(t) \rangle = V_0 \mathrm{e}^{-\frac{\gamma}{m}t} \ , \tag{3.3.9}$$

with V_0 an initial velocity. This result shows that the average of the velocity should decay to zero with a characteristic time $\tau_{\mathrm{B}} = \frac{m}{\gamma}$. For instance, the polystyrene particles used during my experiments which are micrometric we have $\tau_{\mathrm{B}} \approx 10^{-7}$ s. This signifies that if we measure the displacements of a particle with a time interval $\tau >> \tau_{\mathrm{B}}$ the displacement can be taken as independent events as it was stated by Einstein. In physical terms, this means that we are in the over-damped regime, in this case the Langevin equation reads:

$$-\gamma V_t \mathrm{d}t + g \mathrm{d}B_t = 0 \ . \tag{3.3.10}$$

The experiments done during my thesis used a video camera that can reach a maximum of hundreds frames per second (fps) reaching time steps of $\approx 10^{-2}$ s. Therefore, all my work falls into the over-damped regime. Before focusing definitely on Eq. (3.3.10), we can use Eq. (3.3.4) to characterize further the unknown coefficient g. To do so we compute the mean square value of Eq. (3.3.4), starting by taking the second order Taylor expansion:

$$\begin{aligned} \mathrm{d}\left(V_t^2\right) &\simeq \frac{\partial V_t^2}{\partial V_t}\mathrm{d}V_t + \frac{1}{2}\frac{\partial^2 V_t^2}{\partial V_t^2}\left(\mathrm{d}V_t\right)^2 \\ &= 2V_t\mathrm{d}V_t + \left(\mathrm{d}V_t\right)^2 \end{aligned} \tag{3.3.11}$$

combining Eqs. (3.3.1) and (3.3.11), we obtain by only keeping the terms of order dt:

$$\mathrm{d}\left(V_t^2\right) = 2V_t\left(-\frac{\gamma}{m}V_t\mathrm{d}t + \frac{g}{m}\mathrm{d}B_t\right) + \frac{g^2}{m^2}\mathrm{d}B_t^2 \ . \tag{3.3.12}$$

Thus, the average value of $\mathrm{d}(V_t^2)$ reads:

$$\langle \mathrm{d}(V_t^2)\rangle = -2\frac{\gamma}{m}\langle V_t^2\rangle\mathrm{d}t + 2\frac{g}{m}\langle V_t\mathrm{d}B_t\rangle + \frac{g^2}{m^2}\langle \mathrm{d}B_t^2\rangle \ . \tag{3.3.13}$$

Moreover, since $\mathrm{d}B_t$ is chosen independently of the velocity V_t, one can write $\langle V_t\mathrm{d}B_t\rangle = \langle V_t\rangle\langle \mathrm{d}B_t\rangle = 0$. Taking the latter remark into account and the fact that $\langle \mathrm{d}B_t^2\rangle = \mathrm{d}t$, Eq. (3.3.13) becomes:

$$\langle d(V_t^2)\rangle = \left[-2\frac{\gamma}{m}\langle V_t^2\rangle + \frac{g^2}{m^2}\right]\mathrm{d}t \ . \tag{3.3.14}$$

Since equilibrium averages in thermodynamics must become time independent, we have $\langle d(V_t^2)\rangle = 0$, thus:

$$\langle V_t^2\rangle = \frac{g^2}{2\gamma m} \ . \tag{3.3.15}$$

Besides, from the equipartition of energy we also know that:

$$\langle \frac{1}{2}mV_t^2\rangle = \frac{1}{2}k_{\mathrm{B}}T \ . \tag{3.3.16}$$

The latter equation permits a direct determination of the amplitude of the noise g:

$$g = \sqrt{2k_{\mathrm{B}}T\gamma} \ . \tag{3.3.17}$$

The latter result permits computing the amplitude of the random force in the Langevin equation. Taking the over-damped Langevin equation, it reads:

$$V_t \mathrm{d}t = \sqrt{2\frac{k_\mathrm{B}T}{\gamma}}\mathrm{d}B_t \tag{3.3.18}$$

Furthermore, one can write the position of the particle X_t at a time t, such as:

$$X_t = \int_0^t V_{t'} dt' \ , \tag{3.3.19}$$

where we can suppose at the initial time $t = 0$ that $X_0 = 0$. Computing $\langle X_t^2 \rangle$ using Eqs. (3.3.18), (3.3.19) and (3.3.3) thus gives:

$$\langle X_t^2 \rangle = 2\frac{k_\mathrm{B}T}{\gamma}t = 2Dt \tag{3.3.20}$$

By relating $\langle X_t^2 \rangle$ to the Mean Square Displacement (MSD) to the initial position such as:

$$\mathrm{MSD} = \langle (X_0 - X_t)^2 \rangle = \langle X_t^2 \rangle \ , \tag{3.3.21}$$

we obtain that the MSD should be linear with the time. This result confirms that using the over-damped Langevin equation, leads to the Einstein's result Eq. (3.2.11). Where one can identify the diffusion coefficient of the particle to be $D = k_\mathrm{B}T/\gamma$. Additionally, the latter identity is called the Stokes-Einstein relation.

Additionally, the Langevin equation can be used to compute correlator such as the velocity correlator $\langle V_{t'} V_{t''} \rangle$ that we will detail below. Indeed, if we use the full-Langevin equation, $\langle X_t^2 \rangle$ cannot be easily computed since $m\mathrm{d}V_t$ does not vanish. We would thus need to rewrite Eq. (3.3.20) using the velocity correletor such as:

$$\langle X_t^2 \rangle = \int_0^t \int_0^t \langle V_{t'} V_{t''} \rangle dt' dt'' \ . \tag{3.3.22}$$

Let us now study how the two-point correlator function $\langle V_{t'} V_{t''} \rangle$, using the full-Langevin equation multiplied by V_0 and following the same steps as for Eq. (3.3.9), one has:

$$\langle V_t V_0 \rangle = \langle V_0^2 \rangle \mathrm{e}^{-t/\tau_\mathrm{B}} \ . \tag{3.3.23}$$

As the equilibrium state is invariant under temporal translation and assuming that V_0 has an equilibrium steady-state distribution with $\langle V_0^2 \rangle = k_\mathrm{B}T/m$ we have:

$$\langle V_t V_t' \rangle = \frac{k_\mathrm{B}T}{m} \mathrm{e}^{-|t-t'|/\tau_\mathrm{B}} \ . \tag{3.3.24}$$

One can solve Eq. (3.3.22) by splitting the integral in two parts, where $t' > t''$ and $t' < t''$:

$$\begin{aligned} \langle X_t^2 \rangle &= \frac{k_\mathrm{B}T}{m} \int_0^t \mathrm{d}t' \int_0^{t'} \mathrm{d}t'' \mathrm{e}^{-|t'-t''|/\tau_\mathrm{B}} = 2\frac{k_\mathrm{B}T}{\gamma} \left(\int_0^t \mathrm{d}t' \left[1 - \mathrm{e}^{-t'/\tau_\mathrm{B}} \right] \right) \\ &= 2\frac{k_\mathrm{B}T}{\gamma} \left(t - \tau_\mathrm{B} \left[1 - \mathrm{e}^{-t/\tau_\mathrm{B}} \right] \right) \ . \end{aligned} \tag{3.3.25}$$

We can extract two results from that equation. At a short time $t << \tau_\mathrm{B}$, one has:

$$\begin{aligned} \langle X_t^2 \rangle &\simeq 2\frac{k_\mathrm{B}T}{\gamma} \left(t - \tau_\mathrm{B} \left[1 - 1 + \frac{t}{\tau_\mathrm{B}} - \frac{t^2}{2\tau_\mathrm{B}^2} \right] \right) \\ &= \frac{k_\mathrm{B}T}{m} t^2 \ . \end{aligned} \tag{3.3.26}$$

This is the ballistic regime. If one can experimentally explore times shorter than τ_B one will then measure the instantaneous velocity of the particle. At longer times, $t >> t_\mathrm{B}$, the MSD is given by:

$$\langle X_t^2 \rangle \simeq 2\frac{k_\mathrm{B}T}{\gamma} t = 2Dt \ . \tag{3.3.27}$$

This is the diffusive regime where the MSD, as found earlier, Eq. (3.3.20) with the over-damped Langevin equation. To study this different result, it can be interesting to simulate the Brownian motion.

3.4 Numerical simulation of bulk Brownian motion

3.4.1 The numerical Langevin Equation

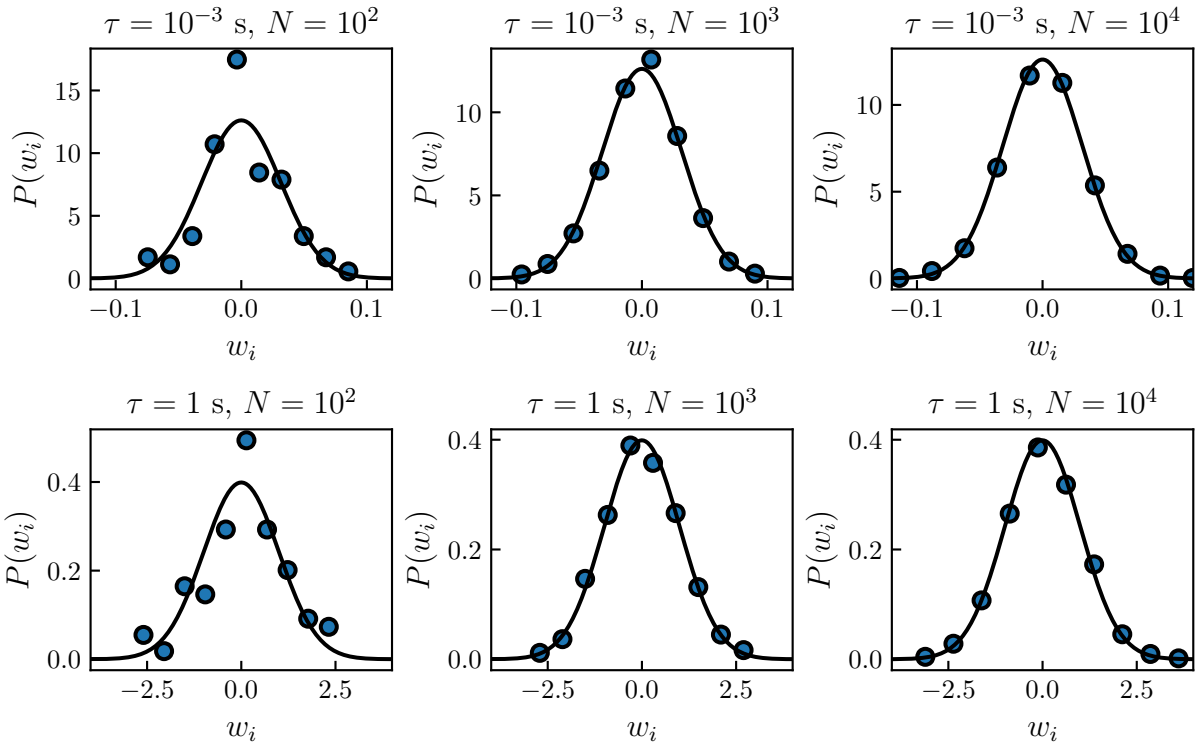


Figure 3.4.1: Bullets represents the probability density function of w_i, a Gaussian-distributed number with a mean value $\langle w_i \rangle$ and a variance $\langle w_i^2 \rangle = \tau$. The plain black line is a Gaussian distribution of zero mean and a variance τ (see Eq. (3.4.3)). In the first row, the simulation is done with $\tau = 10^{-3}$ s and $\tau = 1$ s on the second one. Each column corresponds to a number N of draws. From the left to the right: $N = 10^2,\ 10^3$ and 10^4.

The Langevin equation is an ordinary differential equation that can easily be numerically simulated in the bulk case. We approximate the continuous position X_t of a particle at a time t by a discrete-time sequence x_i which is the solution of the equation at a time $t_i = i\tau$, τ being the time step of the simulation. One can then use the Euler method to numerically write V_t as:

$$V_t \simeq \frac{x_i - x_{i-1}}{\tau} \ , \tag{3.4.1}$$

and $\mathrm{d}V_t/\mathrm{d}t$ as

$$\begin{aligned}\frac{\mathrm{d}V_t}{\mathrm{d}t} &\simeq \frac{\frac{x_i - x_{i-1}}{\tau} - \frac{x_{i-1} - x_{i-2}}{\tau}}{\tau} \\ &= \frac{x_i - 2x_{i-1} + x_{i-2}}{\tau^2} \ .\end{aligned} \tag{3.4.2}$$

The only term remaining to be computed numerically is the random term $\mathrm{d}B_t$. One can thus replace $\mathrm{d}B_t/\mathrm{d}t$ by w_i/τ [1] a Gaussian distributed random number generated with a mean $\langle w_i \rangle = 0$ and a variance $\langle w_i^2 \rangle = \tau$. The Probability Density function (PDF) of the Gaussian distribution is thus given by:

$$P(w_i) = \frac{1}{\sqrt{2\pi\tau}} \mathrm{e}^{-\frac{w_i^2}{2\tau}}. \tag{3.4.3}$$

The random number w_i can be generated with the following Python snippet.

```
import numpy as np

tau = 0.5  # time step in seconds
wi = np.random.normal(0, np.sqrt(tau))
```

In the latter, `random.normal()` is a built-in Numpy module that permits the generation of Gaussian-distributed random numbers. Finally, by combining Eqs. (3.4.1) and (3.4.2), the full-Langevin equation becomes:

$$m\frac{x_i - 2x_{i-1} + x_{i-2}}{\tau^2} = -\gamma\frac{x_i - x_{i-1}}{\tau} + \sqrt{2k_\mathrm{B}T\gamma}\frac{w_i}{\tau} \ . \tag{3.4.4}$$

From the latter, one can write x_i as:

$$x_i = \frac{2 + \tau/\tau_\mathrm{B}}{1 + \tau/\tau_\mathrm{B}} x_{i-1} - \frac{1}{1 + \tau/\tau_\mathrm{B}} x_{i-2} + \frac{\sqrt{2k_\mathrm{B}T\gamma}}{m(1 + \tau/\tau_\mathrm{B})}\tau w_i \ , \tag{3.4.5}$$

where we can observe that two initial conditions are needed, *i.e* the first two positions of the particle. Numerically, these positions could be randomly generated or set to 0. If enough statics are generated, it will not affect the results.

[1] The notation w was choosen since in mathematical terms, a real-valued continuous-time stochastic process such as $\mathrm{d}B_t$ is called a Wiener process in honor of Norbert Wiener [**durrett˙probability˙2019**].

3.4.2 Simulating Brownian Motion Using Python

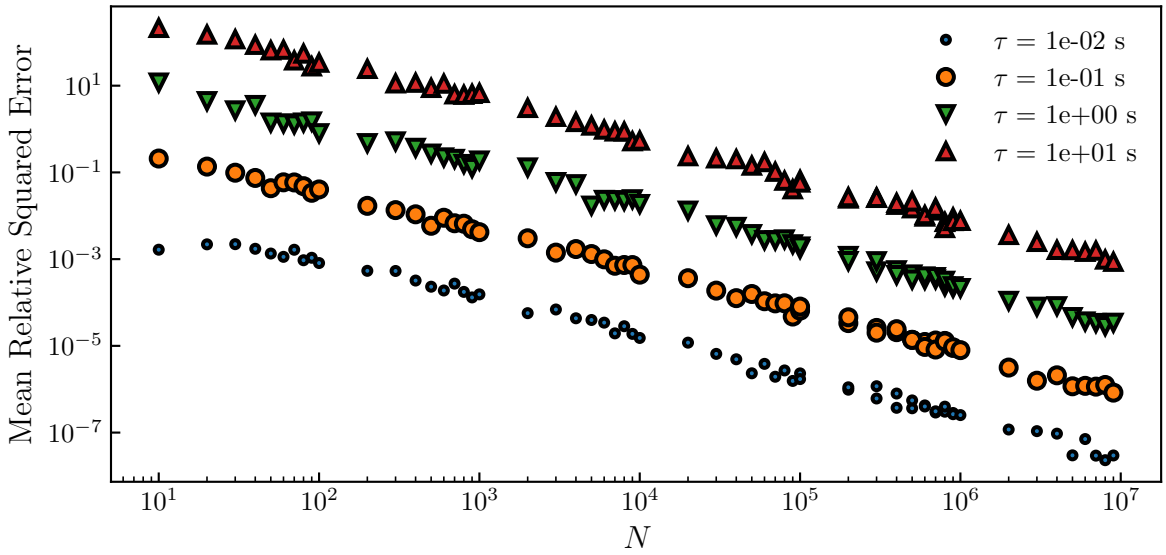


Figure 3.4.2: Mean Relative Squared Error (MRSE) of the Probability Density Function PDF measured from a generation of N Gaussian random numbers w_i and the actual Gaussian (see Eq. (3.4.3)) from which the generation is done. The generation is done over a Gaussian which has a mean value $\langle w_i \rangle = 0$ and variance $\langle w_i^2 \rangle = \tau$. We explore parameter ranges from $N = 10$ to 10^7 and $\tau = 10^{-2}$ to 10 s.

Before, diving into the simulation, it could be interesting to wonder how long the simulation should be. Indeed, at equilibrium, for the different observables' mean values to remain constant, we should wait a sufficient amount of time. It is possible to follow a qualitative approach by generating N numbers w_i, measuring the resulting PDF $P_{\mathrm{c}}(w_i)$ and looking for how much we need to increase N to have $P_{\mathrm{c}}(w_i) \approx P(w_i)$, under some given small-error criterion. As we can see in Fig.3.4.1, for simulations made with $\tau = 10^{-3}$ s and $\tau = 1$ s, we observe that as we increase N, the measured PDF, gets closer to the real one given by Eq. (3.4.3).

To have a more quantitative approach, one can compute the Mean Relative Squared Error (MRSE)) between the measured PDF $P_{\mathrm{c}}(w_i)$ and the nominal function $P(w_i)$ as a function of the number N of generated numbers, as:

$$\mathrm{MRSE} = \left\langle \frac{\left(P_{\mathrm{c}}(w_i) - P(w_i)\right)^2}{P(w_i)^2} \right\rangle \Bigg|_N \tag{3.4.6}$$

where the notation $|_N$ denotes the average over N realizations. Additionally, since we measure $P_c(w_i)$ by doing a histogram, the question of how many bins are used should be an-

swered. It is possible to use the optimal Freedan-Diaconis rule [**freedman˙histogram˙1981**] to compute the width of the bins to be used in a histogram. This rule reads:

$$\text{Bin width} = 2\frac{\text{IQR}(\{w_i\})}{\sqrt[3]{N}}, \tag{3.4.7}$$

where IQR is the interquartile range, and $\{w_i\}$ a sample of N random numbers w_i. Moreover, one should at least take 2 bins as a minimum. The optimal number of bins can be computed using the following Python snippet.

```
1
2  import numpy as np
3
4  def _iqr(wi):
5    """Function to compute interquartile range."""
6      return np.subtract(*np.percentile(x, [75, 25]))
7
8    def optimal_bins(wi):
9      """
10     Function to compute the optinal number of bins using Freedan-Diaconis rule.
11     Input: list of random numbers | Output: optimal bins number
12     """
13
14     n = int(diff(wi) / (2 * _iqr(wi) * np.power(len(wi), -1 / 3)))
15
16     if n <= 2:
17       return 2
18     else:
19       return n
```

As we can see in Fig.3.4.2, for τ varying between 10^{-2} and 10 seconds, and N between 10 and 10^6, the MRSE decreases as N increases. Moreover, it is interesting to observe that the MRSE is greater as τ increases for a fixed N value. As an example, we would need to only generate $N = 10^{-3}$ numbers to obtain an MRSE of 10^{-4} for $\tau = 0.1$ s, while we would need $N = 10^6$ for $\tau = 1$ s.

Now that the Langevin equation has been numerically implemented, one could use it to simulate Brownian trajectories. A simple way to do the simulation using Python is provided in appendix.A.1. A set of trajectories simulated for a fictive particle of radius $a = 1$ μm and mass $m = 10$ μg in water is shown in Fig.3.4.3-a). For such a particle, the diffusive characteristic time is $\tau_B = 0.53$ s. Moreover, as one can see in Fig.3.4.3-b), the

MSD is correctly modeled by Eq. (3.3.26) for $\tau << \tau_{\mathrm{B}}$, and by Eq. (3.3.27) for $\tau >> \tau_{\mathrm{B}}$. Note that for non-continuous data such as the simulation data presented here or sampled experimental trajectories, and for a given time increment Δt, the MSD is generally defined by:

$$\left\langle \Delta x^2 \right\rangle_t = \left\langle (x_i(t+\Delta t) - x_i(t))^2 \right\rangle_t \ , \tag{3.4.8}$$

where the average $\langle\rangle|_t$ is performed over time t.The following Python function can be used to numerically compute the MSD Eq. (3.4.8).

```
def msd(x, Dt):
  """Function that returns the MSD for a list of time indices Dt for a trajectory x"""
  _msd = lambda x, Dt: np.mean((x[:-Dt] - x[Dt:]) ** 2)
  return [_msd(x, i) for i in t]
```

Additionally, as we have seen earlier, the Langevin Equation can be simplified to it is over-damped version of Eq. (3.3.10). In this case, the time step τ of the simulation should be greater than the characteristic time τ_{B}. Thus, if one is interested in the long-time statistical properties of Brownian motion one can use the over-damped Langevin equation. In this case, by putting $m = 0$ into Eq. (3.4.5), one can write x_i as:

$$x_i = x_{i-1} + \sqrt{2D} w_i \ . \tag{3.4.9}$$

The statistical properties at a long time could be retrieved by simulating Brownian motion using the full-Langevin equation. But, since the integration scheme used for Eq. (3.4.5) requires $\tau << \tau_{\mathrm{B}}$, long simulation runs are necessary to retrieve the over-damped properties. A simulation of the over-damped Brownian motion trajectories using Eq. (3.4.9) is shown in Fig.3.2.1 and is realized using the following Python Snippet.

```
import numpy as np

N = 1000  # trajectory length
D = 1  # diffusion coefficient
tau = 0.5  # time step
trajectory = np.cumsum(np.sqrt(2 * D) * np.random.normal(0, np.sqrt(tau), N))
```

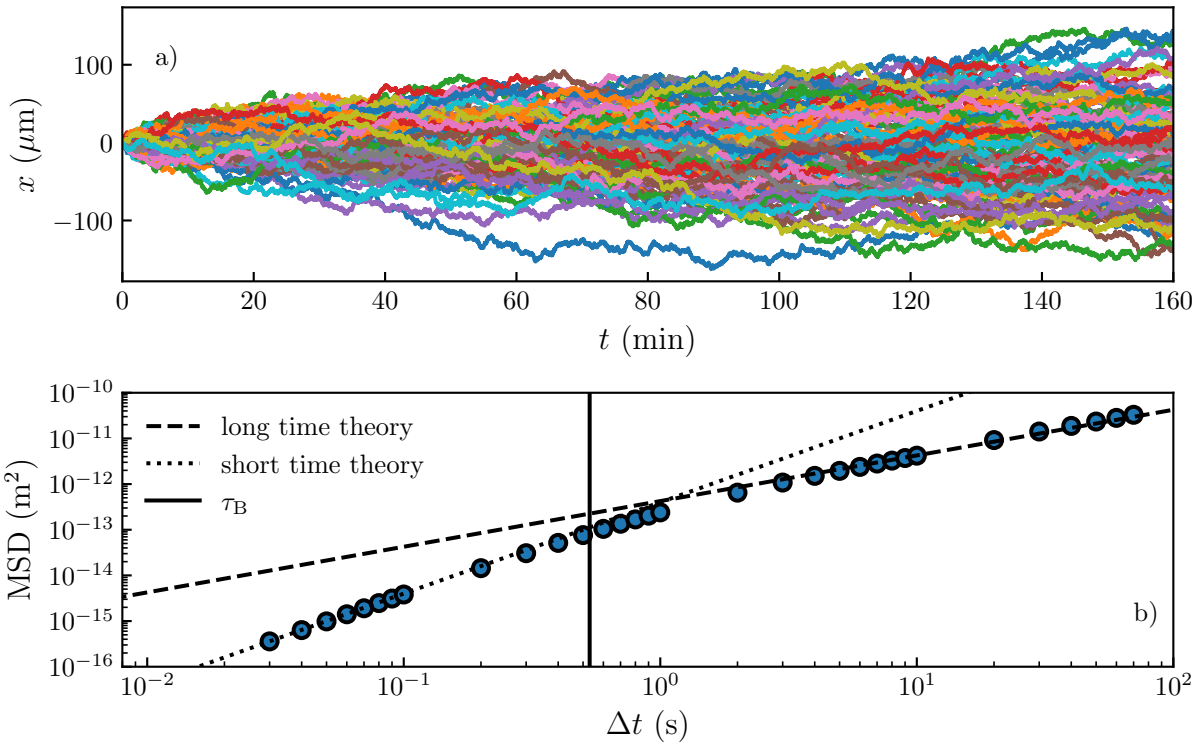


Figure 3.4.3: a) Set of 100 trajectories simulated using the full-Langevin equation (see Eq. (3.4.5)) for particles of a radius $a = 1$ μm and mass $m = 10$ μg in water, with viscosity $\eta = 0.001$ Pa.s. The simulations are done with a time step $\tau = 0.01$ s. b) Bullets represent the measured Mean Squared Displacements (MSD) of the simulated trajectories. The plain black line represents the characteristic inertial timescale, $\tau_B = m/\gamma = 0.53$ s. The dotted line represents the MSD in the ballistic regime (see Eq. (3.3.26)), when $\Delta \ll \tau_B$. The dashed line represents the MSD in the diffusive regime (see Eq. (3.3.27)), when $\Delta t \gg \tau_B$, MSD $= 2D\Delta t$. A detailed explanation of the simulation process can be found in appendix A.1.

3.4.3 Speedup using Cython

I would like to point out that the optimization of a simple simulation of a Brownian trajectory can be interesting. Indeed, using a pure Python code as presented in the first part of appendix.A.1, the simulation of one trajectory of 10^6 steps, needs 6 s to be computed. Thus, more than 10 minutes are required to compute the 100 trajectories shown in Fig.3.4.3. This is long and due to how Python systematically verifies what is allowed. Indeed, it verifies at each step of the `for` loop the object type of each variable and if the mathematical operation are possible. This is generally the main drawback [**ampomah˙qualitative˙2017**] of interpreted language, and the only solution is to use a compiled language (*e.g* C or Fortran).

The difference between an interpreted (*e.g.* Python) and compiled language (*e.g.* C) lies in the result of the process of interpreting or compiling. A compiler (*e.g.* gcc for

the C language) translate the source code into the computer native language, and create an executable file. The execution of compiled language does not require any more translations, and hence run significantly faster. Contrariwise, an interpreted language is not translated in advance, but is done at the execution, line by line, and each time the program is executed. This process is done by the interpreter, such as Python, Matlab or Perl for their eponymous language. At each execution, the time taken by the interpreter to read and execute each line slows the process, causing execution to take more time.

To overcome this problem with Python, the `cython` package has been developed to translate in C and compile the part of the code that is long to execute, especially the `for` loops. Thus, one can transform Python source code into a hybrid Python-C code. As presented in appendix.A.1, compiling the `for` loop using `cython` in the full-Langevin simulations reduces the time to generate a 10^6-step trajectory from 6 s to 30 ms thus achieving a speedup factor of 200. Moreover, in the hybrid version, the execution time is limited by the random number generation. Indeed, it takes $\approx$ 24.0ms using `numpy` to generate 10^6 w_i numbers and $\approx$ 6 ms for the trajectory computation. Additionally, as shown at the end of appendix.A.1, even a pure C implementation of the random generation can be slower than the `numpy` one, thanks to `numpy`'s memory optimization. Thus, by using the above mixed-language strategy, the simulation is optimal with the current tools and language at hand.

3.5 Conclusion

In this chapter, we have covered the history of Brownian motion, from the first observation by Robert Brown in the middle of the 19th century to its mathematical and experimental proofs in the early 20th century. We have then described mathematically the bulk Brownian motion and its important statistical properties. Finally, we have used the latter description to simulate Brownian motion using both the full-Langevin equation, and its over-damped version.

4 Particle Characterization and Tracking Using Optical Interferences

4.1 Introduction

Properties of coherent light to produce interference has been widely deployed in metrology for a long time as illustrated by, for example, the famous Fabry-Pérot [**fabry˙theorie˙1899**, **perot˙application˙1899**] and Michelson interferometers [**michelson˙relative˙1887**]. The latter was initially employed to evaluate the rotation of the Earth and is still deployed today for the recent measurement of gravitational waves [**ligo˙scientific˙collaboration˙and˙virgo˙collab**
Since the beginning of the century, interest on tracking and characterizing colloidal particles risen thanks to the democratization of micro fluidics and lab-on-a-chip technologies. In the following, I will provide some insights on the three most used tracking methods:

- Reflection Interference Contrast Microscopy (RICM)
- Lorenz-Mie theory
- Rayleigh-Sommerfeld back propagation

The first one, RICM, uses the principle of optical-path difference in a Michelson interferometer. The other two, use the interferences between the light scattered by the colloid and the incident light. Generally, both sources are collinear, so that we speak of in-line holography.

4.2 Reflection Interference Contrast Microscopy

RICM was first introduced in cell biology by Curtis to study embryonic chick heart fibroblasts [**curtis˙mechanism˙1964**] in 1964. RICM gained in popularity 40 years after both in biology and physics [**filler˙reflection˙2000**, **siver˙use˙2000**, **weber˙2˙2003**, **limozin˙quantitative˙2009**, **nadal˙probing˙2002**, **raedler˙measurement˙1992**]. It was also used recently in soft matter physics to study the elastohydrodynamic lift at a soft wall [**davies˙elastohydrodynamic˙2018**].

When we illuminate a colloid with a plane wave from the bottom, a part of the light is reflected at the surface of the glass substrate and another part, at the colloid surface.

The difference of optical paths between two reflections creates an interference pattern. Let us focus on the mathematical description of this phenomenon. In the far field, we can describe two one-dimensional electric field vectors with same angular frequency ω [**f˙bohren˙absorption˙1998**] as propagative waves, through:

$$\vec{E}_1(\vec{r},t) = \vec{E}_{0_1}\cos(\vec{k}_1\cdot\vec{r}-\omega t+\epsilon_1)\ , \tag{4.2.1}$$

and:

$$\vec{E}_2(\vec{r},t) = \vec{E}_{0_2}\cos(\vec{k}_2\cdot\vec{r}-\omega t+\epsilon_2)\ , \tag{4.2.2}$$

where the k_i are the wave vector satisfying $|\vec{k_i}| = k = 2\pi n_\mathrm{m}/\lambda$, the angular wavenumber, λ being the illumination wavelength, n_m the optical index of the medium, $\epsilon_{1,2}$ the initial phases of each wave and $\vec{r}$ the position. Here, the origin ($\vec{r} = \vec{0}$) is taken at the position of the first reflection (*i.e.* the glass slide). Thus, on the particle, $\vec{r}$ is given by the particle's height such that $|r| = z$.

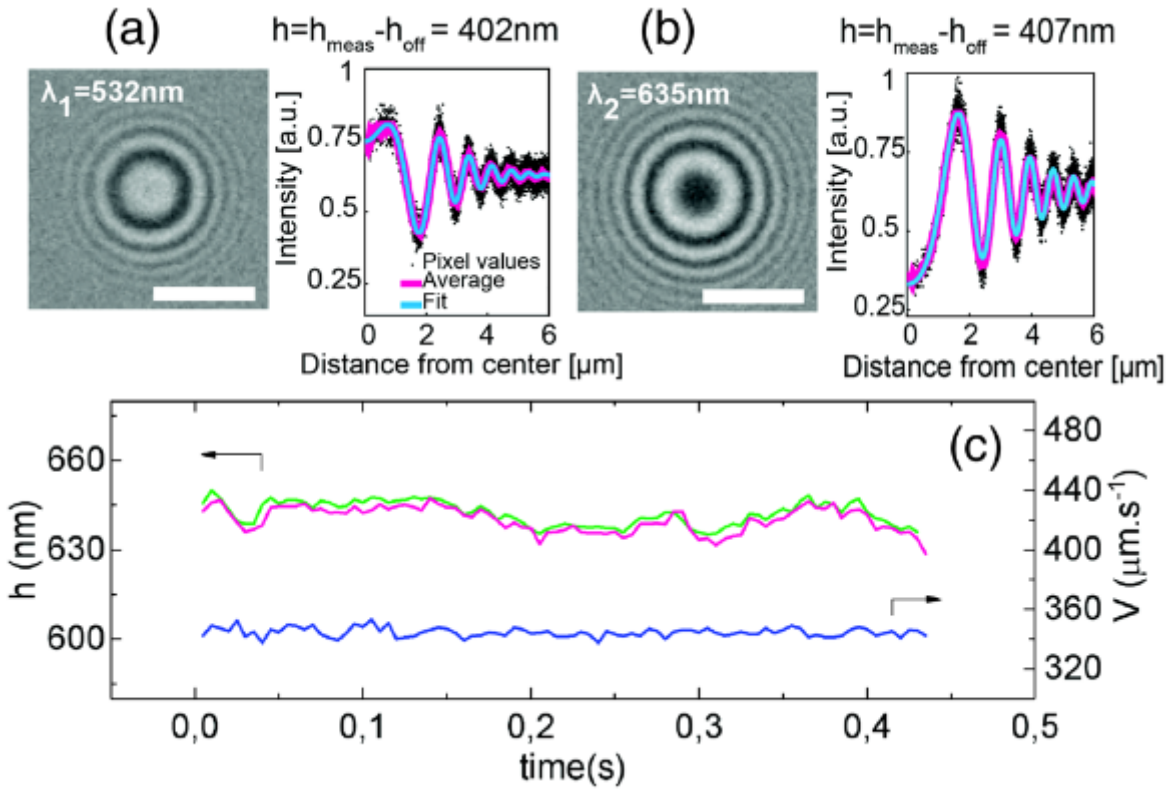


Figure 4.2.1: Figure from [**davies˙elastohydrodynamic˙2018**] representing RICM with two wavelengths. (a) Left: interference patterns created with a wavelength $\lambda_1 = 532$ nm (scale bar 5 μm). Right: radial intensity profile (black dots) extracted from the image, azimuthally averaged (magenta line) and fitted with Eq. (4.2.8) to measure the height of the particle (here noted h). (b) Same as (a) with a wavelength $\lambda_2 = 635$ nm. (c) Time series of the height h of a particle (green: λ_1, purple: λ_2) and the particle velocity measured along the flow (in blue).

Experimentally, one measures the intensity of the interference patterns. They can be computed from the time-averaged squared total electric field $\vec{E} = \vec{E}_1 + \vec{E}_2$. The measured intensity is thus given by:

$$I = \left\langle \vec{E}^2 \right\rangle = \left\langle \vec{E}_1^2 + \vec{E}_2^2 + 2\vec{E}_1 \cdot \vec{E}_2 \right\rangle = \left\langle \vec{E}_1^2 \right\rangle + \left\langle \vec{E}_2^2 \right\rangle + 2\left\langle \vec{E}_1 \cdot \vec{E}_2 \right\rangle \ , \tag{4.2.3}$$

where $\langle \vec{E}_1^2 \rangle$ and $\langle \vec{E}_2^2 \rangle$ are respectively given by I_1 and I_2, the incident intensities. Using trigonometry, we have:

$$\left\langle \vec{E}_1 \cdot \vec{E}_2 \right\rangle = \left\langle \frac{1}{2} \vec{E}_{0_1} \vec{E}_{0_2} \left[\cos\left(\vec{k}_1 \cdot \vec{r} - \vec{k}_1 \cdot \vec{r} + \phi \right) + \cos\left(2\omega t + \phi'\right) \right] \right\rangle_t \ . \tag{4.2.4}$$

As we average over time, the second cosine vanishes. Thus one has:

$$\left\langle \vec{E}_1 \cdot \vec{E}_2 \right\rangle = \frac{1}{2} \left\langle \vec{E}_{0_1} \vec{E}_{0_2} \right\rangle \cos\left(\vec{k}_1 \cdot \vec{r} - \vec{k}_2 \cdot \vec{r} + \phi \right) \ , \tag{4.2.5}$$

with ϕ the phase difference between the two fields, which is usually equal to π due to the reflection properties on a higher optical index. Indeed, a colloid has generally a greater optical index than the dilution medium. Finally, the total intensity can be read as:

$$I = I_1 + I_2 + 2\sqrt{I_1 I_2} \cos\left(\vec{k}_1 \cdot \vec{r} - \vec{k}_2 \cdot \vec{r} + \phi \right) \ . \tag{4.2.6}$$

By taking $k_1 = -k_2$ due to the reflection properties, we have:

$$I = I_1 + I_2 + 2\sqrt{I_1 I_2} \cos\left(\frac{4\pi n_{\mathrm{m}}}{\lambda} z + \phi \right) \ . \tag{4.2.7}$$

So far we supposed that the reflection occurs at a unique point; however, we would likely be using spherical colloids. Therefore, illuminating from the bottom, the reflection happens on half of the sphere surface. Moreover, thanks to the spherical geometry the holograms exhibit a radial symmetry, one can write the radial interference intensity $I(x)$, with x the distance from the pattern center, through [**raedler˙measurement˙1992**]:

$$I(x) = A_0 + A_1 \mathrm{e}^{-b_1 x^2} + A_2 \mathrm{e}^{-b_2 x^2} \cos\left[\frac{4\pi n_m}{\lambda} \left(g(x) + z \right) + \phi \right] \ , \tag{4.2.8}$$

Where A_1 and b_1 are parameters [**raedler˙measurement˙1992**] that fit the slightly bent background that arises from diffuse reflection on the upper part of the sphere, A_2 and b_2 the decaying contrast of the higher order maxima, A_0 background intensity, and

$$g(x) = a - \sqrt{a^2 - x^2} \ , \tag{4.2.9}$$

is the sphere profile, to consider the increase sphere-wall as x increases. Finally, this method benefits from equations that are computationally light and enable a quick tracking of particles. However, as we can see in Eq. (4.2.8), because of the periodicity of the cosine, the interference pattern is the same for all heights z separated by a distance $\lambda/(2n_{\mathrm{m}}) \approx 200$ nm for $\lambda = 532$ nm and $n_{\mathrm{m}} = 1.33$.

It is possible to extend this separation to $\simeq 1.2\ \mu$m as used in [**davies˙elastohydrodynamic˙2018**] length by using two different wavelengths. Despite the spatial resolution of this method which can attain 10 nm, the measurement ambiguity is not compatible with the study of Brownian motion due to the periodicity above. Hence RICM it is not usable in our context. As a matter of fact, we experimentally reach height spans of a few microns.

4.3 Lorenz-Mie theory

When a colloid is illuminated with a plane wave, a part of the light is scattered. In consequence, the incident field $\vec{E}_0$ and scattered field $\vec{E}_\mathrm{s}$ interferes. The interference patterns thus obtained are called holograms. If the particle is not smaller than the illumination wavelength, it is not possible to use Rayleigh's approximations [**strutt˙lviii˙1871**] to describe the scattering. Instead, one needs to use the Lorenz-Mie theory for dielectric spheres. This theory was developed by Lorenz and independently by Mie in 1880 and 1908, respectively [**lorenz˙lysbevaegelsen˙1890**, **mie˙beitrage˙1908**].

It is in the early 2000s that the Lorenz-Mie theory was first used in order to track and characterize particles [**ovryn˙imaging˙2000**, **lee˙characterizing˙2007**]. Since then, a lot of studies have been realized with this technique [**katz˙applications˙2010**]. In the following, I will describe the Lorenz-Mie method. In this part, the height z of the particle is the distance between the particle's center and the focal plane of the objective lens.

Let the incident field be a plane wave uniformly polarized along an axis $\hat{e}$, with an amplitude E_0 and propagating along the $\hat{z}$ direction :

$$\vec{E}_0(\vec{r}, z) = E_0(\vec{r})\mathrm{e}^{ikz}\hat{e} \tag{4.3.1}$$

Let us consider a particle of radius a at a position $\vec{r}_\mathrm{p}$. In such case, the scattered field can be written using the Lorenz-Mie theory [**f˙bohren˙absorption˙1998**], as:

$$\vec{E}_\mathrm{s}(\vec{r}, z) = \vec{f}_\mathrm{s}(k(\vec{r} - \vec{r}_p))E_0(\vec{r}) \exp(-ikz)\ , \tag{4.3.2}$$

with $\vec{f}_\mathrm{s}$, the Lorenz-Mie scattering function. The intensity I that we measure at $\vec{r}$ is given by the intensity of superimposition of the incident and scattered amplitudes. Since the measurements are done at the focal plane, i.e. $z = 0$, I is given by:

$$\begin{aligned} I(\vec{r}) &= |\vec{E}_{\mathrm{s}}(\vec{r},0) + \vec{E}_0(\vec{r},0)|^2 \\ &= E_0^2(\vec{r}) + 2E_0^2\Re\left(\vec{f}_{\mathrm{s}}(k(\vec{r}-\vec{r}_p))\hat{e}\right) + |\vec{f}_{\mathrm{s}}(k(\vec{r}-\vec{r}_p))|^2 \ . \end{aligned} \tag{4.3.3}$$

Most of the experimental defects on the images are due to spacial illumination variations caused by dust particles. They can be corrected by normalizing the image by the background. In another word, we normalize $I(\vec{r})$ by the intensity of the incident field $I_0 = E_0(\vec{r})^2$ which corresponds to the experimental background.

Experimentally, the background can be measured by different methods. One is to have an empty field of view and the other one, which is more convenient, is to compute the median of a stack of images. Additionally, for the latter to work, the video should be long enough for the particle to diffuse sufficiently. If this condition is not satisfied, a ghost of the particle will appear on the background. Moreover, this process permits getting rid of any immobile particles that could generate any additional noise. Examples of hologram before and after the normalization are shown in Figs.4.3.1 a-c).

Finally, we write the normalized intensity as:

$$\frac{I(\vec{r})}{I_0(\vec{r})} = 1 + 2\Re\left(\vec{f}_{\mathrm{s}}(k(\vec{r}-\vec{r}_p))\hat{e}\right) + |\vec{f}_{\mathrm{s}}(k(\vec{r}-\vec{r}_p))|^2 \tag{4.3.4}$$

Now that we have the analytical form of the holograms' intensity, it is possible to fit an experimental one to Eq. (4.3.4) as shown in Figs.4.3.1 d-e). For the sake of completeness, I will detail the Lorenz-Mie scattering function $\vec{f}_{\mathrm{s}}(k\vec{r})$, which is given by the series:

$$\vec{f}_{\mathrm{s}}(k\vec{r}) = \sum_{n=1}^{n_c} \frac{i^n(2n+1)}{n(n+1)}\left(ia_n\vec{N}^{(3)}_{\mathrm{eln}}(k\vec{r}) - b_n\vec{M}^{(3)}_{\mathrm{oln}}(k\vec{r})\right) \tag{4.3.5}$$

where $\vec{N}^{(3)}_{eln}(k\vec{r})$ and $\vec{M}^{(3)}_{oln}(k\vec{r})$ are the vectorial spherical harmonics, and a_n and b_n are coefficients depending on the particle and illumination properties. For a spherical and isotropic particle of radius a and refractive index n_{p}, which is illuminated by a linearly polarized plane wave, the a_n and b_n coefficients are expressed in terms of spherical Bessel functions j_n and Hankel functions h_n, as [**f˙bohren˙absorption˙1998**]:

$$a_n = \frac{\zeta^2 j_n(\zeta ka)kaj_n'(ka) - j_n(ka)[\zeta kaj_n(\zeta ka)]'}{\zeta^2 j_n(\zeta ka)kah_n^{(1)'}(ka) - h_n^{(1)}(ka)\zeta kaj_n'(\zeta ka)} \ , \tag{4.3.6}$$

and:

$$b_n = \frac{j_n(\zeta ka)kaj_n'(ka) - j_n(ka)\zeta kaj_n'(mka)}{j_n(\zeta ka)kah_n^{(1)'}(ka) - h_n^{(1)}(ka)\zeta kaj_n'(mka)} \ , \tag{4.3.7}$$

where $\zeta = n_\mathrm{p}/n_m$, and where the prime notation denotes differentiation with respect to the argument.

Finally, a hologram is mainly given by the Lorenz-Mie scattering function of Eq. (4.3.5). Moreover, as we can observe in Eqs. (4.3.6) and (4.3.7), a hologram depends on a lot of parameters and variables (λ, n_m, n_p, a and $\vec{r}_\mathrm{p}$) . The parameters can be fitted by comparison to experimental data. In general, the illumination wavelength λ and medium index n_m are known and do not need to be fitted. From only one hologram, one can measure the position $\vec{r}_\mathrm{p}$ precisely of the particle and simultaneously characterize the radius and optical index of the colloid. As a side note, it is even possible to characterize a particle without aprioristic knowledge of its characteristics (*i.e.* radius and refractive index) using a Bayesian approach [**gregory˙bayesian˙2005**, **dimiduk˙bayesian˙2016**].

Computing Eq. (4.3.5) numerically brings another interesting question, as it is analytically written as a sum over n. One could ask after which number n_c of terms the series converges. It has actually been found that the series converges after a number of terms given by [**wiscombe˙improved˙1980**]:

$$n_\mathrm{c} = ka + 4.05(ka)^{1/3} + 2 \ . \tag{4.3.8}$$

Consequently, the holograms of bigger particles require more terms to converge and, hence, are longer to fit. As an example, the largest particles used during my thesis have a radius $a = 2.5$ μm leading to $n_c = 55$ in water and for an illumination wavelength $\lambda = 532$ nm. For the smallest ones, where $a = 0.5$ μm we find $n_c = 18$ which makes a huge difference in practice. Indeed, if each of the terms of the sum takes the same time to be computed; a 2.5 μm particle's hologram is $55/18 \simeq 3$ times longer to be fitted compared to the hologram of a 0.5 μm particle.

If a reader wants to evaluate a hologram given by the Lorenz-Mie theory for a peculiar particle and position, it can be done in a few lines with the `holopy` module utilizing the following Python snippet which was employed to make Fig.4.3.7 and 4.3.8:

```
1 import holopy as hp
```

```python
from holopy.scattering import calc_holo, Sphere

sphere = Sphere(n=1.59, r=1.5, center=(4/0.1, 4/0.1, 10))
# n is the optical index of the particle, r its radius in microns
# center is its center position in microns.

medium_index = 1.33
illum_wavelen = 0.532
illum_polarization = (1, 0)
detector = hp.detector_grid(shape=100, spacing=0.1)
# shape is the size in pixels of the camera and the spacing is the pixel's size in microns.

holo = calc_holo(
  detector, sphere, medium_index, illum_wavelen, illum_polarization, theory="auto"
)
#the hologram can directly be plotted using:
hp.show(holo)
```

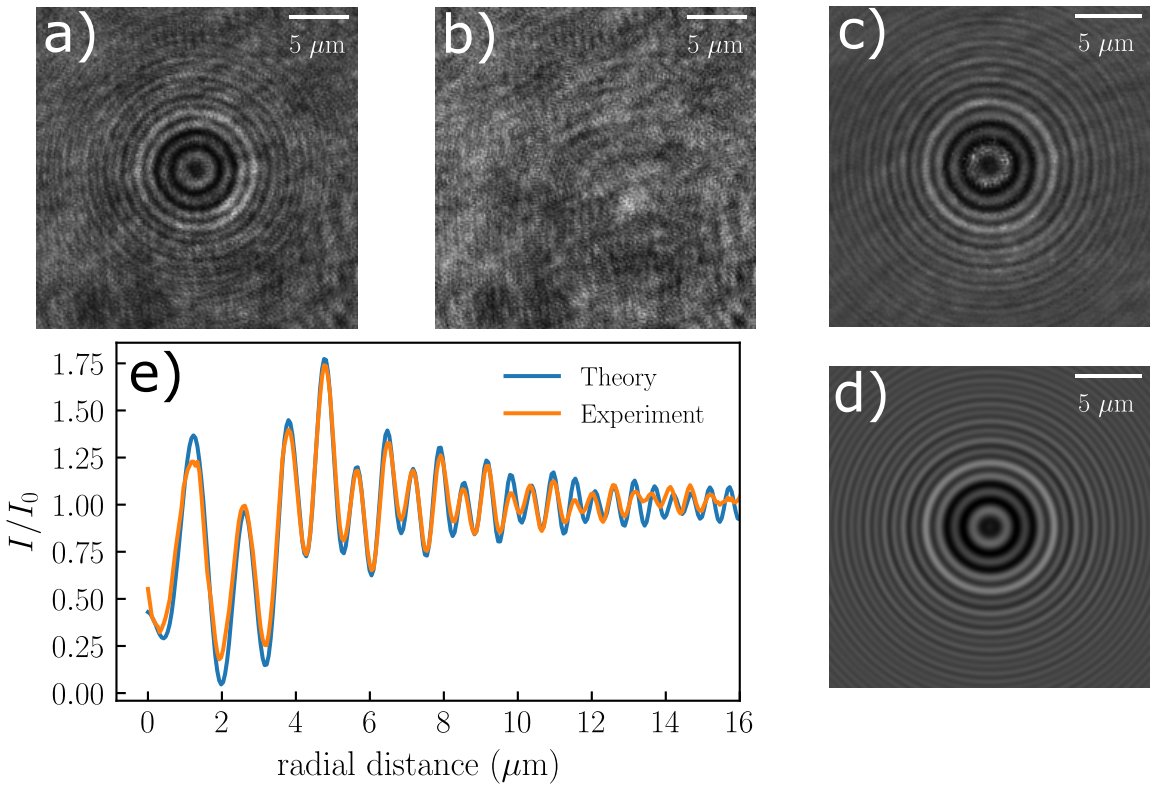


Figure 4.3.1: a) Raw hologram of a 2.5 μm polystyrene particle measured experimentally with the setup detailed in the section 4.5. b) Background obtained by taking the median value of an image time series. c) Normalized hologram given by dividing a) by b). d) Result of the fit of c) using Eq. (4.3.4), from which the particle is found to be at a height $z = 14.77\ \mu$m. e) Comparison of the normalized radial intensities, obtained experimentally from c) and theoretically from d).

4.3.1 Hologram dependence on the particle's characteristics

As we can see with the Eq. (4.3.5), the in-line holograms vary with the position, radius and optical index of the particle. For in-line holograms, as both incident and scattered field are collinear, the x and y positions of the particle are given by the center of the hologram. Thus, it is possible to track the motion of a colloid only in two dimensions by using algorithms such as the Hough transforms to find the center. As a side note, in that case, it would be optimal to place the particle just above the focal plane to have an Airy disk-like hologram, as shown in Fig.4.3.8 for $a = 2.5$ μm and $z = 5$ μm.

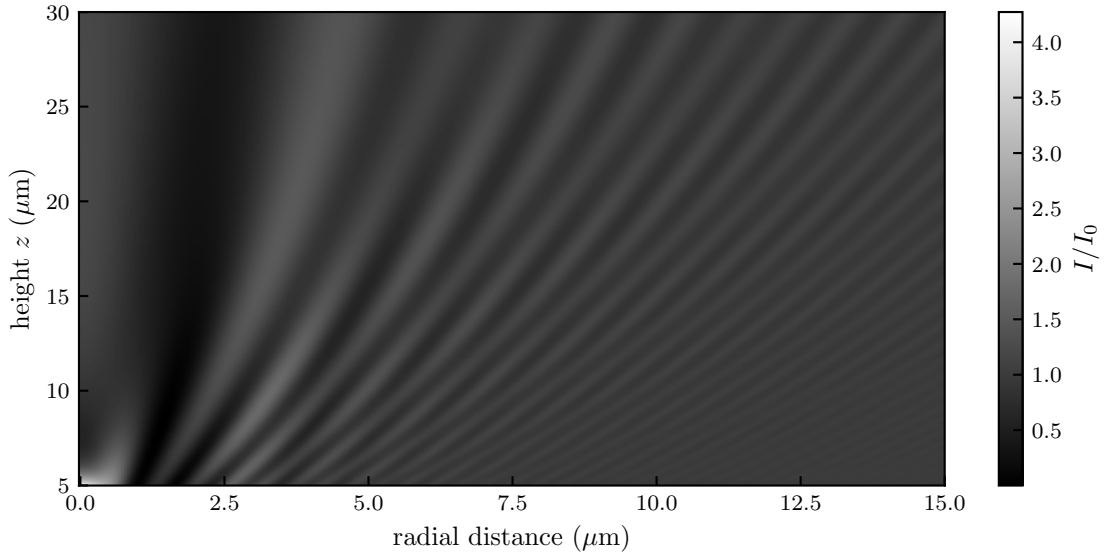


Figure 4.3.2: Hologram intensity map in the (r, z)-plan, calculated (see. Eq. (4.3.4)) with a wavelength $\lambda = 532$ nm for a particle of radius $a = 1.5$ μm and optical index $n = 1.59$.

In order to gain some insights on how the holograms vary with the various parameters, one can compute theoretical (see. Eq. (4.3.4)) holograms for particles of different sizes and heights. We start by considering a particle of radius $a = 1.5$ μm, and optical index $n_{\mathrm{p}} = 1.59$ as shown in Fig.4.3.2. In this case, one can observe that as the distance z between the particle and the focal plane increases, the hologram's rings get wider.

Additionally, this thickening of the rings can also be observed in the Fig.4.3.6, where hologram intensity profiles are plotted as a function of the height z both theoretically and experimentally for a polystyrene colloidal particle of radius $a = 1.5$ μm, and for different couples of parameters in Fig.4.3.8.

Also, we note that if z is not large enough compared to the radius of the particle, the center of a hologram can be so bright that the rings could not be seen if the camera does

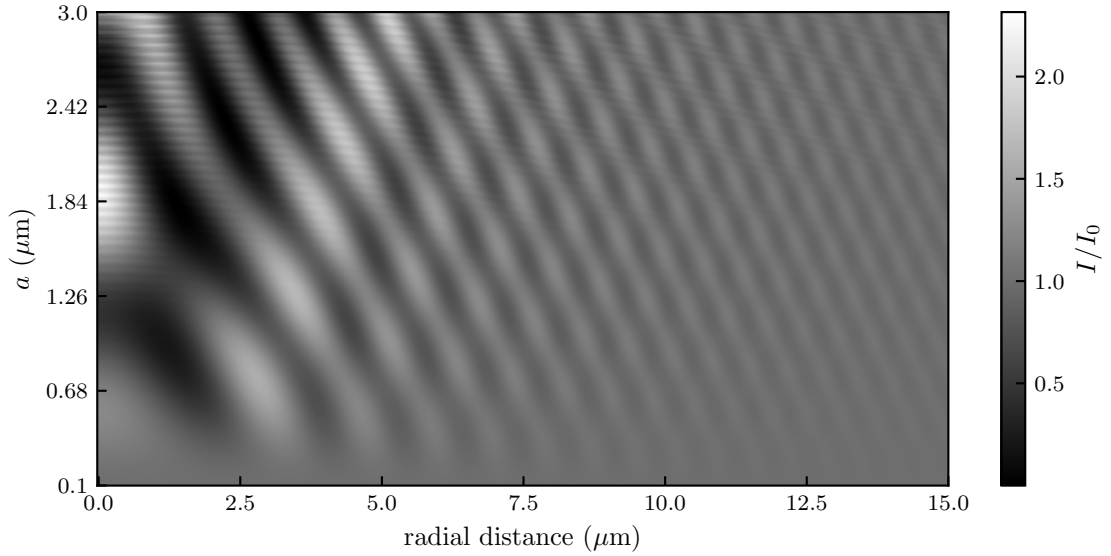


Figure 4.3.3: Hologram intensity map in the (r, a)-plan, calculated (see. Eq. (4.3.4)) with a wavelength $\lambda = 532$ nm for a particle of optical index $n = 1.59$, and a distance $z = 15$ μm between the particle center and the focal plane of the objective lens.

not have a large enough dynamic range. Thus, for having an optimal condition for the fits, one should take care of defocusing enough the objective lens to have $z >> a$.

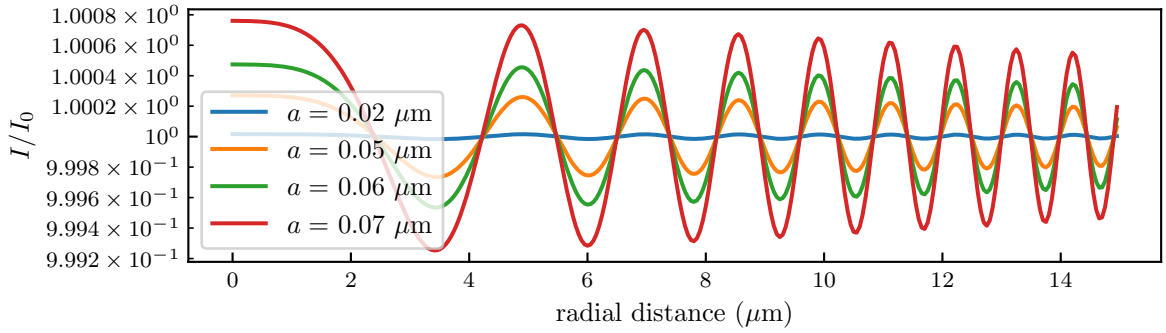


Figure 4.3.4: Radial intensity profile for particle radius $a << \lambda$, and an optical index $n_{\mathrm{p}} = 1.59$ with a distance $z = 15$ μm between the particle center and the focal plane of the objective lens, and for a wavelength $\lambda = 532$ nm.

We can now look at the holograms variation with respect to the radius of the particle as shown in Fig.4.3.3 for a particle of optical index $n = 1.59$ and at a distance $z = 15$ μm for a wavelength $\lambda = 532$ nm. One can observe that for small particles compared to the wavelength, *i.e.* $a << \lambda$, we do not observe the rings. This is due to the fact that for the small particles, the scattering can be approximated using the Rayleigh theory in which the scattering is isotropic. Thus, the variation of intensity around I_0 will be smaller for smaller particles.

Also, in this small-particle regime, the particle size does not affect the general shape of the hologram but just its intensity as shown in Fig.4.3.4, for particles of radii between $a = 0.02$ μm and $a = 0.07$ μm, and for a wavelength $\lambda = 532$ nm.

Additionally, since the signal-to-noise ratio is lower than for larger particles, it is less precise to characterize small colloids compared to the wavelength.

As the particle gets bigger, the scattering becomes anisotropic and is mostly oriented towards the incident plane-wave direction. This effect leads to an increase of the amplitude I/I_0 of the rings, as one can see in Fig.4.3.3. Thus, the signal-to-noise ratio is high enough to easily discern the hologram on top of the noise as one can see on the experimental picture of Fig.4.3.1-a). One who wants to employ this method should thus use large enough particles for the hologram intensity to be greater than the camera noise level.

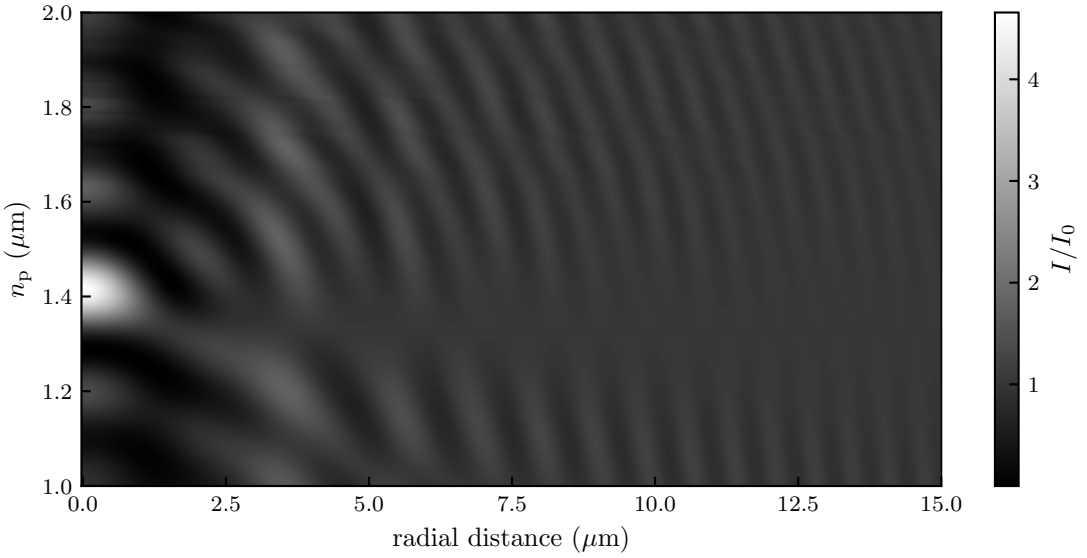


Figure 4.3.5: Hologram intensity map in the (r, a)-plan, calculated (see. Eq. (4.3.4)) with a wavelength $\lambda = 532$ nm, for a particle of optical index $n = 1.59$, and a distance $z = 15$ μm between the particle center and the focal plane of the objective lens.

Finally, one can check how the holograms are varying with the optical index of a particle. In this case, it is not the particle's optical index n_p which matters the most but the ratio $\zeta = n_\mathrm{p}/n_\mathrm{m}$ which can be found in the a_n and b_n formulas, in Eqs.4.3.6 and 4.3.7. Indeed, for the scattering to happen, the optical index n_p of the colloid needs to be different from the optical index n_m of the surrounding medium. Additionally, the numerical solution of the Lorenz-Mie framework needs precaution for $n_\mathrm{p} \simeq n_\mathrm{m}$ [**wiscombe˙improved˙1980**, **lentz˙generating˙1976**]. In Fig.4.3.5, we can observe holograms of a particle of radius $a = 1.5$ μm at fixed distance $z = 15$ μm between the particle and the focal plane of the ob-

jective lens with a varying colloid's optical index, in water where $n_\mathrm{m} = 1.33$. In Fig.4.3.5, one can thus observe that for $n_\mathrm{p} \simeq n_\mathrm{m}$ we do not see any holograms. Additionally, one can observe that the signal-to-noise ratio gradually increases as n_p becomes different from n_m. One who wants to use this technique should thus have n_m different enough from n_p for the hologram intensity to be greater than the camera noise level.

4.3.2 Summary on the Lorenz-Mie method

A given set of height, optical index and radius of a colloid thus gives unique holograms. Conversely, this uniqueness of the holograms permits precise extraction of the position, optical index and radius of a colloid. Holograms for different sets of parameters are shown in Figs. 4.3.7 and 4.3.8. Additionally, the interested reader can use the Jupyter Notebook on my Github repository in order to plot any hologram .

Finally, the Lorenz-Mie framework provides the most versatile in-line holographic method. Indeed, it permits tracking and characterizing unique particles even without a priori knowledge on its characteristics. Besides, it is possible to write the Lorenz-Mie function $\vec{f_\mathrm{s}}$ for particular cases such as anisotropic particles [**fung˙holographic˙2013**, **wang˙using˙2014**] or particle clusters [**fung˙holographic˙2013**, **perry˙real-space˙2013**] to name a few. Such possibilities pave the way to a lot of experimental studies. Additionally, the method allows reaching a really high precision as the tenth of nanometers on the position and radius as well as 10^{-3} on the optical index [**lee˙characterizing˙2007**].

Unfortunately, the Lorenz-Mie framework suffers from a major drawback which is the time needed to fit one image. For example, a 200 by 200 pixels image, of a 2.5 μm particle's hologram, can take up to two minutes to be fitted using a pure and straightforward Python algorithm. A lot of work has been done to permit faster tracking, such as random-subset fitting [**dimiduk˙random-subset˙2014**], GPU (graphical processing units) acceleration, machine learning [**yevick˙machine-learning˙2014**, **hannel˙machine-learning˙2018**] and deep neural networks [**altman˙catch˙2020**].

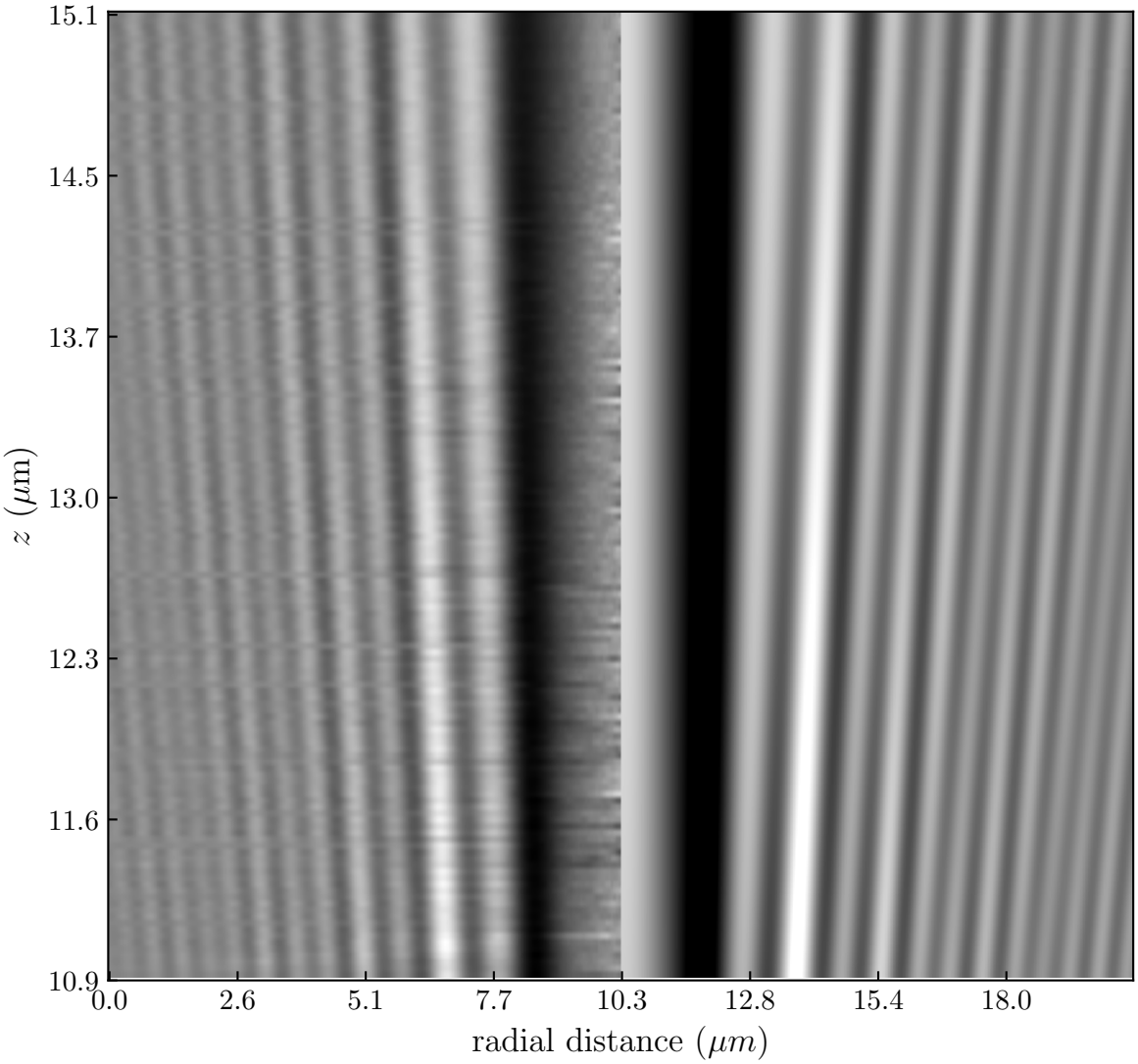


Figure 4.3.6: Hologram intensity map in the (r, z)-plan, calculated (see. Eq. (4.3.4)) with a wavelength $\lambda = 532$ nm, for a particle of optical index $n = 1.59$, and radius $a = 1.51$ μm using the experimental setup presented in the section 4.5. On the right, the corresponding theoretical intensity using the result of each individual hologram's fit to Eq. (4.3.5).

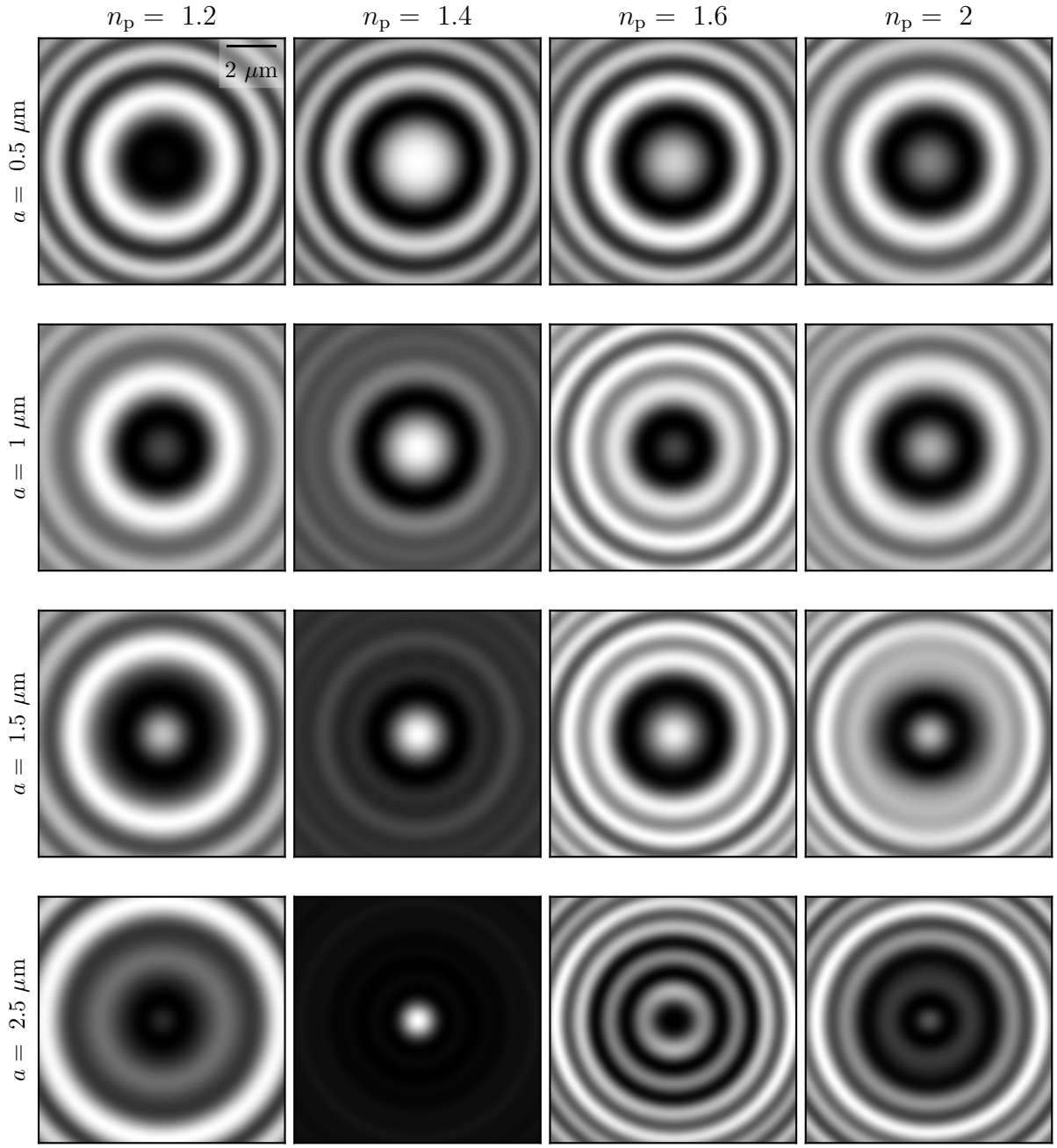


Figure 4.3.7: Holograms calculated (see. Eq. (4.3.4)) with a wavelength $\lambda = 532$ nm, for different set of parameters (a, n_p), and for a distance $z = 15$ μm between the particle center and the focal plane of the objective lens.

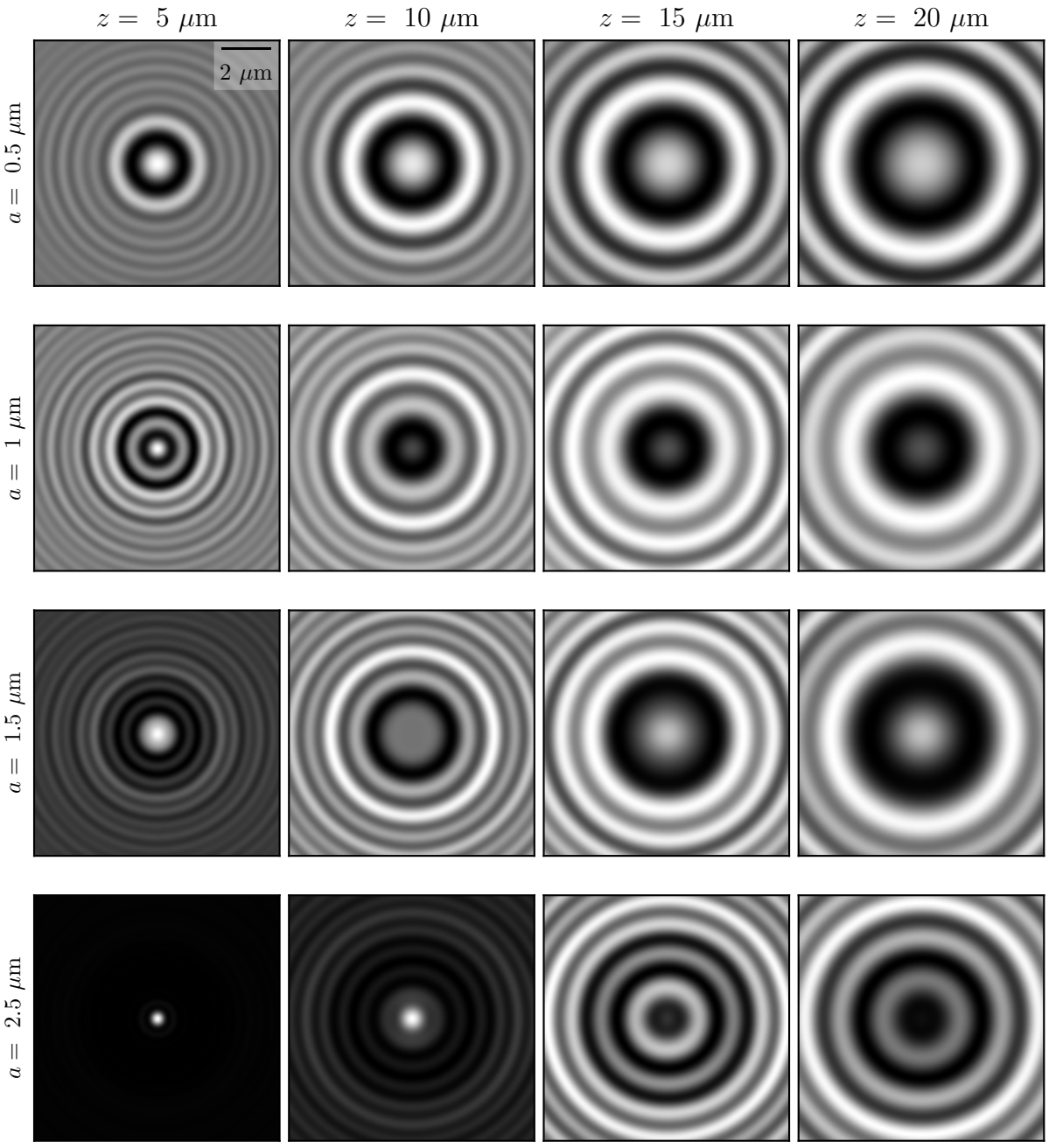


Figure 4.3.8: Holograms calculated (see. Eq. (4.3.4)) with a wavelength $\lambda = 532$ nm, for different set of parameters (a, z), and for an optical index $n_\mathrm{p} = 1.59$.

4.4 Rayleigh-Sommerfeld back propagation

Rayleigh-Sommerfeld back propagation [**wilson˙3d˙2012**] works on the same principle as the Lorenz-Mie scattering but assumes small scatterers, and, a low difference of optical indices, such that:

$$|\zeta - 1| << 1 \text{ and } ka|\zeta - 1| << 1 \ . \tag{4.4.1}$$

In this case, at the focal plane, the intensity of the scattered field is smaller than the intensity of the incident field, hence, the term $|\vec{E}_\mathrm{s}|^2$ can be ignored. Thus, the Eq. (4.3.4) can be rewritten as:

$$\frac{I(\vec{r})}{I_0(\vec{r})} = 1 + 2\Re\left(\frac{E_\mathrm{s}(\vec{r},0)}{E_0(\vec{r})}\right) \ . \tag{4.4.2}$$

If one can retrieve the scattered field completely from an image, it is possible to reconstruct it above the focal plane by convolution using the Rayleigh-Sommerfeld propagator [**goodman˙introduction˙2005**]:

$$h_{-z}(\vec{r}) = \frac{1}{2\pi}\frac{\partial}{\partial z}\frac{\mathrm{e}^{ikR}}{R} \ , \tag{4.4.3}$$

where $R^2 = r^2 + z^2$ and the sign convention on the propagator indicates that the particle is above the focal plane. Using this propagator we have:

$$E_\mathrm{s}(\vec{r},z) = E_z(\vec{r},0) \otimes h_{-z}(\vec{r}) \ . \tag{4.4.4}$$

By using the convolution theorem [**cheong˙strategies˙2010**, **goodman˙introduction˙2005**, **sherman˙application˙1967**, **schnars˙digital˙1994**] and supposing a uniform illumination, one can approximately reconstruct the scattered field at height z as:

$$E_\mathrm{s}(\vec{r},z) \approx \frac{\mathrm{e}^{ikz}}{4\pi^2}\int_{-\infty}^{\infty} B(\vec{q})H(\vec{q},-z)\mathrm{e}^{i\vec{q}\cdot\vec{r}}\mathrm{d}^2q \ , \tag{4.4.5}$$

where $B(\vec{q})$ is the Fourier transform of I/I_0 and where:

$$H(\vec{q}, -z) = \mathrm{e}^{iz\sqrt{k^2-q^2}} \ . \tag{4.4.6}$$

Finally, using Eq. (4.4.5) one can reconstruct the scattered field and intensity since $I(\vec{r}) = |E_s(\vec{r})|^2$, as shown in Fig.4.4.1. Moreover, by finding the position where we have an inversion of the center from bright to dark in Fig.4.4.1, we measure the position of the particle. These equations are way less computationally expensive than Eq. (4.3.5). Thus tracking can be faster.

Additionally, as Eq. (4.4.5) takes only into account the intensity of the image, this method does not require any information on the particle and number of particles. As a matter of fact, to write Eq. (4.4.5), one just needs to assume that we have spherical colloids. Thus, this method is powerful to reconstruct the 3D position of a lot of particles or clusters. However, the Rayleigh-Sommerfeld back propagation suffers from being less precise of the presented methods and we cannot use it to characterize the particles generating the holograms.

4.4.1 Numerical Rayleigh-Sommerfeld back propagation

The `holopy` Python module provides a set of methods that permit implementing the Rayleigh-Sommerfeld back propagation. Given the `hologram` variable containing all the needed metadata about the hologram such as the pixel size, medium index n_n, illumination wavelength λ. Then, one can use the `propagate` method to back propagate a hologram over a set `zstack` of height using the following Python snippet.

```
1 import holopy as hp
2 import numpy as np
3
4 zstack = np.linspace(0, 20, 11)
5 rec_vol = hp.propagate(holo, zstack)
```

Note that using the `propagate` function, each propagation is done by performing a convolution of the reference hologram over the distance to be propagated. However, better reconstruction can be obtained iteratively by propagating holograms over several short distances. The latter method is called Cascaded Free Space Propagation, and is particularly useful when the reconstructions have fine features or when propagating over large distances [**kreis˙frequency˙2002**]. It can be done by specifying the argument `cfsp` to

the `propagate` method. For example, to change the source of the propagation every three steps, one can use `hp.propagate(holo, zstack, cfsp=3)`.

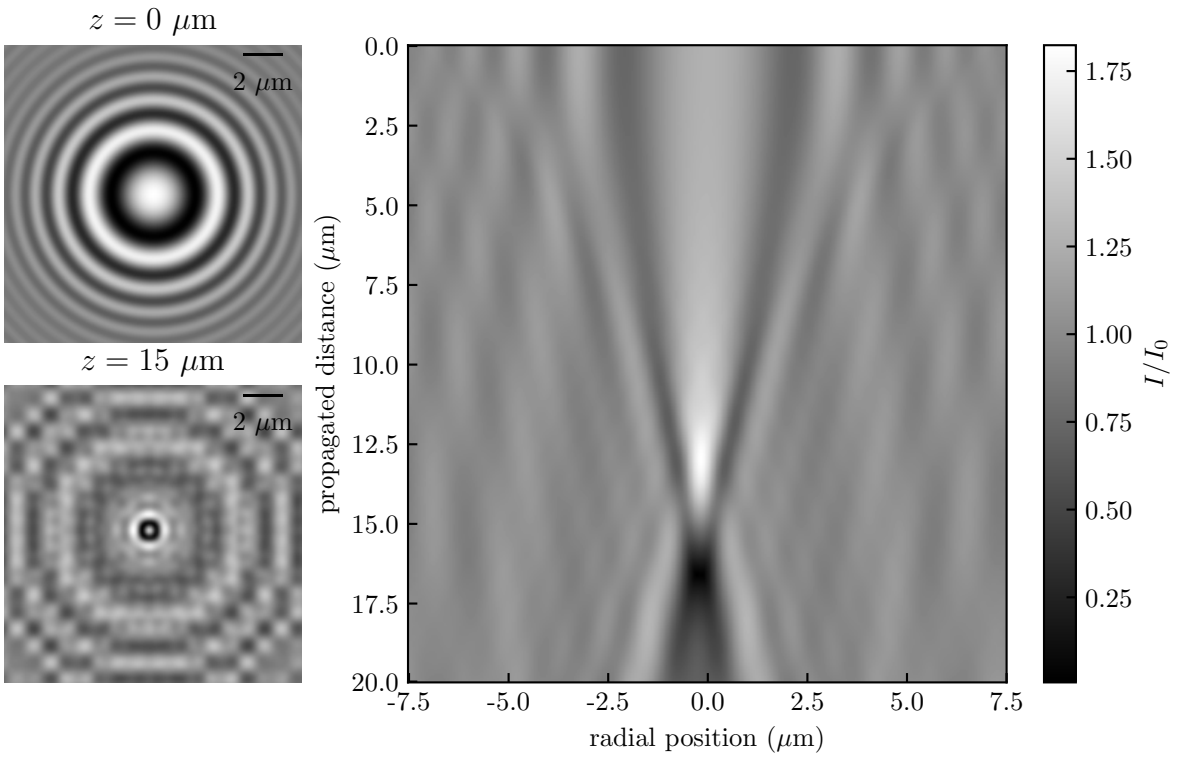


Figure 4.4.1: On the left: the original hologram on the top and propagated along 15 μm on the bottom. On the right: reconstruction using Eq. (4.4.5) of the scattered intensity by a single colloidal sphere of radius $a = 0.1$ μm, with optical index $n_\mathrm{p} = 1.59$ in water whose index is $n_\mathrm{m} = 1.59$, and for a height of 15 μm.

4.5 The experimental setup

The experimental setup I developed during my PhD can be employed to implement both the Lorenz-Mie and Rayleigh-Sommerfeld back propagation methods. In order to observe the holograms, we use a homemade inverted microscope as shown in Fig.4.5.1 and schematized in Fig.4.5.2. This microscope is built using a ThorLabs cage system. Using the microscope, we observe the holograms resulting from the interactions between a laser source and the beads present in a sample.

A sample consists of a parallelepipedic chamber (1.5 cm $\times$ 1.5 cm $\times$ 150 μm), made from two glass covers, a parafilm spacer, and sealed with vacuum grease, containing a dilute suspension of spherical polystyrene beads. Sealing the sample with vacuum grease permits to drastically decrease evaporation, which reduces the possible evaporation driven-flow in the sample.

We used 3 different colloidal sizes, of nominal radii 0.56 μm, 1.5 μm and 2.5 μm, at room temperature T, in distilled water (type 1, MilliQ device) of dynamic shear viscosity $\eta = 1$ mPa.s. The particles are made of polystyrene of density $\rho = 1050$ kg.m^{-3} and optical index $n_{\mathrm{p}} = 1.598$ at a wavelength $\lambda = 532$ nm. The particle we mostly used are the Polybead® 17134-15 1.5 μm microsphere that are packaged in aqueous suspension with minimal surfactant. These particles contain a slight anionic charge from sulfate ester. Thus, the surface charge density of the colloids remain constant as a function of the pH, this would not be the case if the particle surface charge were due to carboxyl head groups [**behrens˙charge˙2001**].

The sample is illuminated by a collimated laser beam with a 532 nm wavelength (Laser diode CPS532, spectral width of the nanometer). The laser used delivers a power of 4 mW and has a waist of 3.5 mm. Since the laser is collimated, it has a near-zero exentricity so that it can be seen as a plane wave. As presented in the section 4.3, the light scattered, by one colloidal particle at a given time t, interferes with the incident beam.

An oil-immersion objective lens (Olympus, plan apochromat, x60 magnification, 1.30 numerical aperture) collects the resulting instantaneous interference pattern, and relays it to a camera (Basler acA1920-155um) with a 51.6 nm/pixel resolution (see Fig.4.3.1-a)). The exposure time of the camera is set to $\tau_{\mathrm{expo}} = 3$ ms to avoid motion-induced blurring of the image. As a general rule, the particle should not diffuse more than the pixel size during that time, such that $\sqrt{2D\tau_{\mathrm{expo}}} < 51.6$ nm.

4.6 Optical forces

As we illuminate particles with the laser light, it is important to know if the optical forces that arise from the interactions between the light and the particle needs to be taken into account. When a plane wave is incident on a sphere, it scatters and absorbs light. This process depends both on the light wavelength λ and on the sphere properties, its radius a and refractive index $n_{\mathrm{p}} = n_r - jn_i$. For polystyrene $n_i \ll 1$, as shown in Fig.4.6.1 such that we neglect it in the Lorenz-Mie framework. However, computing the optical forces, and hence, the light quantity which is absorbed requires n_i, so we consider it in this section. Additionally, in the Mie theory, the particle is characterized by the size parameter $\tilde{x} = 2\pi a/\lambda$. The optical force F_{opt} is given by [**f˙bohren˙absorption˙1998**]

$$F_{\mathrm{opt}} = \frac{I_{\mathrm{r}} n_{\mathrm{m}} \pi a^2}{c} (Q_{\mathrm{ext}} - g Q_{\mathrm{sca}}) \ , \tag{4.6.1}$$

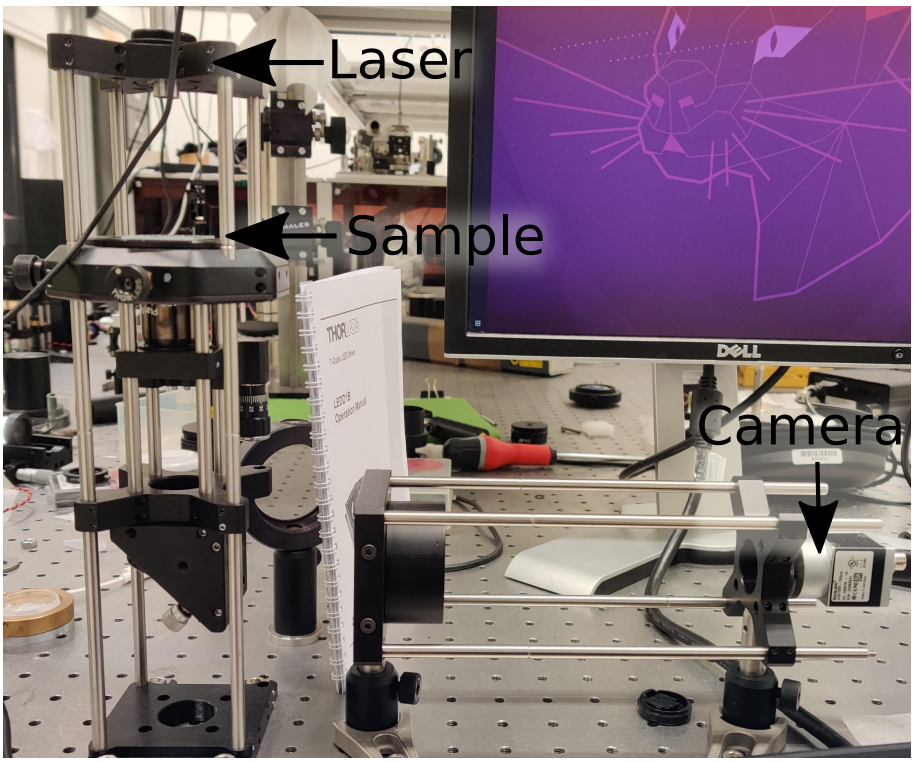


Figure 4.5.1: Photo of the custom-built microscope developed in my thesis. It is composed of a Thorlabs cage system. The camera used is a Basler acA1920-155um. We use a x60 magnification and 1.30 numerical aperture oil-immersion objective lens. The light source is a collimated $\lambda = 521$ nm wavelength laser.

where I_r is the irradiance in W.m^{-2} on the sphere, c is the speed of light in vacuum, $g = \pi a^2$ the sphere cross section and Q_{ext} and Q_{sca} being respectively the extinction and scattering efficiency given by:

$$Q_{\text{ext}} = \frac{2}{k^2 a^2} = \sum_{n=1}^{\infty} (2n+1)(|a_n|^2 + |b_n|^2) \ , \tag{4.6.2}$$

and,

$$Q_{\text{ext}} = \frac{2}{k^2 a^2} = \sum_{n=1}^{\infty} (2n+1)\Re(a_n + b_n) \ , \tag{4.6.3}$$

where the a_n and b_n coefficient are given by the Eqs.4.3.6 and 4.3.7 respectively.

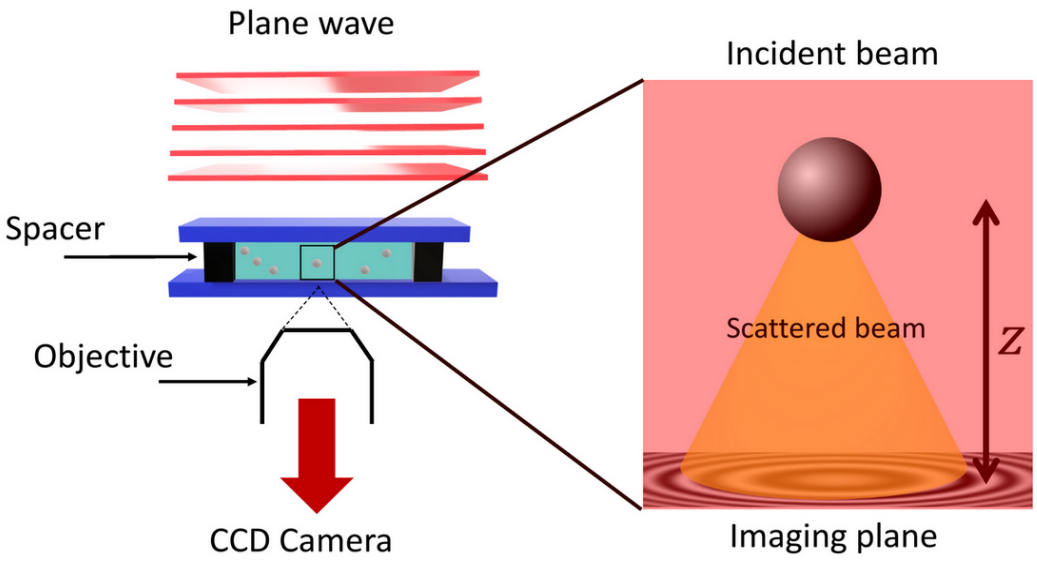


Figure 4.5.2: Schematic of the experimental setup. A laser plane wave of intensity I_0 illuminates the chamber containing a dilute suspension of microspheres in water. The light scattered by a particle interferes with the incident beam onto the focal plane of an objective lens, which magnifies the interference patten and relays it to a camera.

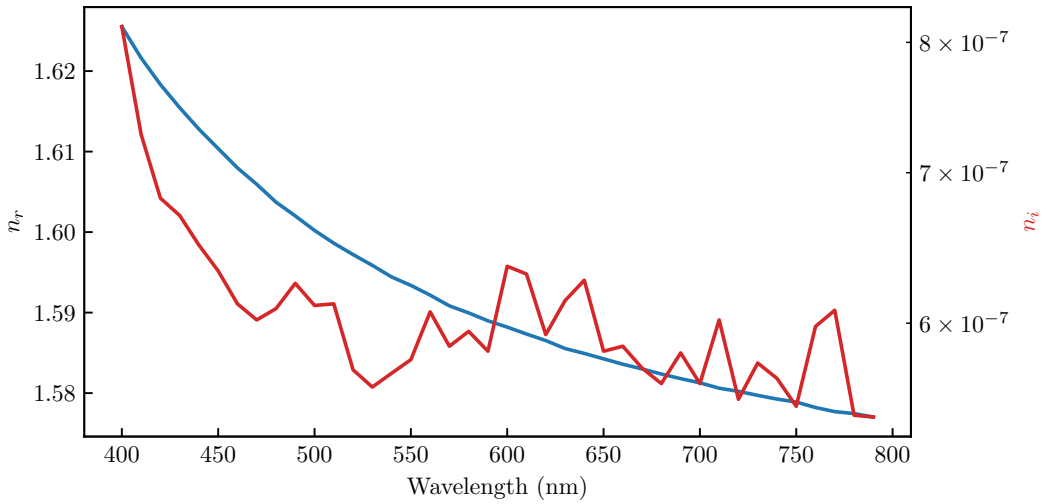


Figure 4.6.1: Real (left axis) and imaginary (right axis) part of the refractive index of polystyrene as a function of the incident wavelength. Data obtained from [**zhang˙complex˙2020**].

To compute the optical force I personally employ the `miepython` python's module and retrieving the optical index data from the refractiveindex.info website, using the following Python snippet.

```python
import miepython as mp
import numpy as np

# Download the data on the refractiveindex.info website
poly = np.genfromtxt(
        r"https://refractiveindex.info/tmp/data/organic/(C8H8)n%20-%20polystyren/Zhang.txt",
        delimiter="\t",
)
N = len(poly) // 2
poly_lam = poly[1:N, 0]  # wavelength
poly_nre = poly[1:N, 1]  # real part
poly_nim = poly[N + 1 :, 1]  # imaginary part

x = 2 * np.pi * a / poly_lam
n = poly_nre - 1.0j * poly_nim
qext, qsca, qback, g = mp.mie(n, x)  # compute the efficiencies
E0 = 4.5e-3 / (np.pi * 1.75e-3 ** 2)  # compute the irradience
c = 299792458 / 1.33  # light velocity in the  medium

F = E0 * np.pi * r0 ** 2 * (qext - g * qsca) / c  # compute the optical force
```

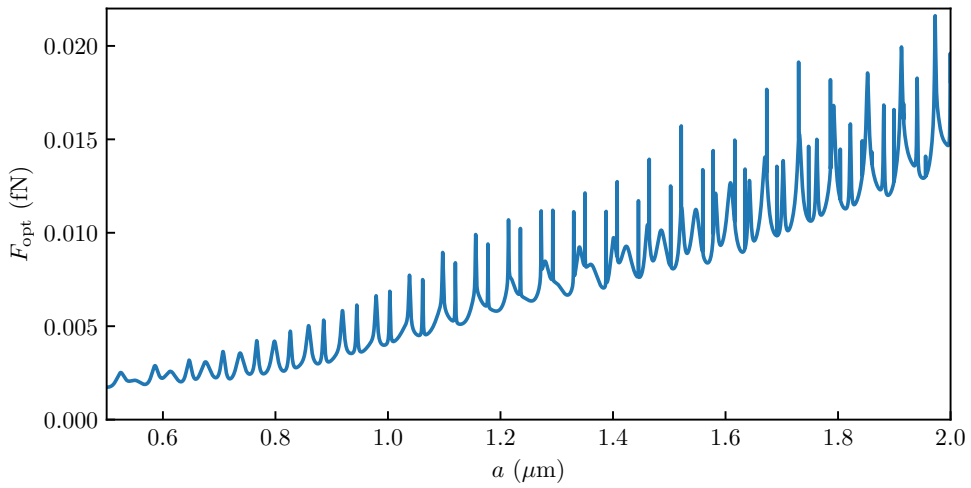


Figure 4.6.2: Optical force F_{opt} (see Eq. (4.6.1)) exerted on a spherical particle of radius a by a plane wave of wavelength $\lambda = 532$ nm, and of irradiance $I_{\text{r}} = 467.7$ W.m^{-2}. The optical force is calculated employing the `miepython` python's module and the refractive index of polystyrene [**zhang˙complex˙2020**].

Using Python, one can thus compute the optical force F_{opt} as a function of the particle radius. As shown in Fig. 4.6.2 F_{opt} is of the order of 10^{-2} fN. By comparing this result with the gravitational force:

$$F_{\mathrm{g}} = \frac{4}{3}\pi(\rho_{\mathrm{m}} - \rho_{\mathrm{p}})g \ , \tag{4.6.4}$$

we found that for a particle of radius $a = 0.5$ μm, $F_{\mathrm{g}} = -0.3$ fN, while for $a = 1.5$ μm, $F_{\mathrm{g}} = -7$ fN. Thus, we can conclude that $F_{\mathrm{opt}} \ll F_{\mathrm{g}}$ for the range of radii we used. Hence, in the following we neglect the optical forces as they are lower than the other external forces acting on the colloids. However, experiments with a controlled irradiance (up to 10^6 W.cm^{-2}) were conducted in [**prieve˙measurement˙1999**] and the authors were able to measure the optical forces from the motion statistics. Additionally, in Fig. 4.6.2 we observe resonances, these happen when the denominator of the coefficients a_n or b_n (see Eqs. (4.3.6) and (4.3.7)) goes to zero, this happens when $2a \approx \lambda/in_{\mathrm{p}}$, with i and integer. Moreover, since this resonances are localized and highly depend on the particle radius a and optical index n_{p}, they have been used to characterize particles with high accuracy.

4.7 The experimental procedure

We implement here the Lorenz-Mie fitting method, since it permits the characterization of single particles. Indeed, since we are interested in fine effects near surfaces, we require to know perfectly the radius of the tracked particle. This feature also makes our whole procedure calibration-free, as we do not need to assume any physical properties. An example of the procedure that permits tracking a single-particle trajectory is provided in appendix A.2. In the following, the different steps of the procedure are described.

4.7.1 Recording the holograms

Since we use a Basler camera, we use the provided computer program (Pylon) by the manufacturer in order to record the holograms. The software permits adjusting the parameters of the camera, such as the region of interest (ROI),number of frames per second (fps) or the opturation time to name a few. Also, video can be recorded as a time series of images, in AVI or MP4 formats. AVI files or times series are a great way to save the video since it is lossless. However, in general, we use a ROI of 1000×1000 pixels to record the particle during a long-enough time.

Additionally, since the recording is done using 8 bits per pixel (or 256 gray levels), an

image of 1000×1000 pixels needs a disk space of 1 MB [2]. One can see that image sequences and AVI files are not suitable for our case because i) at 100 fps one would need 108 GB to store a 30-minute film, which would lead to several TB of data per experiment which is not manageable; ii) it would require a sequential writing speed of 60 MB/s, which is just below the limit of the better Hard Drive Disks. iii) AVI files are bound to 2 GB maximum thus dramatically reducing the length of the experiments.

To conclude, for all of these reasons, we chose to use the MP4 file format (MPEG-4 encoding) for the video recording. Using the lowest compression, we did not observe any impact on the fitting process due to quality loss. Finally, a video of 30 minutes has an approximate size of 3 GB.

4.7.2 Fitting the holograms

Once the holograms are recorded, we fit all of the images to retrieve the trajectory of the particle. To do so, we chose to use the `pylorenzmie` module developed by the Grier's lab at New York University. Although this module presents a lot of capabilities, it is not adapted to MP4 input. Thus, I developed a wrapper[3] around `pylorenzmie` that I called Wraplorenzmie which can be found on my Github repository .

This wrapper permits to directly load the MP4 files, compute the background and choose if what parameters should be fitted. Also, it manages the process fitting a time series of images by using results of previous image as initial fit parameters.

However, as presented in the section 4.3, the main drawback is the time to fit an image. Indeed, using a Python algorithm, one needs 30 seconds to fit images of 100×100 pixels and a few minutes for a 500×500 pixels hologram. We can directly see a bottleneck, if one wants to track one trajectory made of 100 000 images. In such case, one would need to typically 70 days for a series of images that needs only a few minutes to be recorded experimentally.

When I started my PhD, two groups, the Grier's lab and the Manoharan's lab, had

[2] An uppercase B denotes Byte which is equivalent to 8 bits denoted by b, *i.e.* $1\ B = 8\ b$. For storage indications, Bytes are generally used, since historically a set of 8 bits encodes a single text character, and are for this reason the smallest addressable memory units in most computer architectures. As an example, in binary "LOMA" would be encoded by "01001100 01001111 01001101 01000001."

[3] A wrapper is a code that encapsulates or "wraps" another code to make it easier to use. For example, it is particularly useful to adapt a program to a particular type of input data. Creating wrapping function is commonly done by developers. In the end, wrapping adds some abstraction and readability to the source codes.

already created Python packages, respectively Pylorenzmie and Holopy, in order to inverse holograms. They had introduced ways to only fit a set of randomly chosen pixels, and demonstrated that taking only 1 % of the image pixels, could lead to similar precision thus improving considerably the fit's execution time [**dimiduk˙random-subset˙2014**].

Unfortunately, even if fitting a random subset of pixels is faster, it leads to a few images per second, and is still too long for the amount of data we want to have. This part of my thesis is certainly the one where I spent the longest time, and I learned a lot about code optimization and computer cluster usage.

In the middle of my thesis, `pylorenzmie` got a new commit[4] on the authors' Github repository which was saying that the authors succeeded in using GPU acceleration with CUDA[5]. This was not an easy task since they needed to reconstruct the Bessel functions in an understandable way for the GPU. Fortunately, it is possible to do so by using continued fractions [**lentz˙generating˙1976**]. This appreciable update permits fitting whole images with a speed improvement of 20 fps. At this speed, we precisely extract the three dimensional position of the particle, as well as the radius and optical index.

As a remark, the fits are done by solving a least-squares problem using the Levenberg-Marquardt algorithm [**more˙levenberg-marquardt˙1978**]. This algorithm is largely used in curve-fitting applications due to its capabilities to find a minimum even by starting far from it. As mentioned, it is also possible to use various models of Machine Learning or Deep Learning to do the fits [**altman˙catch˙2020**]. However, since we can write analytically the holograms, the Deep Learning's models cannot be more accurate than a standard least-square-fitting process. Deep Learning, however, could be a great option is one wanted to prioritize the computation time over the fit's precision.

Finally, to have a more reliable and fast-tracking, we begin by fitting the first 10 000 images with $\vec{r}_\mathrm{p}$, a and n_p as free parameters. Using the results of this fit, we can characterize the physical properties of the tracked colloid with high precision. Then, using these results we can then fit image with only the position $\vec{r}_\mathrm{p}$ as a free parameter.

[4] A commit is an update of the files on a Git repository.

[5] CUDA is the acronym of Compute Unified Device Architecture. It is a parallel computing platform, and programming model made to permit an easier use of the GPU for general purposes. CUDA is developed by NVIDIA since 2012, thus all recent NVIDIA's GPUs are CUDA-enable. It is possible to use CUDA with every language as long as a library has been developed, such as `cupy` for Python.

4.7.3 Radius and optical index characterization

Using 10 000 measurements of a and n_p one can do a 2D histogram, as presented in Fig.4.7.1 here smoothed using a Gaussian Kernel Density Estimator. Doing so, we determine that the radius of the observed particle is $a = 1.514 \pm 0.003$ μm and its optical index is $n_\mathrm{p} = 1.585 \pm 0.002$.

Finally, as explain above, using this measurement of the radius and optical index, we then fit the whole video by removing them from the free parameters. Doing so, we measure the 3D trajectory of the particle as shown in Fig.4.7.2 in three dimensions for the particle previously characterized.

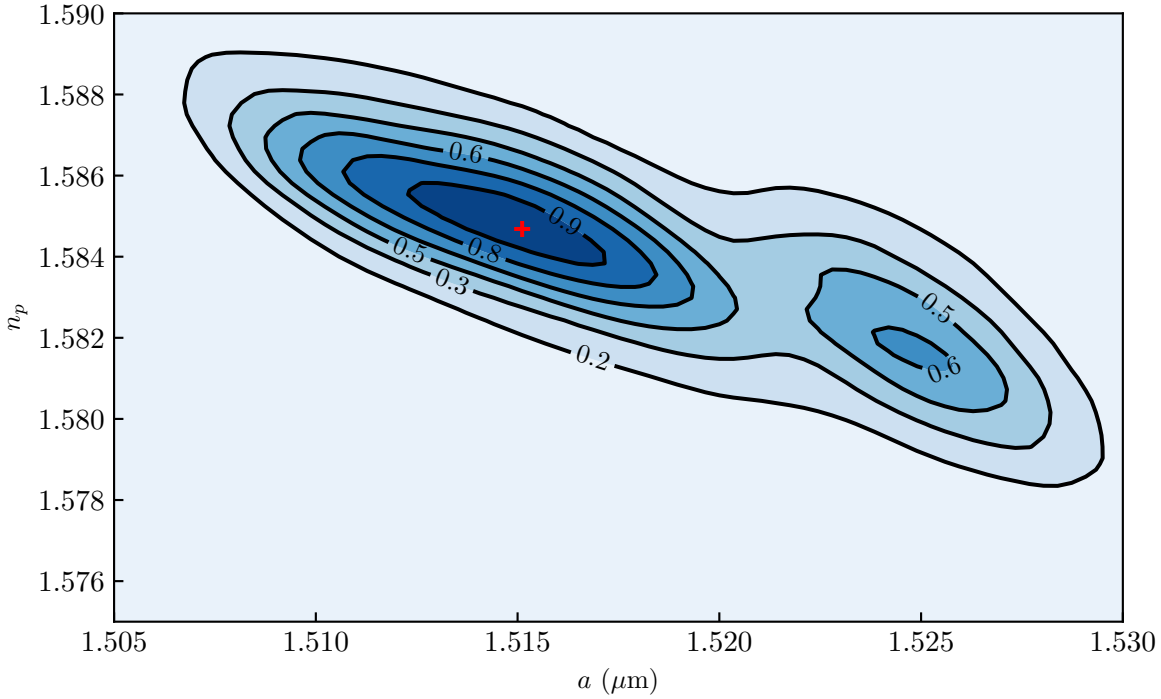


Figure 4.7.1: 2D Probability density function of the measurements of the optical index n_p and radius a. Black lines indicate iso-probability. Taking the 10% top probability, we measure $n_\mathrm{p} = 1.585 \pm 0.002$ and $a = 1.514 \pm 0.003$ μm.

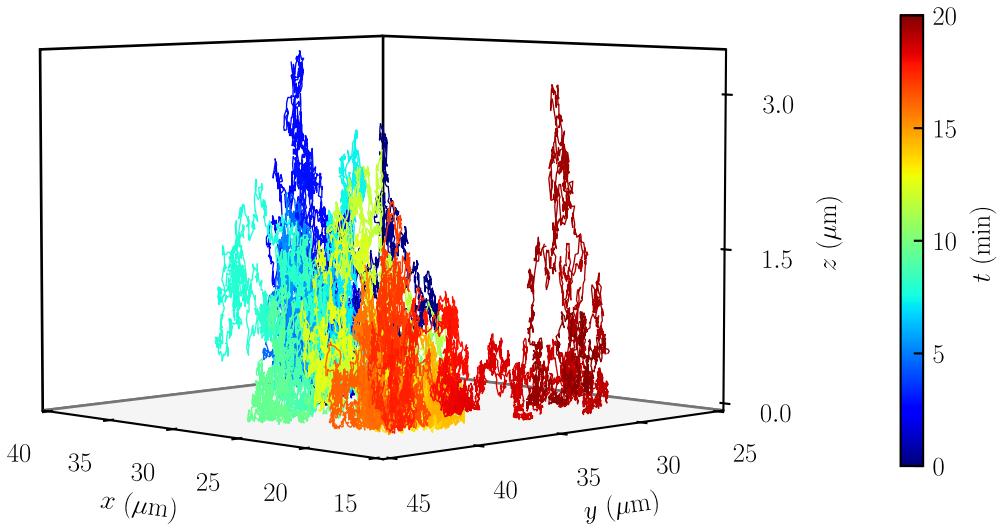


Figure 4.7.2: 3D plot of an experimental trajectory measured in water for a particle of optical index $n_{\mathrm{p}} = 1.585$ and radius $a = 1.514$ μm.

4.7.4 Test case scenario: a sedimenting particle

In order to test the Lorenz-Mie framework, we consider a well-known problem: a sedimenting particle. We study a polystyrene particle of commercial radius $a = 1.5$ μm sedimenting in water. The particle tracking is done over a minute using a frame rate of 100 fps. The time over which we can track the colloid is bound by the terminal sedimentation velocity v_{sed}. Indeed, the particle, should not be too far from the imaging plane, otherwise we do not capture enough light intensity, and not too close as discussed in section 4.3. To ensure that we can correctly record and fit the holograms, we experimentally find that with our experimental setup the particle height z should be in a working range of 10 to 30 μm. This working range could be greatly increased by using high-sensitivity cameras. The trajectory of the sedimenting colloid is shown Fig. 4.7.3, moreover, we measure its radius $a_{\mathrm{sed}} = 1.45 \pm 0.03$ μm and optical index $n_{\mathrm{sed}} = 1.59 \pm 0.01$.

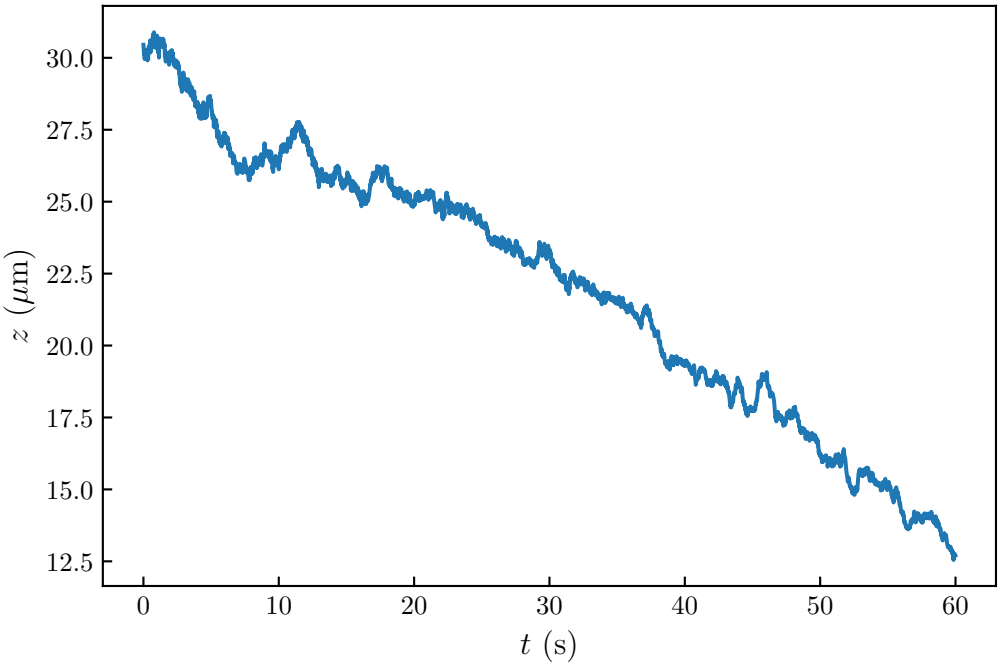


Figure 4.7.3: Experimental trajectory of a polystyrene particle of radius $a = 1.5\mu$m sedimenting in water, along the z-axis — *i.e.* the axis where the gravity acts.

The terminal sedimentation velocity can be found by equaling the drag force (*i.e.* the Stokes's drag) to the gravitational force. Doing so, the terminal velocity of a sedimenting colloid writes:

$$v_{\text{sed}} = \frac{2}{9}\frac{\Delta\rho a^2 g}{\eta} \ , \tag{4.7.1}$$

where $\Delta\rho$ is the density difference between the fluid and colloid density, and g the gravitational acceleration. For the tracked particle, we calculate $v_{\text{sed}} = 229$ nm.s^{-1} using the radius measured from the holograms and tabulated $\Delta\rho = 50$ for polystyrene in water. To measure the latter velocity from the trajectory, we use the MSD which writes by taking into account the sedimentation:

$$\langle \Delta z^2 \rangle = 2D_0\Delta t + v_{\text{sed}}^2\Delta t^2 + (2\xi)^2 \ , \tag{4.7.2}$$

where D_0 is the bulk diffusion coefficient (see Eq. (3.2.12)) and ξ being the noise-level due to the statistical uncertainty on the position measurement. Using the radius measured from the Lorenz-Mie method, we find for the observed colloid $D_0 = 146$ μm^2.ms^{-1}. The

experimental MSD of the sedimenting colloid is shown in Fig. 4.7.4. Once the MSD is computed, we measure both the diffusion coefficient D_0 and the terminal sedimentation velocity v_{sed} by fitting the experimental MSD to Eq. 4.7.2. Doing so, we experimentally measure $D_0 = 144 \pm 1\ \mu\text{m}^2.\text{ms}^{-1}$ and $v_{\text{sed}} = 222 \pm 1\ \text{nm}.\text{s}^{-1}$, corresponding to errors of 1% and 3%, respectively. From the MSD we also evaluate the statistical error on the particle position measurement along the z-axis to $\xi = 25$ nm. This uncertainty can be reduced by using a larger images and more sensitive cameras. However, for the study of confined particle where the range of motion is smaller, we optimized the position of the focal plane to reduce the uncertainty on the position measurement. Finally, using Eq. 4.7.1 we can calculate the density difference $\Delta\rho = 48.4 \pm 0.2\ \text{kg}.\text{m}^{-3}$. The discrepancy from the tabulated $\Delta\rho = 50\ \text{kg}.\text{m}^{-3}$ for polystyrene in water might be attributed to rugosities and/or porosity. Nonetheless, with the small measured deviations from the theory, we correctly showed the viability of the Lorenz-Mie framework to study Brownian dynamics.

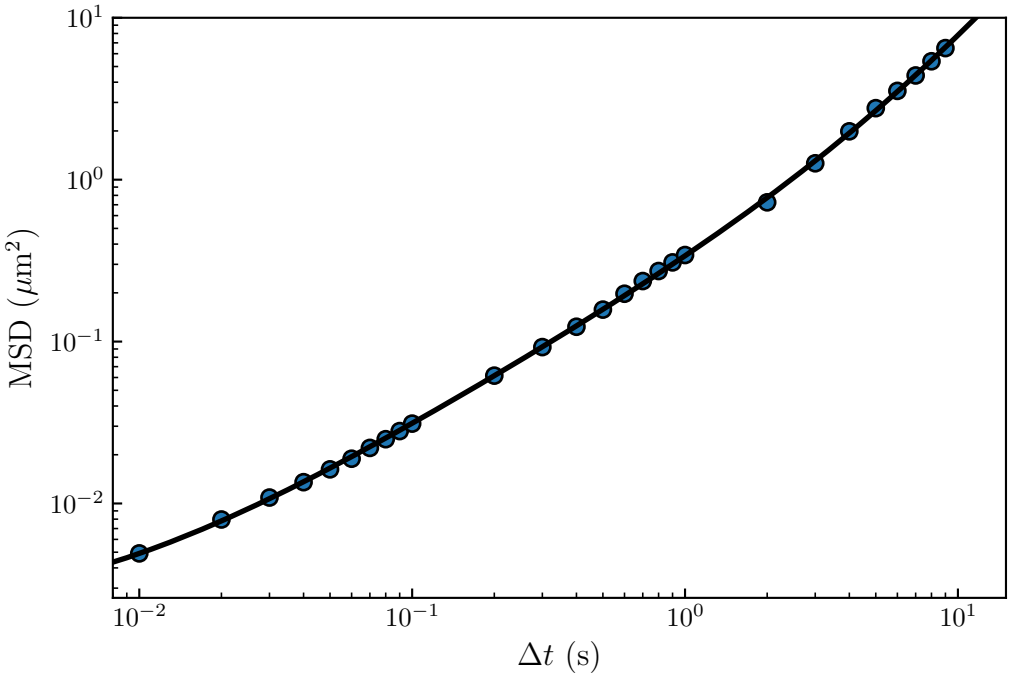


Figure 4.7.4: Measured mean-squared displacement as a function of the time increment Δt, along the z-axis. The solid line is the best fit to Eq. (4.7.2), with $D_0 = 144 \pm 1\ \mu\text{m}^2.\text{ms}^{-1}$, $v_{\text{sed}} = 222 \pm 1\ \text{nm}.\text{s}^{-1}$ and $xi = 25$ nm.

4.7.5 Conclusion

In this chapter, we have covered different techniques that enable the tracking of individual microparticles. Each method has pros and cons. We decided to employ the Lorenz-Mie

framework since it requires no calibration. Then, we have shown how we implement it in practice, from the experimental setup to the numerical treatment. An example of the Jupyter notebooks employed for the tracking can be found in appendix A.2. We have discussed how to have fast and accurate fits to retrieve the particle trajectory. To do so, we first characterize the particle fully, namely, its radius and optical index, before tracking a whole video.

Now that we have an understanding on the tracking of single colloids, we can use the measured trajectories in order to understand how the Brownian motion is affected in various configurations.

5 Stochastic Inference of Surface-Induced Effects Using Brownian Motion

5.1 Confined Brownian motion theory

By observing the experimental trajectory along the z-axis of a particle of 1.5 μm radius as shown in Fig. 5.1.1, one can notice that the particle's height does not get higher than approximately 4 μm. Indeed, due to gravity, a colloid is confined near the surface. This confinement induces near-wall effects, such as hindered mobility and electrostatic interactions.

In the first part of this chapter, I will detail the theory of confined Brownian motion and how to numerically simulate it. In a second part, I will present how to analyze experimental data. In particular, I will detail a multi-fitting procedure that enables thermal-noise-limited inference of diffusion coefficients spatially resolved at the nanoscale, equilibrium potentials, and forces at the femtonewton resolution.

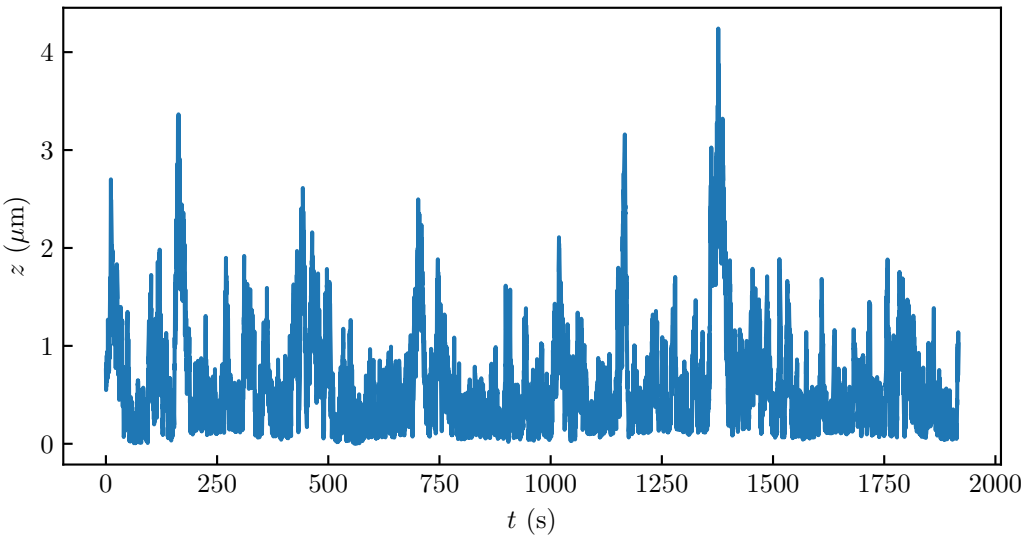


Figure 5.1.1: Experimental trajectory of a polystyrene particle of radius $a = 1.5$ μm in water near a glass wall ($z = 0$) along the z-axis — *i.e* perpendicular to the wall.

5.1.1 Gravitational potential

The density ρ_p of an observed colloid is different from the medium density ρ_m. In our experiment, we used water whose density is $\rho_m = 1000$ kg.m^{-3}. Thus, the particles are

subject to gravitational potential given by:

$$U_{\mathrm{g}}(z) = \Delta m g z = \frac{4}{3}\pi a^3 g \Delta\rho z \ , \tag{5.1.1}$$

where Δm is the difference between the mass of the particle and that of a fluid sphere of the same size, $\Delta\rho = \rho_{\mathrm{m}} - \rho_{\mathrm{p}}$ is the corresponding density difference, and g is the gravitational acceleration. By invoking the definition of the Boltzmann length:

$$\ell_{\mathrm{B}} = \frac{k_{\mathrm{B}}T}{(4/3)\pi a^3 \Delta\rho g} \ , \tag{5.1.2}$$

one can rewrite Eq. (5.1.1) as:

$$U_{\mathrm{g}}(z) = \frac{k_{\mathrm{B}}T}{\ell_{\mathrm{B}}} z \ . \tag{5.1.3}$$

The Boltzmann length ℓ_{B} corresponds to the spatial extent over which the change of gravitational energy equals thermal energy. This distance was first measured by Perrin [**perrin˙les˙2014**]. To do so, using a microscope he counted the number of colloidal particles as a function of the height in the sample. Then, he reconstructed the concentration profile of the colloidal suspension that exponentially decays as $\mathrm{e}^{-z/\ell_{\mathrm{B}}}$. As an example, for a polystyrene particle of radius $a = 1.5$ μm in water, one has $\ell_{\mathrm{B}} = 580$ nm.

For systems with $\ell_{\mathrm{B}} >> h$, where h is the vertical thickness of the sample, one can consider that the particle does not feel gravity. This is particularly the case when the densities of the colloids and the fluid are equal. In this particular case, one has $\ell_{\mathrm{B}} = \infty$. Thus, density matching can be a way to do gravity-free experiments. In our experiment, we want to measure confinement-induced effects. Therefore, we need gravity for particles to be driven towards the substrate. As particles get larger or denser, ℓ_{B} decreases and particles are, on average, closer to the substrate.

5.1.2 Sphere-wall interactions

As we have seen, external forces such as gravity act on the particles. As Brownian particles are close to a wall, we can also expect some interactions between the particles and the wall. In our case, we suppose that the Brownian particles do not interact with each other,

as we consider dilute solutions only. Indeed, the studied particles are at least 50 μm apart from each other, which corresponds to 10 times their size for the largest beads.

To describe the interaction between a Brownian particle and the wall, we use the DLVO theory, named after Derjaguin, Landau, Verwey, and Overbeek [**israelachvili˙intermolecular˙2015**]. This theory was first developed to describe the interactions between colloids, and explains the stability of colloidal suspensions. It involves two force components; the Hamaker force which arises from van der Waals interactions between the molecules of the two surfaces and a screened electrostatic force due to a double layer of charges formed near each surface, and involving the ions present in the solution.

5.1.2.1 Double layer interactions

When a surface is immersed in water, it usually acquire charges [**israelachvili˙intermolecular˙2015**] due to a high water dielectric constant $\epsilon = \epsilon_0\epsilon_\mathrm{r}$, where ϵ_0 is the vacuum permittivity and ϵ_r the medium relative permittivity; for water $\epsilon_\mathrm{r} = 80$. Commonly, surface charging is done through the ionization of surface groups[6], or from the binding of ions from the solution — for example, adsorption of $-\mathrm{OH}^-$ onto the water-air interface that charges it negatively. In the bulk, a fluid is electrically neutral; thus the fluid contains as equal number of ions of opposite charges (including the proper stoichiometry due to ionic valencies). However, when a surface is negatively charged, the negative ions are repelled from it, while positive ions are attracted towards it. Therefore, a double-layer charge distribution is formed near the surface, as shown in Fig. 5.1.2. Experimentally, we use glass slides and polystyrene beads that are both negatively charged in water leading to a repulsive interaction between them. This repulsive force prevents the colloids from sticking together, or to the substrate's surface.

The DLVO theory states that the electrostatic potential $\Psi(\vec{r})$ generated by an ion of one given species i at a distance $\vec{r}$ satisfies the Poisson equation [**israelachvili˙intermolecular˙2015**]:

$$\nabla^2\Psi(\vec{r}) = -\frac{1}{\epsilon_\mathrm{r}\epsilon_0}\rho_\mathrm{e}(\vec{r}) \ , \tag{5.1.4}$$

with:

[6] For example, the dissociation of protons from surface carboxylic groups [**israelachvili˙intermolecular˙2015**] ($-\mathrm{COOH} \rightarrow$ -COO^- + H^+) which charges negatively the surface.

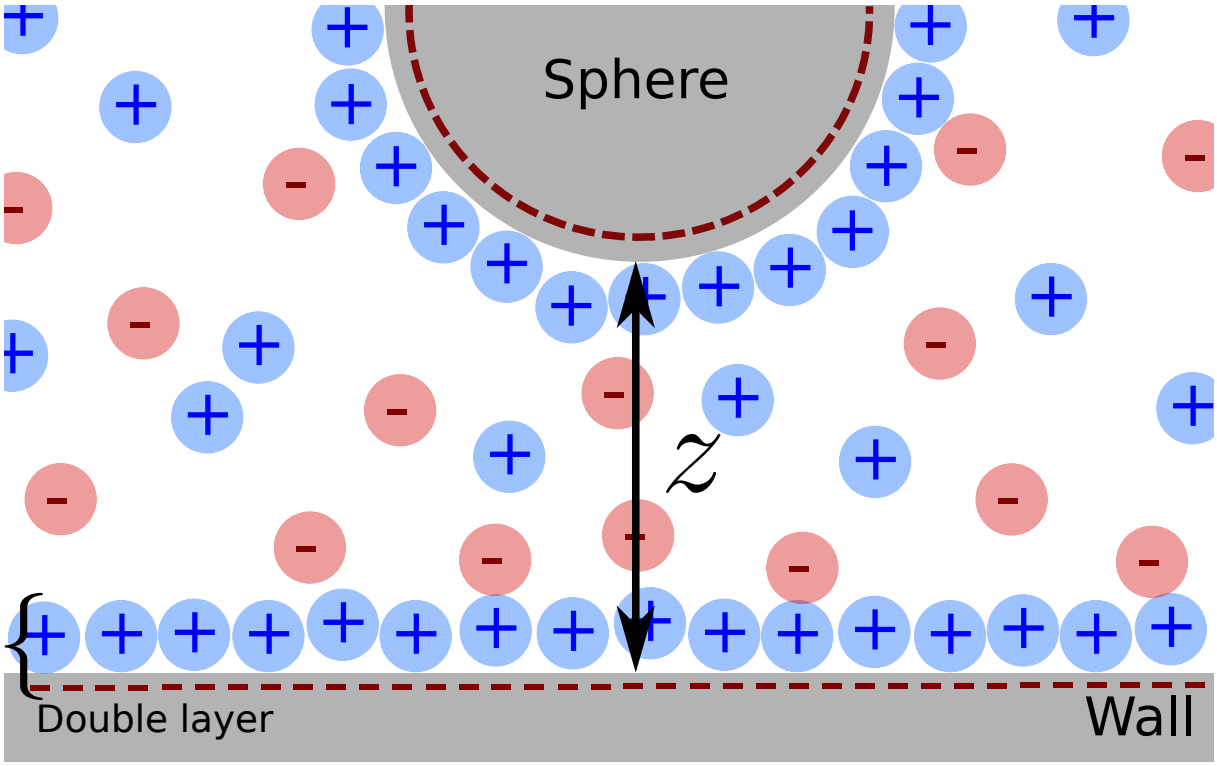


Figure 5.1.2: A colloid diffusing near a wall. Both the wall and colloid surfaces charge negatively. As a consequence, a layer of positively-charged ions is attracted towards each surface, forming a double layer.

$$\rho_{\mathrm{e}}(\vec{r}) = e \sum_i z_i c_i(\vec{r}) \ , \tag{5.1.5}$$

the local charge density, where e is the elementary charge, and where the index i denotes an ionic species of valence z_i and local ionic concentration c_i (number density). If the solution is at thermodynamic equilibrium, the local ionic density is given by a Gibbs-Boltzmann distribution, as:

$$c_i(\vec{r}) = c_i^0 \exp\left(\frac{-z_i e \Psi(\vec{r})}{k_{\mathrm{B}} T}\right) \ , \tag{5.1.6}$$

where c_i^0 is the bulk concentration (number density) of the ionic species i. By combining Eqs. (5.1.4), (5.1.5) and (5.1.6), one obtains the Poisson-Boltzmann equation:

$$\nabla^2 \Psi(\vec{r}) + \sum_i \frac{z_i e c_i^0}{\epsilon_0 \epsilon_{\mathrm{r}}} \exp\left(-\frac{z_i e \Psi(\vec{r})}{k_{\mathrm{B}} T}\right) = 0 \ . \tag{5.1.7}$$

Since Eq. (5.1.7) is nonlinear, it is typically solved numerically. However, for some simple configurations such as uniformly-charged plane or sphere it can be solved analytically. To simplify, let us consider that we have a monovalent electrolyte, meaning that the electrolyte is composed of two ions of valencies both equal to 1 — Na^+ Cl^- for example — and c_i^0 is equal to the bulk electrolytic concentration c_s^0. In such a case, Eq. (5.1.7) simplifies and becomes:

$$\begin{aligned} &\nabla^2\Psi(\vec{r}) + \frac{ec_s^0}{\epsilon_0\epsilon_\mathrm{r}}\left[\exp\left(\frac{-e\Psi(\vec{r})}{k_\mathrm{B}T}\right) - \exp\left(\frac{+e\Psi(\vec{r})}{k_\mathrm{B}T}\right)\right] = 0 \\ &\nabla^2\Psi(\vec{r}) + 2\frac{ec_s^0}{\epsilon_0\epsilon_\mathrm{r}}\sinh\left(\frac{e\Psi(\vec{r})}{k_\mathrm{B}T}\right) = 0 \ . \end{aligned} \tag{5.1.8}$$

Another situation leading to analytical results, is when Ψ is small enough such that $e\Psi \ll k_\mathrm{B}T$, which is generally the case when using dilute enough solutions, it is possible, through a Taylor expansion at first order to write:

$$\exp\left(-\frac{z_i e\Psi(\vec{r})}{k_\mathrm{B}T}\right) \simeq 1 - \frac{z_i e\Psi(\vec{r})}{k_\mathrm{B}T} \ . \tag{5.1.9}$$

In such case, Eq. (5.1.7) becomes:

$$\nabla^2\Psi(\vec{r}) + \sum_i \frac{z_i e c_i^0}{\epsilon_0\epsilon_\mathrm{r}}\left(1 - \frac{z_i e\Psi(\vec{r})}{k_\mathrm{B}T}\right) = 0 \ . \tag{5.1.10}$$

In addition, a fluid is electrically neutral. Therefore, $\sum_i z_i c_i^0 = 0$. Thus, one can simplify Eq. (5.1.10) to get the Debye-Hückel equation:

$$\nabla^2\Psi(\vec{r}) = \left[\sum_i \frac{z_i^2 e^2 c_i^0}{\epsilon_0\epsilon_\mathrm{r}k_\mathrm{B}T}\right]\Psi(\vec{r}) \ . \tag{5.1.11}$$

One can identify the term between brackets as the inverse of a length squared. We thus define the Debye length as:

$$\ell_\mathrm{D} = \sqrt{\sum_i \frac{\epsilon_0\epsilon_\mathrm{r}k_\mathrm{B}T}{z_i^2 e^2 c_i^0}} \ , \tag{5.1.12}$$

which is the characteristic ion-induced screening length of the electrostatic interactions, as we will see below. For a monovalent electrolyte, at 25 °C, the Debye length of an aqueous solution is:

$$\ell_{\mathrm{D}} = \sqrt{\frac{2\epsilon_0\epsilon_{\mathrm{r}}k_{\mathrm{B}}T}{c_s^0 e^2}} \quad = \frac{0.304}{\sqrt{C}} \text{ nm} , \tag{5.1.13}$$

with C the value of the molar concentration in mol.L^{-1}:

$$C = \frac{c_{\mathrm{s}}^0}{N_{\mathrm{A}}} 10^{-3} . \tag{5.1.14}$$

For example, for NaCl salt in water, $\ell_{\mathrm{D}} \approx 100$ nm for a concentration $C = [\mathrm{NaCl}] = 9.2$ μmol.L^{-1} and $\ell_{\mathrm{D}} \approx 10$ nm for a concentration $[\mathrm{NaCl}] = 9.2$ mmol.L^{-1}.

Finally, combining Eqs. (5.1.11) and (5.1.12), the Debye-Hückel equation reads:

$$\nabla^2\Psi(\vec{r}) = \kappa^2\Psi(\vec{r}) , \tag{5.1.15}$$

with $\kappa = 1/\ell_{\mathrm{D}}$. Using the latter, one can compute the electrostatic potential around a sphere immersed in an ionic solution. Let us consider a sphere of radius a and charge Qe, *i.e.* a charge density $\sigma = Qe/(4\pi a^2)$, Q being the number of charges on the surface. Since the system has a spherical symmetry, one has $\Psi(\vec{r}) = \Psi(r)$ with $r = |\vec{r}|$. Using the Laplacian operator ∇^2 in spherical coordinates, Eq. (5.1.15) becomes:

$$\frac{1}{r^2}\left[\frac{\partial}{\partial r}\left(r^2\frac{\partial\Psi(r)}{\partial r}\right)\right] = \kappa^2\Psi(r) , \tag{5.1.16}$$

which has a general solution:

$$\Psi(r) = C_1\frac{\exp(\kappa r)}{r} + C_2\frac{\exp(-\kappa r)}{r} . \tag{5.1.17}$$

The electrostatic potential vanishes at infinity such that $C_1 = 0$. Therefore, the electrostatic potential (and thus the electrostatic energy potential) takes the form of a Yukawa potential:

$$\Psi(r) = C_2 \frac{\exp(-\kappa r)}{r} \ . \tag{5.1.18}$$

Additionally, invoking the Gauss theorem, at the surface of the charged sphere, the electrostatic potential satisfies:

$$\left.\frac{\partial \Psi(r)}{\partial r}\right|_{r=a} = \frac{-Qe}{4\pi\epsilon_0\epsilon_\mathrm{r} a^2} = \frac{-\sigma}{\epsilon_0\epsilon_\mathrm{r}} \ , \tag{5.1.19}$$

where we introduced the surface density of charges σ. By applying the latter boundary condition to Eq. (5.1.18), we find:

$$\Psi(r) = \frac{\sigma a^2}{\epsilon_0\epsilon_\mathrm{r}} \frac{\exp(\kappa a)}{1+\kappa a} \frac{\exp(-\kappa r)}{r} \ . \tag{5.1.20}$$

This solution can be used to determine the electrostatic potential between two spheres of radii a_1 and a_2, and surface charge densities σ_1 and σ_2, respectively. Supposing that the presence of a second sphere does not modify the distribution of ions in the double layer of the other sphere, one can use the superposition approximation to obtain the potential $U^{ss}_\mathrm{elec}(z)$ between the two spheres [**bell˙approximate˙1970**]:

$$U^{ss}_\mathrm{elec}(z) = \frac{4\pi}{\epsilon_0\epsilon_\mathrm{r}} \left(\frac{\sigma_1 a_1^2}{1+\kappa a_1}\right) \left(\frac{\sigma_2 a_2^2}{1+\kappa a_2}\right) \frac{\exp(-\kappa z)}{a_1 + a_2 + z} \ , \tag{5.1.21}$$

with z the gap between the two colloids. From the latter equation, it is possible to write the electrostatic interaction energy U_elec between a planar wall of charge density σ_w and a spherical colloid of radius a and surface charge density σ, by setting one of the two radii to infinity. Doing so, one gets:

$$\frac{U_\mathrm{elec}(z)}{k_\mathrm{B}T} = B\mathrm{e}^{-\frac{z}{\ell_\mathrm{D}}} \ , \tag{5.1.22}$$

where:

$$B = \frac{4\pi}{k_\mathrm{B}T\epsilon_0\epsilon_\mathrm{r}} \left(\frac{\sigma a^2}{1+\kappa a}\right) \frac{\sigma_\mathrm{w}}{\kappa} \ . \tag{5.1.23}$$

Let us note that, B is often written as [**behrens˙charge˙2001**]:

$$B = 16\epsilon_r\epsilon_0 a\frac{k_B T}{e^2}\tanh\left(\frac{e\phi}{4k_B T}\right)\tanh\left(\frac{e\phi_w}{4k_B T}\right) , \tag{5.1.24}$$

where ϕ and ϕ_w are the Stern potentials of the sphere and wall surface, respectively. Typical values for B range from 1 to 50. In our study, we will use B to characterize the dimensionless magnitude of the electrostatic interaction. Indeed, it is complicated to decouple σ and σ_w when the colloid and wall materials are different [**behrens˙charge˙2001**].

5.1.2.2 van der Waals interactions

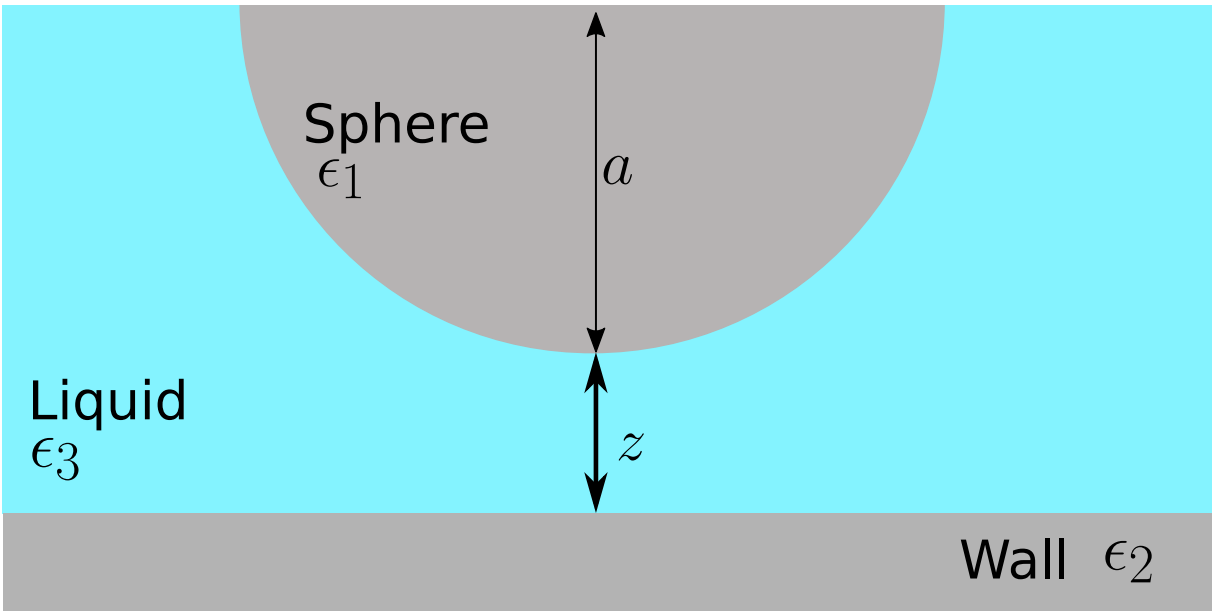


Figure 5.1.3: A colloid of radius a is located at a distance z from the wall. The dielectric constants of the sphere, wall and liquid are respectively ϵ_1, ϵ_2, and ϵ_3.

In the DLVO theory, van der Waals interactions, after integration over all surfaces contribute through a global Hamaker potential energy. This potential is short-ranged and attractive, in our case.The interaction potential reads: [**israelachvili˙intermolecular˙2015**]:

$$U_{vdW} = -\frac{Aa}{6z} \tag{5.1.25}$$

where A is the nonretarded Hamaker constant. For our system, where the particle, medium and wall are different media as schematize in Fig. 5.1.3, the Hamaker constant is given by [**israelachvili˙intermolecular˙2015**]:

$$A = \frac{3}{4} k_{\mathrm{B}} T \left(\frac{\epsilon_1 - \epsilon_3}{\epsilon_1 + \epsilon_3} \right) \left(\frac{\epsilon_2 - \epsilon_3}{\epsilon_2 + \epsilon_3} \right) + \frac{3h}{4\pi} \int_{\nu_1}^{\infty} \left(\frac{\epsilon_1(j\nu) - \epsilon_3(j\nu)}{\epsilon_1(j\nu) + \epsilon_3(j\nu)} \right) \left(\frac{\epsilon_2(j\nu) - \epsilon_3(j\nu)}{\epsilon_2(j\nu) + \epsilon_3(j\nu)} \right) \mathrm{d}\nu \ , \tag{5.1.26}$$

where ϵ_1, ϵ_2 and ϵ_3 are the static dielectric constants of the three media, $\epsilon_{1,2,3}(j\nu)$ are the dielectric constant at a imaginary frequency $j\nu$. The first term gives the zero-frequency energy of the Van der Waals interaction and the second term the dispersion energy. In the literature, we found for polystyrene colloids in water near a glass substrate $A \approx k_{\mathrm{B}}T$. Since A is positive, the interaction is attractive, moreover, we estimate that the van der Waals forces play a role only within a few nanometers from the surface ($z < 10$ nm), as commonly observed [**prieve˙measurement˙1999**]. In our experiments, the Debye length ℓ_{D} (> 20 nm) is large enough for the particles to avoid this region. Therefore, in the following, the van der Waals interactions are neglected. It is possible to study the van der Waals interactions with Brownian motion, provided that one adds enough salt to have $\ell_{\mathrm{D}} \simeq 1$ nm. However, with such a short Debye length, all the colloids would stick to the surface and with each other, as a result of van der Waals forces. Interestingly, we have experimentally observed stuck particles. Further work on these events may lead to interesting insights about the near-wall interactions. Nonetheless, in the case where A needs to be computed a special attention needs to be taken on the evaluation of the tems at high frequency dielectric constant [**parsegian˙van˙2005**]

5.1.2.3 Total potential and equilibrium distribution

If we combine the gravitational and electrostatic energy potentials the particles lie into a total energy potential $U(z)$, given by:

$$U(z) = U_{\mathrm{g}} + U_{\mathrm{elec}} \ . \tag{5.1.27}$$

By combining Eqs. (5.1.3), (5.1.22) and (5.1.27), and adding the condition that a particle cannot go inside the wall, one finally gets:

$$\frac{U(z)}{k_{\mathrm{B}}T} = \begin{cases} B\, \mathrm{e}^{-\frac{z}{\ell_{\mathrm{D}}}} + \dfrac{z}{\ell_{\mathrm{B}}} \ , & \text{for } z > 0 \\ +\infty \ , & \text{for } z < 0 \end{cases} \ . \tag{5.1.28}$$

From this total potential energy, one can then construct the Gibbs-Boltzmann distribution to write the equilibrium PDF of position $P_{\mathrm{eq}}(z)$:

$$P_{\mathrm{eq}}(z) = A \exp\left(-\frac{U(z)}{k_{\mathrm{B}}T}\right) , \tag{5.1.29}$$

where A is a normalization constant such that $\int_0^\infty P_{\mathrm{eq}}(z)\mathrm{d}z = 1$. Given an ensemble of heights z_i, one can compute P_{eq} using the following Python function `Peq`, where the A is computed using the `np.trapz` function. Examples of a theoretical energy potential and associated PDF of position can be seen in Fig. 5.1.4 for $\ell_{\mathrm{B}} = 500$ nm, $B = 4$ and $\ell_{\mathrm{D}} = 50$ nm.

```
import numpy as np

def _Peq(z):
  if z <= 0:
    return 0
  else:
    return np.exp(-(B * np.exp(-z / ld) + z / lb))


def Peq(z):
  P = np.array([_Peq(zi) for zi in z])
return P / np.trapz(P,z)
```

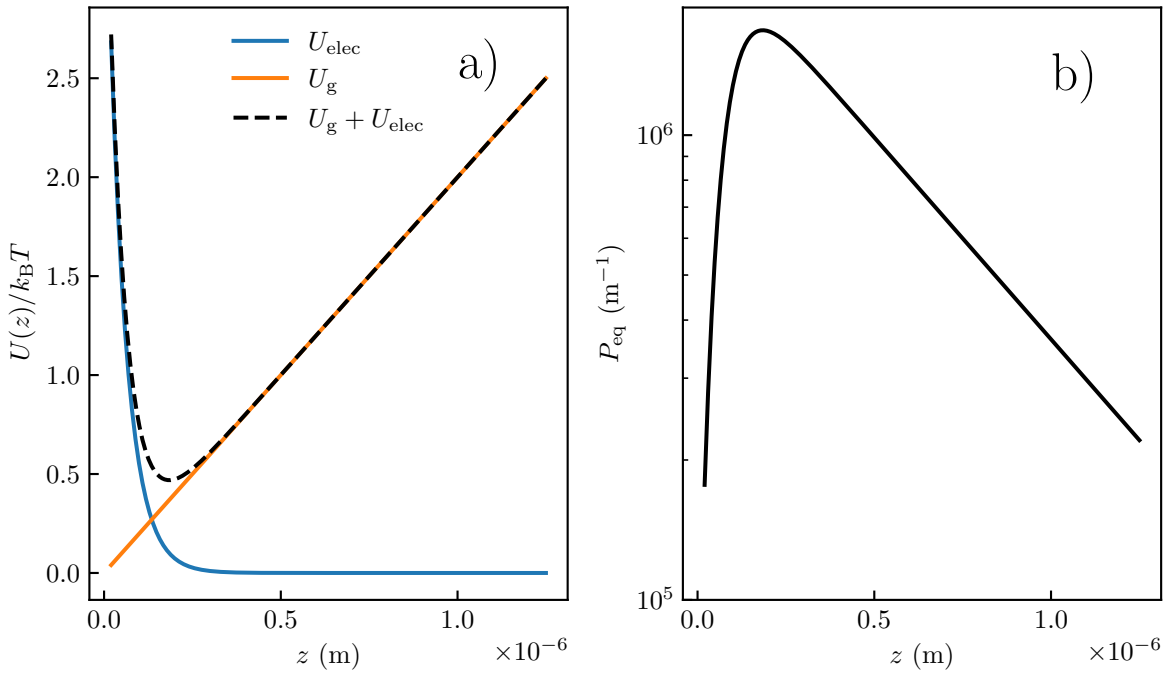


Figure 5.1.4: a) In orange potential energy U_{g} (see Eq. (5.1.3)) of a colloid with a Boltzmann length $\ell_{\text{B}} = 500$ nm. In blue, the electrostatic potential energy U_{elec} (see Eq. (5.1.22)) is characterized by a dimensionless magnitude $B = 4$ and a Debye length $\ell_{\text{D}} = 50$ nm. The dashed line corresponds to the total potential U, see Eq. (5.1.27). b) Corresponding Gibbs-Boltzmann equilibrium distribution of position calculated using the energy potential of panel a).

5.1.3 Local diffusion coefficient

As we have seen in Chapter 3, for a freely diffusing colloid in the bulk, the diffusion coefficient is given by Eq. (3.2.12) and is a constant. However, when a particle is confined by a rigid wall, the diffusion is hindered. This means that the diffusion coefficient varies with the particle-wall distance and becomes anisotropic. A seminal measurement of this effect was done by Faucheux and Libchaber [**faucheux˙confined˙1994**]. As we can see in Fig. 5.1.5, using a microscope, they tracked colloids within a parallelepipedic chamber, and measured the thickness-averaged diffusion coefficients for different values of the confinement parameter $\gamma = \langle z \rangle_t / a$, with $\langle z \rangle_t$ time-averaged particle-wall distance. As experiments reach equilibrium, $\langle z \rangle_t$ is given by the Gibbs-Boltzmann distribution as $\langle z \rangle_t = \int \mathrm{d}z P_{\text{eq}}(z)) z$. We observe that the diffusion coefficient parallel to the surface decreases as the particle gets closer to the wall, and seems to saturate around $0.3 D_0$ at low γ, with D_0 the diffusion coefficient in the bulk (see Eq. 5.1.33).

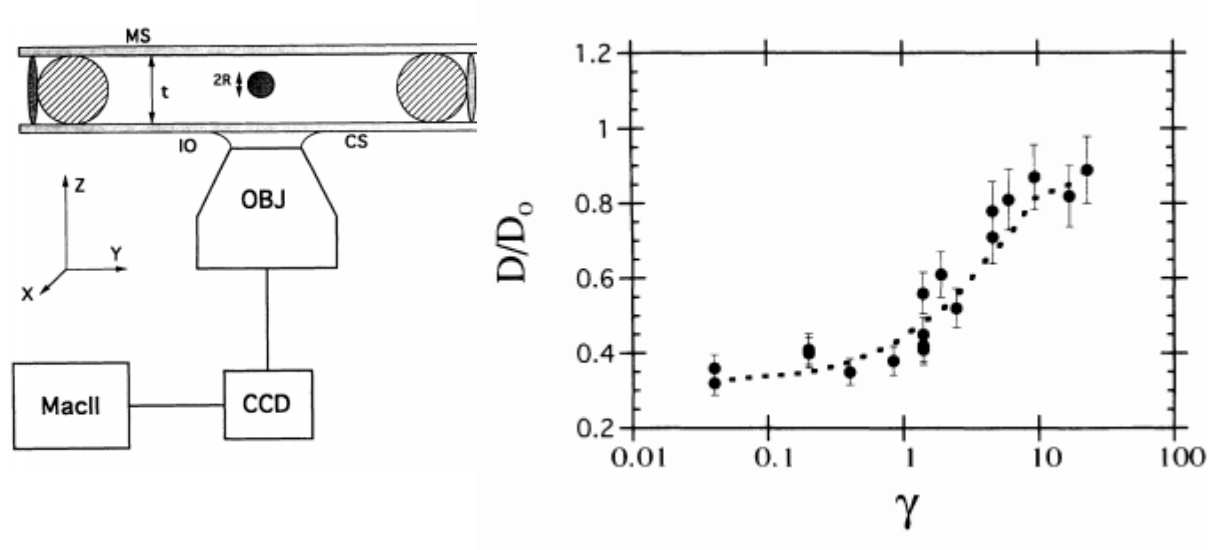


Figure 5.1.5: Figure extracted from [**faucheux˙confined˙1994**]. On the left is the experimental setup. It is an inverted microscope used in order to track micrometric particles of diameter $2R$ inside a liquid cell of thickness t. On the right is the final result, where the authors measure the diffusion parallel (*i.e.* along x or y) coefficient $D_{\parallel}$ (see Eqs. (5.1.47) and (5.1.45)), normalized by the bulk diffusion coefficient D_0, as a function the confinement parameter $\gamma = \langle z \rangle_t / a$, with $\langle z \rangle_t$ time-averaged particle-wall distance.

To understand the reason for this hindered diffusion coefficient, let us start by writing the diffusion coefficient D using the fluctuation dissipation theorem:

$$D = \frac{1}{\gamma} k_{\mathrm{B}} T \ , \tag{5.1.30}$$

with the mobility defined as:

$$\frac{1}{\gamma} = \left| \frac{v_{\mathrm{sphere}}}{F_{\mathrm{drag}}} \right| \ , \tag{5.1.31}$$

where v_{sphere} is the terminal velocity to an applied force F_{drag}. For a spherical colloid of radius a moving at a velocity v_{sphere}, the drag force $F^{\mathrm{B}}_{\mathrm{drag}}$ is given by the Stokes' law:

$$F^{\mathrm{B}}_{\mathrm{drag}} = -c\pi\eta a v_{\mathrm{sphere}} \ , \tag{5.1.32}$$

where c is a constant that depends on the boundary conditions imposed at the surface

of the colloid. Typically, one has $c = 6$ for a no-slip boundary conditions and $c = 4$ for a full-slip boundary conditions, such as for air bubbles, for example. Combining Eqs. (5.1.30), (5.1.31) and (5.1.32) for a freely diffusing no-slip hard sphere in the bulk we retrieve Eq. (3.2.12):

$$D = D_0 = \frac{k_{\mathrm{B}}T}{6\pi\eta a} \ . \tag{5.1.33}$$

The Stokes' drag force can be computed by solving the Navier-Stokes equation:

$$\rho\left[\frac{\partial\vec{v}}{\partial t} + (\vec{v}\cdot\nabla\vec{v})\right] + \nabla p = \eta\nabla^2\vec{v} \ , \tag{5.1.34}$$

and the continuity equation for incompressible fluids:

$$\nabla\cdot\vec{v} = 0 \ , \tag{5.1.35}$$

where $\vec{v}$ and p are respectively the velocity and pressure fields, and where ρ is the liquid density. When the Reynolds number $\mathrm{Re} = \rho a v_{\mathrm{sphere}}/\eta \ll 1$, the second inertial term is negligibly small compared to the viscous term $\eta\nabla^2\vec{v}$. In that case, and at long-enough time for the first inertial term to be negligible, the Eq. (5.1.34) is simplified to the steady Stokes equation:

$$\nabla p = \eta\nabla^2\vec{v} \ . \tag{5.1.36}$$

By solving Eqs. (5.1.35) and (5.1.36) with a no-slip boundary condition on the particle surface and the field vanishing at infinity, one can calculate the velocity and the pressure fields in the fluid. By integration of the pressure and viscous stress on the particle surface, one eventually gets the Stokes mobility. However, in the case of a confined particle near a wall there is an additional no-slip condition at the wall surface. At the macro scale, this effect can be seen with a frisbee, indeed, as it gets closer to the ground, hydrodynamic pressure increases due to the increasing air velocity gradient in the gap and one can observe a slowing down of the free fall.

To get some physical insight on this effect, one can use the lubrication theory to make a scaling of the drag force experience by a particle confined near a wall. As schematized in

Fig. 5.1.6, we consider a particle of radius a moving at a velocity V.

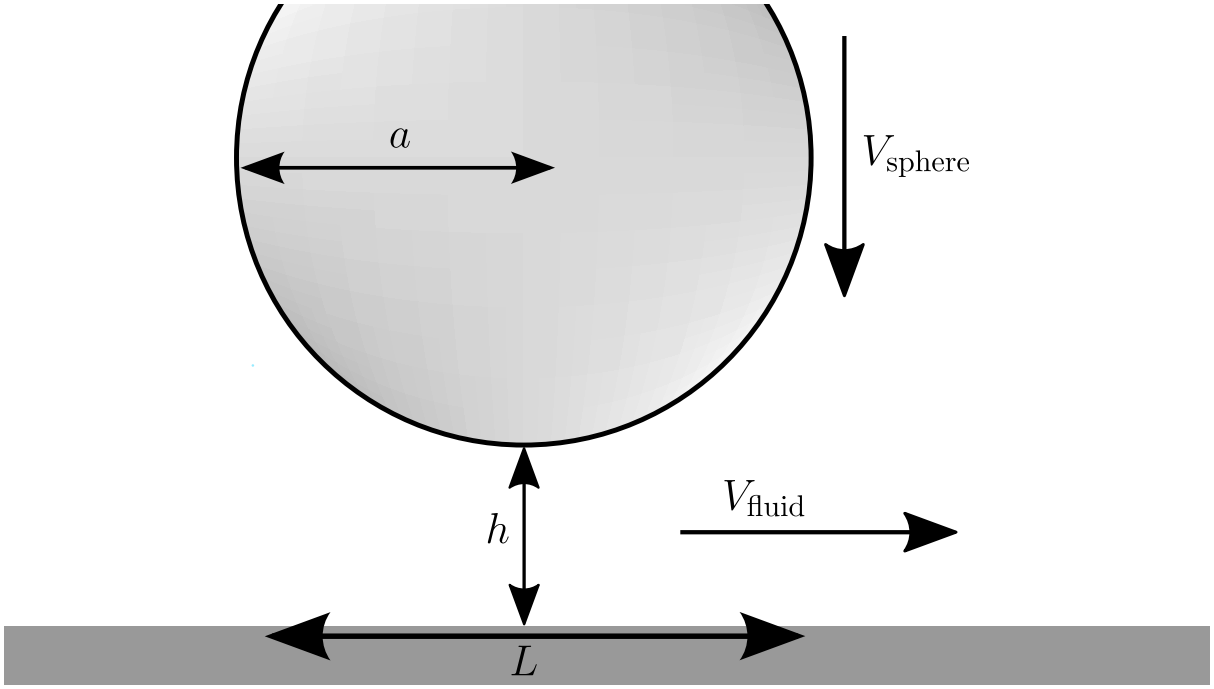


Figure 5.1.6: Schematic representation of a spherical object of radius a moving towards a wall at velocity V_{sphere} and inducing a fluid velocity V_{fluid}

As we are using the lubrication theory, we suppose that the particle is close to the wall such that $h \ll a$. In that condition, we suppose that a particle moving towards a wall (*along the z-axis*) at a velocity V_{sphere} induce a fluid velocity V_{fluid} along the x-axis, we further suppose that the induced velocity along the z-axis is negligible. Moreover, the typical length scale along the z-axis is the particle-wall distance z; and the distance $L = \sqrt{az}$ (*i.e.* Hertz contact), along the x-axis. In this approximation, velocity terms along the z-axis will be negligible in Eq. (5.1.36), a projection along the z-axis thus gives:

$$\frac{\partial p}{\partial z} = 0 \ . \tag{5.1.37}$$

On the right-hand side of Eq. 5.1.36, the viscous term simplifies to $\eta \partial_z^2 V_{\text{fluid}}$ as:

$$\frac{\partial^2 V_{\text{fluid}}}{\partial x^2} \approx \frac{V_{\text{fluid}}}{L^2} \ll \frac{V_{\text{fluid}}}{h^2} \approx \frac{\partial^2 V_{\text{fluid}}}{\partial z^2} \ . \tag{5.1.38}$$

In the lubrication theory, Eq. (5.1.36) is finally simplified to:

$$\frac{\partial p}{\partial x} = \eta \frac{\partial^2 V_{\text{fluid}}}{\partial z^2} \ , \tag{5.1.39}$$

which scales as:

$$p \sim \eta V_{\text{fluid}} \frac{L}{h^2} \ . \tag{5.1.40}$$

To compute the Stokes mobility, one needs to evaluate the viscous stress σ, in the lubrication theory, it reads:

$$\sigma = \eta \frac{\partial V_{\text{fluid}}}{\partial z} \sim \eta \frac{V_{\text{fluid}}}{h} \ . \tag{5.1.41}$$

When the particle is moving towards the wall we thus have $p \gg \sigma$, we thus only consider the pressure. To evaluate the mobility, we need to write Eq. (5.1.40) as a function of V_{sphere}. Using Eq. (5.1.35) one has $V_{\text{fluid}}/L \sim V_{\text{sphere}}/h$, Eq. (5.1.40) thus become:

$$p \sim \eta V_{\text{sphere}} \frac{L^4}{h^3} \tag{5.1.42}$$

Integrating this pressure over the particle surface (*i.e* over the typical surface $L^2 = ah$) leads a scaling of the drag force:

$$F_{\text{drag}}^{z \ll a} \sim \eta V_{\text{sphere}} \frac{L^4}{h^3} = \eta V_{\text{sphere}} \frac{a^2}{z} \tag{5.1.43}$$

The mobility (see Eq. (5.1.31)) of the displacement perpendicular to the wall thus scale as $\gamma_0^{-1} h/a$, with γ_0^{-1} the bulk mobility. Therefore, the diffusion coefficient of a confined colloid near a wall in the lubrication approximation is hindered and inversely proportional to the particle-wall distance. It is possible to do the same scaling for the parallel mobility by supposing that the particle is moving along the x-axis. In that case, one can find that the viscous stress is greater than the pressure and finally find that along the x-axis, the parallel mobility follows the same scaling as in bulk and remain constant.

To recap, a colloid diffusing near a wall experience a local drag force that depends on both its distance z to the wall and direction of motion. Thanks to the linearity of the Stokes equation, one can decompose this local drag force in two contributions, for motions parallel and perpendicular to the wall. As the presence of the wall modifies the drag force with a space-dependent multiplicative factor, the confinement effect is often expressed in terms of effective viscosities:

$$\eta_\perp(z) = \eta\lambda_\perp(z) \text{ , and } \eta_\parallel(z) = \eta\lambda_\parallel(z) \text{ ,} \tag{5.1.44}$$

where $\lambda_\perp$ and $\lambda_\parallel$ are respectively the perpendicular and parallel correction factors. Taking into account these corrections, the diffusion coefficients for perpendicular and parallel motions relative to the wall read:

$$D_\perp(z) = \frac{D_0}{\lambda_\perp(z)} \text{ , and } D_\parallel(z) = \frac{D_0}{\lambda_\parallel(z)} \text{ .} \tag{5.1.45}$$

For no-slip boundary conditions imposed at both the wall and the surface of the colloid, Brenner [**brenner˙slow˙1961**] has obtained for the perpendicular motion:

$$\lambda_\perp(z) = \frac{4}{3}\sinh\beta \sum_{n=1}^{\infty} \frac{n(n+1)}{(2n-1)(2n+3)} \left[\frac{2\sinh(2n+1)\beta + (2n+1)\sinh 2\beta}{4\sinh^2(n+1/2)\beta - (2n+1)^2\sinh^2\beta} - 1 \right] \text{ ,} \tag{5.1.46}$$

where $\beta = \cosh^{-1}((z+a)/a)$. For the motion of a sphere parallel to a wall, Faxén found [**faxen˙fredholm˙1924**]:

$$\lambda_\parallel(z) = \left[1 - \frac{9}{16}\xi + \frac{1}{8}\xi^3 - \frac{45}{256}\xi^4 - \frac{1}{16}\xi^5 \right]^{-1} \text{ ,} \tag{5.1.47}$$

where $\xi = a/(z+a)$. Eqs. (5.1.46) and (5.1.47) are exact for all z and shown in Fig. 5.1.7-a). However, the solution for the perpendicular motion can be complex to compute as it is an infinite series. It requires a software that enables arbitrary-precision floating-point arithmetic[7] — such as Mathematica or the `mpmath` Python's module, for example. $D_\perp$ can be evaluated using the following Python snippet, where the `nsum` function is used to compute the summation:

[7] Arbitrary-precision floating-point arithmetic enables to evaluate mathematical expressions with any precision, *i.e.* any number of digits.

```
from mpmath import nsum

def Dz(eta, z, a):
  a = (z + a) / a
  beta = float(acosh(a))
  summ = nsum(
    lambda n: (n * (n + 1) / ((2 * n - 1) * (2 * n + 3)))
    * (
        (
          (2 * sinh((2 * n + 1) * xi) + (2 * n + 1) * sinh(2 * beta))
          / (
              4 * (sinh((n + 1 / 2) * beta) ** 2)
              - ((2 * n + 1) ** 2) * (sinh(beta) ** 2)
          )
    )
    - 1
  ),
  [0, inf],
  )
  summ = float(summ)
  return kT / (6 * pi * eta * 4 / 3 * float(sinh(beta)) * summ * a)
```

To simplify the computation of $\lambda_\perp$, Honig [**honig˙effect˙1971**], and Bevan and Prieve [**bevan˙hindered˙2000**] showed that Eq. (5.1.46) can be Padé approximated[8], giving:

$$\lambda_\perp = \frac{6z^2 + 9az + 2a^2}{6z^2 + 2az} \, . \tag{5.1.48}$$

In the near-wall regime, such that $z \ll a$, it is possible to further approximate $\lambda_\perp$ by its asymptotic expression:

$$\lambda_\perp \simeq \frac{a}{z} \, . \tag{5.1.49}$$

[8] A Padé aproximant is the approximation of a power series by a rational fraction [**baker˙pade˙1996**].

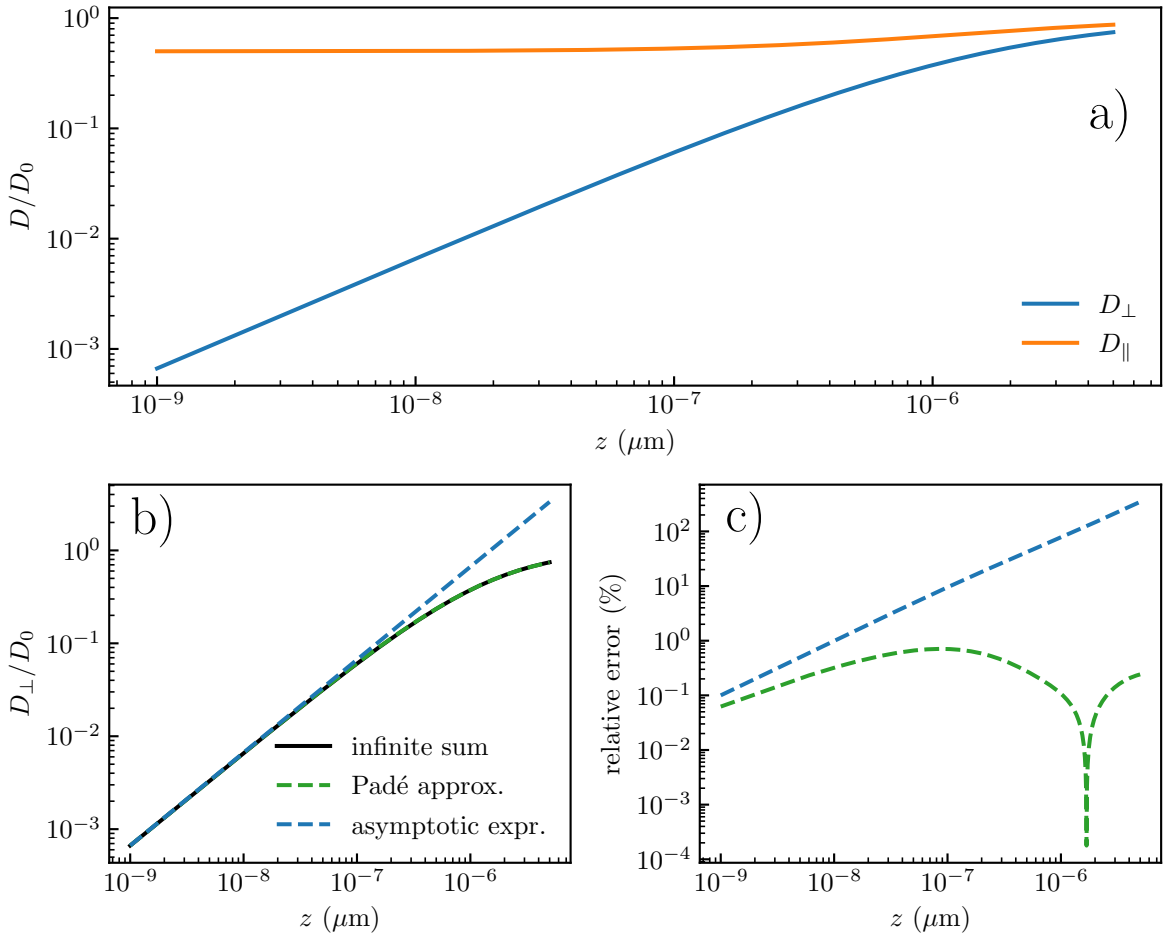


Figure 5.1.7: a) Parallel and perpendicular normalized diffusion coefficients for a colloidal particle of radius $a = 1.5$ μm. b) Perpendicular normalized diffusion coefficient at a distance z from a wall. The solid black line is the exact solution given by the infinite sum of Eq. (5.1.46). The green dashed line is the Padé approximation of Eq. (5.1.48). The blue dahed line is the near-wall asymptotic expression of Eq. (5.1.49). c) Relative errors between the two approximations (dashed lines of panel b), same color code) and the exact result (solid line of panel b)).

The exact result, together with the Padé approximation and the near-wall asymptotic expression for the hindered vertical diffusion are plotted in Fig. 5.1.7-b). The Padé approximation fits well the exact solution, the near-wall asymptotic expression fits well when $z < a/10$ typically. To check how precise both approximations are, we plot the relative error in Fig. 5.1.7-c). The Padé approximation shows precision up to 1%. Thus, in the following, when evaluating perpendicular diffusion coefficients, or equivalently vertical effective viscosities, the Padé approximation will be used.

5.1.4 Fokker-Plank equation

The Fokker-Plank equation is an alternative way to describe Brownian motion. Instead of explicitly calculating a Brownian trajectory by solving the Langevin equation, Fokker-Plank equation describes the Probability Density Function $P(X_t, X_0; t)$ in position. Where for simplicity we place ourselve in 1D, with X_t denoting the particle position at a time t and X_0 its initial ($t = 0$) position. To derive the Fokker-Plank equation, let us start by taking a generic Langevin equation in 1D:

$$\mathrm{d}X_t = u(X_t)\mathrm{d}t + g\mathrm{d}B_t \ , \tag{5.1.50}$$

where $u(X_t)$ is the drift velocity due to external forces and g is the magnitude of the random force. Let consider the average value of an arbitrary function $f(X_t)$ for a stochastic process obeying Eq. (5.1.50), which started at position x_0 at time $t = 0$. By definition, this ensemble average reads [**le˙bellac˙equilibrium˙2004**]:

$$\langle f(X_t)\rangle = \int \mathrm{d}X_t \ P(X_t, X_0; t) f(X_t) \ , \tag{5.1.51}$$

with the initial condition that can be written as:

$$P(X_t, X_0; 0) = \delta(X_t - X_0) \ . \tag{5.1.52}$$

We now expand f at the first order in the time increment $\mathrm{d}t$ as:

$$\begin{aligned}\left\langle \frac{\mathrm{d}f(X_t)}{\mathrm{d}t} \right\rangle_t &\simeq \left\langle \frac{1}{\mathrm{d}t}\left(\frac{\partial f(X_t)}{\partial X_t} u(X_t)\mathrm{d}t + \frac{\partial f(X_t)}{\partial X_t} g\mathrm{d}B_t + \frac{1}{2}\frac{\partial^2 f(X_t)}{\partial X_t^2} g^2 \mathrm{d}B_t^2 \right) \right\rangle_t \\ &= \frac{1}{\mathrm{d}t}\left(\frac{\partial f(X_t)}{\partial X_t} u(X_t)\mathrm{d}t + \frac{1}{2}\frac{\partial^2 f(X_t)}{\partial X_t^2} g^2 \mathrm{d}t \right) \\ &= \frac{\partial f(X_t)}{\partial X_t} u(X_t) + \frac{1}{2} g^2 \frac{\partial^2 f(X_t)}{\partial X_t^2} \ .\end{aligned} \tag{5.1.53}$$

By combining Eqs. (5.1.51) and (5.1.53), we get:

$$\begin{aligned}\int \mathrm{d}X_t \ \frac{\partial P(X_t, X_0; t)}{\partial t} f(X_t) &= \frac{\partial f(X_t)}{\partial X_t} u(X_t) + \frac{1}{2} \frac{\partial^2 f(X_t)}{\partial X_t^2} g^2 \\ &= \int \mathrm{d}X_t \ P(X_t, X_0; t) G f(X_t) \ ,\end{aligned} \tag{5.1.54}$$

where G is a differential operator called the generator and is defined by its action on a function f as:

$$Gf = \frac{1}{2} g^2 \frac{\partial^2 f(X_t)}{\partial X_t^2} + u(X_t) \frac{\partial f(X_t)}{\partial X_t} \ . \tag{5.1.55}$$

Using the definition of the adjoint of G, denoted $G^\dagger$, one has:

$$\begin{aligned}\int \mathrm{d}X_t \ \frac{\partial P(X_t, X_0; t)}{\partial t} f(X_t) &= \int \mathrm{d}X_t \ P(X_t, X_0; t) G f(X_t) \\ &= \int \mathrm{d}X_t \ G^\dagger P(X_t, X_0; t) f(X_t) \ .\end{aligned} \tag{5.1.56}$$

From the latter, we thus have:

$$\frac{\partial P(X_t, X_0; t)}{\partial t} = G^\dagger P(X_t, X_0; t) \ , \tag{5.1.57}$$

which leads to the Forward Fokker-Planck equation [**del˙moral˙introduction˙2010**]:

$$\frac{\partial P(X_t, X_0; t)}{\partial t} = \frac{\partial^2}{\partial X_t^2} \left[\frac{g^2}{2} P(X_t, X_0; t) \right] - \frac{\partial}{\partial X_t} \left[u(X_t) P(X_t, X_0; t) \right] \ . \tag{5.1.58}$$

The latter is called Forward because the partial differential equation is written in terms of the variable X_t, *i.e.* the position of the particle, at time t. For a free Brownian motion in bulk, the Fokker-Plank equation reads:

$$\frac{\partial P(X_t, X_0; t)}{\partial t} = D_0 \frac{\partial^2}{\partial X_t^2} P(X_t, X_0; t) \ . \tag{5.1.59}$$

5.1.5 Fokker-Planck and multiplicative noise

Due to the hindered diffusion coefficient the magnitude of the random force g now depends on the particle position X_t. Therefore, for a displacement between a time t and $t+\tau$, the integration of the noise term $\int_t^{t+\tau} g(X_t)\mathrm{d}B_t$ is not trivial and the time at which the random force magnitude $g(X_t)$ is evaluated needs to be answered. In this part we derive the Fokker-Plank equation when the system is subjected to multiplicative noise. Let us write a generic Langevin equation with multiplicative noise:

$$\mathrm{d}X_t = u(X_t)\mathrm{d}t + g(X_t)\mathrm{d}B_t \ . \tag{5.1.60}$$

We start by deriving Eq. (5.1.60) for a time step τ such that:

$$X_{t+\tau} = X_t + \int_t^{t+\tau} u(X_{t'})dt' + \int_t^{t+\tau} g(X_{t'})\mathrm{d}B_{t'}\mathrm{d}t' \tag{5.1.61}$$

The expansion of the first term gives $u(X_t)\tau$ at the first order in τ. However, for the second term, the position at which $g(X_t)$ is evaluated needs to be addressed. The second integral of Eq. (5.1.61) is not unequivocally defined as [**sancho˙brownian˙2011**]:

$$\int_t^{t+\tau} g(X_{t'})\mathrm{d}B_{t'}\mathrm{d}t' \equiv B([1-\alpha]X_t + \alpha X_{t+\tau})\Delta W(t), \tag{5.1.62}$$

where $\Delta W(t)$ the the Wiener increment:

$$\Delta W(t) = \int_t^{t+\tau} \mathrm{d}B_{t'}\mathrm{d}t', \tag{5.1.63}$$

which is a Gaussian process with zero mean and variance $\langle \Delta W(t)^2 \rangle = \Delta t$. The parameter α in Eq. (5.1.62) corresponds to stochastic interpretation of the Langevin equation, and at which point in the interval $[t, t+\tau]$ the Langevin force magnitude g is evaluated. Theoretically, α can take any value between 0 and 1. However, two canonical values are usually employed : $\alpha = 0$, in the Itô convention [**ito˙stochastic˙1944**], corresponding to the use of the initial value of $g(X_t)$; $\alpha = 1/2$, in the Stratonovich convention [**stratonovich˙new˙1966**], corresponding to the mid-point value $g(X_{t+(1/2)\tau})$.

To derive the Fokker-Plank equation we now need to expand Eq. (5.1.61) to first order in τ. Combining Eqs. (5.1.61) and (5.1.62), at the order $\tau^{1/2}$, Eq .(5.1.62) can be approximated by [**sancho˙brownian˙2011**]:

$$X_{t+\tau} = X_t + g(X_t)\Delta W(t) \ . \tag{5.1.64}$$

By combining Eqs. (5.1.61) and (5.1.64) one has:

$$\begin{aligned} \int_t^{t+\tau} g(X_{t'})\mathrm{d}B_{t'}\mathrm{d}t' &= g(X_t + \alpha[X_{t+\tau} - X_t])\Delta W(t) \\ &\simeq g(X_t + \alpha g(X_t)\Delta W(t))\Delta W(t) \\ &\simeq g(X_t)\Delta W(t) + \alpha g(X_t)g'(X_t)[\Delta W(t)]^2, \end{aligned} \tag{5.1.65}$$

where $g'(X_t) = \frac{\mathrm{d}}{\mathrm{d}X_t}g(X_t)$. By finally combining Eqs. (5.1.61) and (5.1.65), one can expand Eq. (5.1.61) to the first order in τ:

$$X_{t+\tau} \simeq X_t + u(X_t)\Delta t + g(X_t)\Delta W(t) + \alpha g(X_t)g'(X_t)[\Delta W(t)]^2 \ . \tag{5.1.66}$$

Finally, by writing the Fokker-Plank equation in its standard form:

$$\frac{\partial P(X_t,t)}{\partial t} = -\frac{\partial}{\partial X_t}K_1(X_t)P(X_t,t) + \frac{1}{2}\frac{\partial^2}{\partial X_t^2}K_2P(X_t,t), \tag{5.1.67}$$

where K_1 and K_2 are the first two different moments, and invoking the Kramers-Moyal expansion [**sancho˙brownian˙2011**, **stratonovich˙new˙1966**], K_1 and K_2 are obtained by:

$$K_1(X_t) = \lim_{\tau\to 0}\frac{\langle \Delta X_t\rangle}{\tau} = u(X_t) + \alpha g(X_t)g'(X_t) \ , \tag{5.1.68}$$

and:

$$K_2(X_t) = \lim_{\tau\to 0}\frac{\langle [\Delta z]^2\rangle}{\tau} = g^2(X_t) \ . \tag{5.1.69}$$

The Fokker-Plank equation with multiplicative noise finally writes:

$$\frac{\partial P(X_t,t)}{\partial t} = -\frac{\partial}{\partial X_t}\{u(X_t) + \alpha g(X_t)g'(X_t)\}P(X_t,t) + \frac{1}{2}\frac{\partial^2}{\partial X_t^2}\{g^2(X_t)\}P(X_t,t) \ . \quad (5.1.70)$$

due to the presence of multiplicative noise, we now have a new drift velocity term that depends on the stochastic interpretation of the Langevin equation.

5.1.6 Spurious drift

Now that we derived the Fokker-Planck equation for multiplicative s noise, we can inspect how the new drift velocity term $\alpha g(X_t)g'(X_t)$ plays a role, in the case where one wants to simulate near-wall confined Brownian motion, or, infer forces from a measured trajectory. Let first rewrite Eq. (5.1.60) along the z-axis for a particle confined near a wall:

$$\mathrm{d}z = \frac{F(z)}{\gamma_\perp(z)}\mathrm{d}t + \sqrt{2D_\perp(z)}\mathrm{d}B_t \ , \quad (5.1.71)$$

where $F(x) = -k_\mathrm{B}T\ln(U(z))$ is the force (due to the DLVO interaction and gravity) exerted on the particle if the system was deterministic. Using Eq. (5.1.70), the average velocity writes:

$$\left\langle \frac{\Delta z}{\tau} \right\rangle \equiv \bar{v}_\mathrm{d} = \left\langle \frac{\mathrm{d}z}{\mathrm{d}t} \right\rangle = \frac{F(z)}{\gamma_\perp(z)} + \alpha\frac{\mathrm{d}D_\perp(z)}{\mathrm{d}z} \ . \quad (5.1.72)$$

However, one needs to take into account that at long time, the steady-state solution of the Fokker-Plank equation should be given by the Gibbs-Boltzmann equilibrium distribution $P_\mathrm{eq}(z)$. Different choice of α gives different drift velocity $\bar{v}_\mathrm{d}$ of Eq. (5.1.72), nevertheless that would mean that only one value of α permits recovering the equilibrium distribution. Yet, to my knowledge, the Itô and Stratonovich conventions permit retrieving the correct distribution.

Due to this ambiguity, it is often observed in the literature that α is numerically inferred to retrieve the correct distribution and forces. Doing so, we find in the literature some $\alpha = 1$, corresponding to an anti-Itô convention, meaning an anticipation of the Langevin equation. However, this is due to a misuse of Eq. (5.1.72) since the calculus are done

using the Itô or Stratonovich convention.

To solve this situation, one needs to take into account that it is not the deterministic force $F(z)$ that should be used in Eq. (5.1.72) but the force:

$$F_{\text{eq}} = -\frac{\mathrm{d}U_{\text{ss}}(z)}{\mathrm{d}z} \tag{5.1.73}$$

which is computed through the steady-state solution of the Fokker-Planck equation for multiplicative noise, such that:

$$U_{\text{ss}}(z) = -k_{\text{B}}T \ln(P_{\text{ss}}(z)), \tag{5.1.74}$$

with $P_{\text{ss}}(z)$ the steady-state solution of Eq. (5.1.70), that can be obtain as follows. Let us start by setting the time derivative $\partial P_{\text{ss}}/\partial t$ to zero in Eq. (5.1.70), which now writes along the z-axis:

$$\frac{\mathrm{d}}{\mathrm{d}z}\left\{-u(z) - \alpha g(z)g'(z) + \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}z}g^2(z)\right\}P_{\text{ss}}(z) = \frac{\mathrm{d}J}{\mathrm{d}z} = 0 \ , \tag{5.1.75}$$

where J is a probability flux, $u(z) = F/\gamma_{\perp}$, and $g(z)g'(z) = \mathrm{d}D_{\perp}/\mathrm{d}z$. As $\partial_z J = 0$, J needs to be a constant. Moreover, we need P_{ss} and the moments of P_{ss} to be finite, therefore we require P_{ss} to decay to zero at infinity[9], and in particular P_{ss} and $\partial_z P_{\text{ss}}$ are zero at infinity, and therefore $J = 0$ and Eq. (5.1.75) becomes:

$$\left(-u(z) - \alpha g(z)g'(z)\right)P_{\text{ss}}(z) + \frac{1}{2}\frac{\partial}{\partial z}\left\{g^2(z)P_{\text{ss}}(z)\right\} = 0 \ . \tag{5.1.76}$$

By invoking $Y = g^2(z)P_{\text{ss}}$, Eq. (5.1.76) becomes:

$$-\frac{u(z) + \alpha g(z)g'(z)}{g^2(z)/2}Y(z) + \frac{\mathrm{d}}{\mathrm{d}z}Y(z) = 0 \ , \tag{5.1.77}$$

which leads to:

[9] Using the Riemann integrals, P_{ss} should decay faster than $|z|^{-n-1}$ to have the n-th first moment to be finite.

$$P_{\mathrm{ss}}(z) = \frac{1}{g^2(z)} \exp\left[\int^z \frac{u(z') + \alpha g(z')g'(z')}{g^2(z')/2} \mathrm{d}z\right] \ . \tag{5.1.78}$$

Using the fact that $-\ln(g^2(z))$ can be written as $-\int^z 2g(z')g'(z')/g^2(z')\mathrm{d}z$, Eq. (5.1.74) becomes:

$$U_{\mathrm{ss}} = -k_{\mathrm{B}}T \int^z \frac{u(z') + (\alpha - 1)g(z')g'(z')}{g^2(z')/2} \mathrm{d}z \ . \tag{5.1.79}$$

By combining Eqs. (5.1.73) and (5.1.74), one has:

$$\frac{F_{\mathrm{eq}}(z)}{\gamma_\perp(z)} = \frac{F(z)}{\gamma_\perp(z)} + (\alpha - 1)g(z)g'(z) \ . \tag{5.1.80}$$

The difference between Eqs. (5.1.72) and (5.1.80) gives $g(z)g'(z) = \mathrm{d}D_\perp/\mathrm{dz}$, and does not depend on the interpretation of the Langevin equation. Finally, we write the overall drift as:

$$\bar{v}_{\mathrm{d}} = -\frac{1}{\gamma_\perp(z)} \frac{\mathrm{d}U(z)}{\mathrm{d}z} + \frac{\mathrm{d}D_\perp(z)}{\mathrm{d}z} = v_{\mathrm{d}} + v_{\mathrm{spurious}} \ , \tag{5.1.81}$$

where, to recap, the first term v_{d} is the drift velocity due to deterministic forces (electrostatic and gravitational interactions in our case). The spurious drift velocity v_{spurious} disappears when the diffusion coefficient is homogeneous, and would also disappear along the x- and y-axes since the diffusion coefficients only depend on the colloid-wall distance z (a derivative with respect to x would replace the one in z in the equivalent of Eq. (5.1.81) for the x-direction). To conclude on that subject, and as pointed out by Mennella *et al.* [**mannella˙ito˙2012**], this derivation is not formal as it requires several approximations. However, the result is still correct. A formal derivation has already been done [**sancho˙adiabatic˙1982**] and is done by taking the Fokker-Planck of the underdamped Langevin equation. Then they formally compute the limit of the underdamped Fokker-Planck equation when the mass of the particle tends to zero, using a mathematical method called adiabatic reduction.

Finally, Using Eqs. (5.1.45) (5.1.48) and (5.1.81), the deterministic drift velocity reads:

$$v_{\mathrm{d}} = -\frac{k_{\mathrm{B}}T}{\gamma_{\perp}(z)}\left[-\frac{1}{\ell_{\mathrm{D}}}B\exp\left(-\frac{z}{\ell_{\mathrm{D}}}\right)+\frac{1}{\ell_{\mathrm{B}}}\right] , \tag{5.1.82}$$

and the spurious drift velocity reads:

$$v_{\mathrm{noise}}(z) = 2\alpha D_0 a\frac{2a^2+12az+21z^2}{(2a^2+9az+z^2)^2} , \tag{5.1.83}$$

A typical example of drift velocity $\bar{v}$ for a colloidal particle of radius $a = 1.5$ μm in water, moving near a wall, and interacting with the latter through an electrostatic potential with a Debye length $\ell_{\mathrm{D}} = 50$ nm and a dimensionless magnitude of the interaction $B = 4$, and evolving in a gravity field characterized by a Boltzmann length $\ell_{\mathrm{B}} = 500$ nm, is plotted in Fig. 5.1.8. As one can observe, the spurious drift is not negligible.

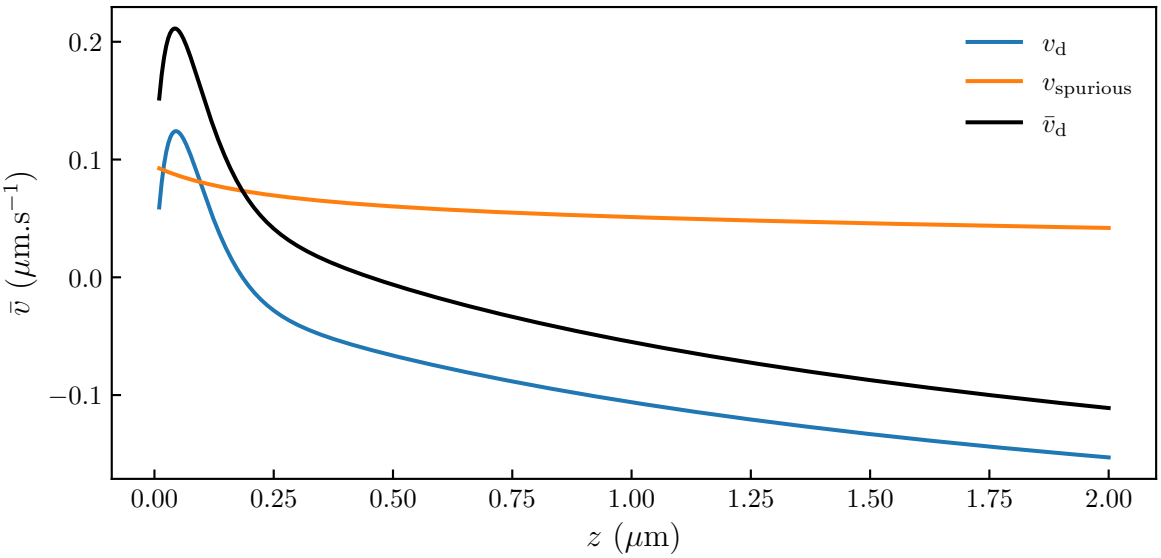


Figure 5.1.8: Theoretical drift velocity for a colloidal particle of radius $a = 1.5$ μm in water and at a distance z from the wall. The physical parameters $\ell_{\mathrm{D}} = 50$ nm, $B = 4$ and $\ell_{\mathrm{B}} = 500$ nm.

5.1.7 Numerical simulations of confined Brownian motion

We previously determined that the simulation of a bulk Brownian motion without external forces can be simulated using Eq. (3.4.9). However, in the case of confined Brownian motion, and without density matching, one needs to take into account the hindered mobility, external forces due to gravity and the double-layer interaction, as well as the confinement-induced spurious drift. Putting all that together leads to a new equation for x_i which

reads for the motion parallel to wall (*i.e.* along the x- and y-axes):

$$x_i = x_{i-1} + \sqrt{2D_{\parallel}} w_i \ , \tag{5.1.84}$$

where we recall that we approximate the continuous position X_t of a particle at a time t by a discrete-time sequence x_i, which is the solution of the equation at a time $t_i = i\tau$. τ being the numerical integration time increment, and w_i is a Gaussian-distributed number of mean value $\langle w_i \rangle = 0$ and variance $\langle w_i^2 \rangle = \tau$. For the perpendicular motion of the particle (along the z-axis), one needs to add the total drift $\bar{v}_{\mathrm{d}}$ of Eq. (5.1.81), such that:

$$z_i = z_{i-1} + \bar{v}_{\mathrm{d}}(z_{i-1})\tau + \sqrt{2D_{\perp}} w_i \ , \tag{5.1.85}$$

where z_i is the discrete-time position sequence of a particle along the z-axis. Compared to bulk Brownian motion where the time step τ can be chosen only according to the desired precision as shown previously on Fig. 3.4.2, confinement adds another constraint. Indeed, the time step should be short enough for the drift $\bar{v}_{\mathrm{d}}$ and local diffusion coefficients to be relatively constant (as detailed later) in the time period $t_{i+1} - t_i = \tau$ and in the displacement range $\Delta z = z_{i+1} - z_i$, such that:

$$\bar{v}_{\mathrm{d}}(z \in [z_i, z_{i+1}]) \simeq \bar{v}_{\mathrm{d}}(z_i) \ , \tag{5.1.86}$$

and:

$$D_{\perp,\parallel}(z \in [z_i, z_{i+1}]) \simeq D_{\perp,\parallel}(z_i) \ . \tag{5.1.87}$$

Since the diffusion coefficient does not vary for the parallel motion, one can consider only the perpendicular motion to determine the optimal simulation time step. Also, as it can be seen in Fig. 5.1.10 the relative variation of the drift velocity $1/\bar{v}_{\mathrm{d}}\partial_z \bar{v}_{\mathrm{d}}$ reaches higher values than the relative variation of the diffusion coefficient $1/D_{\perp}\partial_z D_{\perp}$. Thus, finding τ that satisfies Eq. (5.1.86) is sufficient. Moreover, the vertical drift velocity varies more when the colloid is near the surface, *i.e.* in the region where one can approximate the diffusion coefficient $D_{\perp}$ using Eqs. (5.1.45) and (5.1.49):

$$D_\perp(z)|_{z\ll a} = D_0\frac{z}{a} \ . \tag{5.1.88}$$

In that case, Eq. (5.1.81) near the surface simplifies to:

$$\begin{aligned}\bar{v}_\mathrm{d} &\simeq \frac{k_\mathrm{B}T}{\gamma_0}\frac{z}{a}\left[\frac{B}{\ell_\mathrm{D}}\exp\left(-\frac{z}{\ell_\mathrm{D}}\right) - \frac{1}{\ell_\mathrm{B}}\right] + \frac{\partial}{\partial z}D_0\frac{z}{a} \\ &= \frac{D_0}{a}\left[1 + \frac{Bz}{\ell_\mathrm{D}}\exp\left(-\frac{z}{\ell_\mathrm{D}}\right) - \frac{1}{\ell_\mathrm{B}}\right]\end{aligned} \tag{5.1.89}$$

By expanding the exponential term at the first order in z/ℓ_D, we get:

$$\bar{v}_\mathrm{d} = \frac{D_0}{a}\left(1 + \frac{Bz}{\ell_\mathrm{D}} - \frac{1}{\ell_\mathrm{B}}\right) \tag{5.1.90}$$

To satisfy Eq. (5.1.86), we need to have a small relative change of the vertical drift velocity in an interval $[z, z+\Delta z]$ [**matse˙state-dependent˙nodate**], *i.e.*:

$$\frac{|\bar{v}_\mathrm{d}(z+\Delta z) - \bar{v}_\mathrm{d}(z)|}{|\bar{v}_\mathrm{d}(z)|} \ll 1 \ . \tag{5.1.91}$$

Combining Eqs. (5.1.90) and (5.1.91), we get:

$$|\Delta z| \ll \left[\frac{B}{\ell_\mathrm{D}}\right]^{-1} + z \ . \tag{5.1.92}$$

Besides, invoking the vertical MSD over the time step, as well as Eq. (5.1.88), one gets:

$$\langle \Delta z^2\rangle(z) = 2D_\perp(z)\tau = 2D_0\frac{z}{a}\tau \ . \tag{5.1.93}$$

Combining Eqs. (5.1.92) and (5.1.93) thus leads to:

$$\tau = \frac{a\langle\Delta z^2\rangle}{2D_0 z} \ll \frac{a}{2D_0}\frac{\left[\left(\frac{B}{\ell_\mathrm{D}} - \frac{1}{\ell_\mathrm{B}}\right)^{-1} + z\right]^2}{z} = \tau_\mathrm{max}(z) \ . \tag{5.1.94}$$

At this point, there are two different options for the time step in the simulation: the first one is to do an adaptive time step using a local $\tau(z)$ that satisfies $\tau(z) \ll \tau_{\max}(z)$ for each step of the simulation; the second one is to find the smallest $\tau_{\max}(z)$ and use for all the simulation a time step τ satisfying $\tau \ll \min(\tau_{\max})$. The latter can be evaluated by finding the height $z_{\min}$, at which the derivative of $\tau_{\max}$ nullifies, *i.e.*:

$$\left.\frac{\partial \tau_{\max}}{\partial z}\right|_{z_{\min}} = 0 \ . \tag{5.1.95}$$

Solving the latter gives,

$$z_{\min} = \left| \left(\frac{B}{\ell_{\mathrm{D}}} - \frac{1}{\ell_{\mathrm{B}}} \right)^{-1} \right| \ . \tag{5.1.96}$$

which finally gives:

$$\min(\tau_{\max}) = \frac{2a}{D_0} \left| \left(\frac{B}{\ell_{\mathrm{D}}} - \frac{1}{\ell_{\mathrm{B}}} \right)^{-1} \right| \ . \tag{5.1.97}$$

In Fig. 5.1.10-b) $\tau_{\max}$ is plotted as a function of z, for $a = 1.5$ μm, $B = 4$ and ℓ_{D} varying between 20 and 100 nm. We observe that for this range of values that represents well the experiments that I have performed during my thesis, taking a constant simulation time step $\tau \approx 0.01$ s is satisfactory.

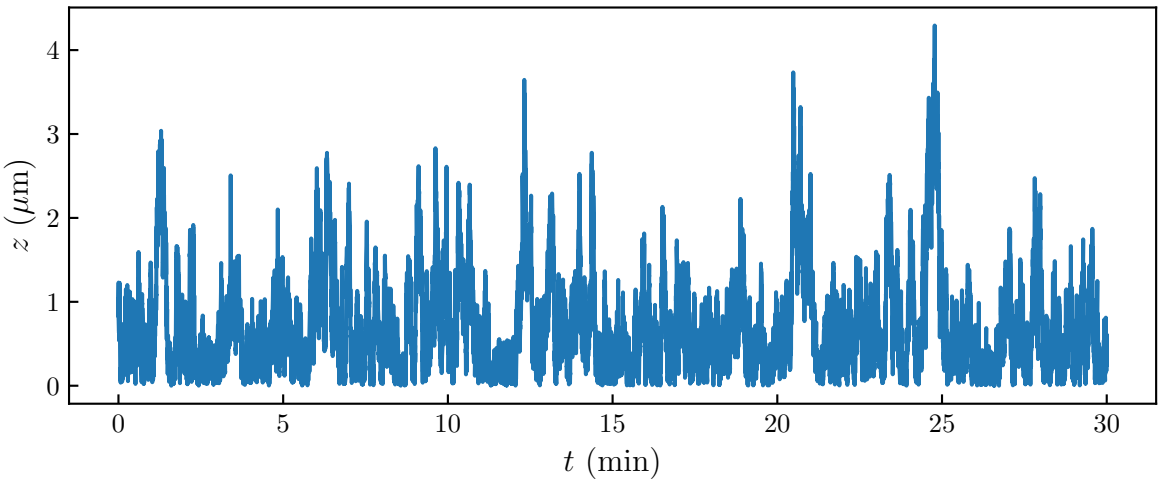


Figure 5.1.9: Simulated trajectory along the direction normal to the wall, using Eq. (5.1.85) with $\alpha = 1$ (isothermal convention), of radius $a = 1.5$ μm, density $\rho_p = 1050$ kg.m^{-3}, placed in water near a wall. The particle-wall interaction is characterized by $\ell_D = 50$ nm and $B = 4$.

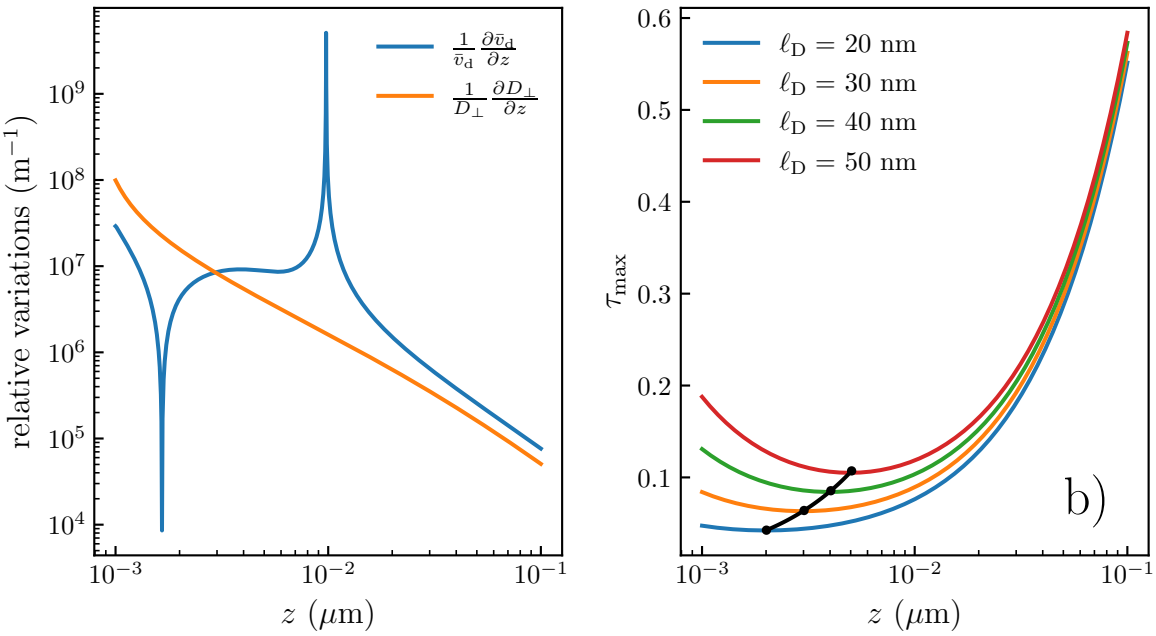


Figure 5.1.10: a) Theoretical computation of the relative variation of the drift velocity $\bar{v}_d$ and perpendicular diffusion coefficient $D_\perp$ as a function particle-wall distance z. The parameters are $\ell_D = 50$, $B = 4$ and $\ell_B = 500$ nm b) Optimal numerical integration time step τ_{max} as a function of the particle-wall distance z for a particle of radius $a = 1.5$ μm, with $\ell_B = 500$ nm, $B = 4$, and for different Debye lengths as indicated. The black line connects the minima.

We have developed the numerical simulation of Eqs. (5.1.84) and (5.1.85) using Python, as part of the Master's internship of Élodie Millan. The interested reader will find more

information on the simulation of confined Brownian motion in complex systems in her forthcoming thesis. A typical trajectory of a water-immersed colloidal particle of radius $a = 1.5$ μm and density $\rho_\mathrm{p} = 1050$ kg.m^{-3}, near a wall with which the electrostatic interaction is characterized by $\ell_\mathrm{D} = 50$ nm and $B = 4$, is plotted in Fig. 5.1.9. It qualitatively resembles the experimental trajectory that was shown in Fig. 5.1.1 as an introduction to the chapter. In this trajectory, the noise-induced lift is taken into account by using the isothermal convention (*i.e.* $\alpha = 1$). To check if the spurious drift is correctly implemented, the constraint we have is that the long-time statistics should satisfy Eq. (5.1.29). To compute an experimental probability density function from a set of points, one can use the following Python snippet.

```python
def pdf(data, bins=10, density=True):

  pdf, bins_edge = np.histogram(data, bins=bins, density=density)
  bins_center = (bins_edge[0:-1] + bins_edge[1:]) / 2

  return pdf, bins_center
```

The long-time PDF in position, for with and without the spurious drift v_{spurious} are shown in Fig. 5.1.11. We see that the spurious drift velocity v_{spurious} permits retrieving the correct distribution. In the case ($v_{\text{spurious}} = 0$), we observe that the particle is more likely to be found closer to the surface, as a result of the missing compensating spurious drift of Eq. (5.1.83).

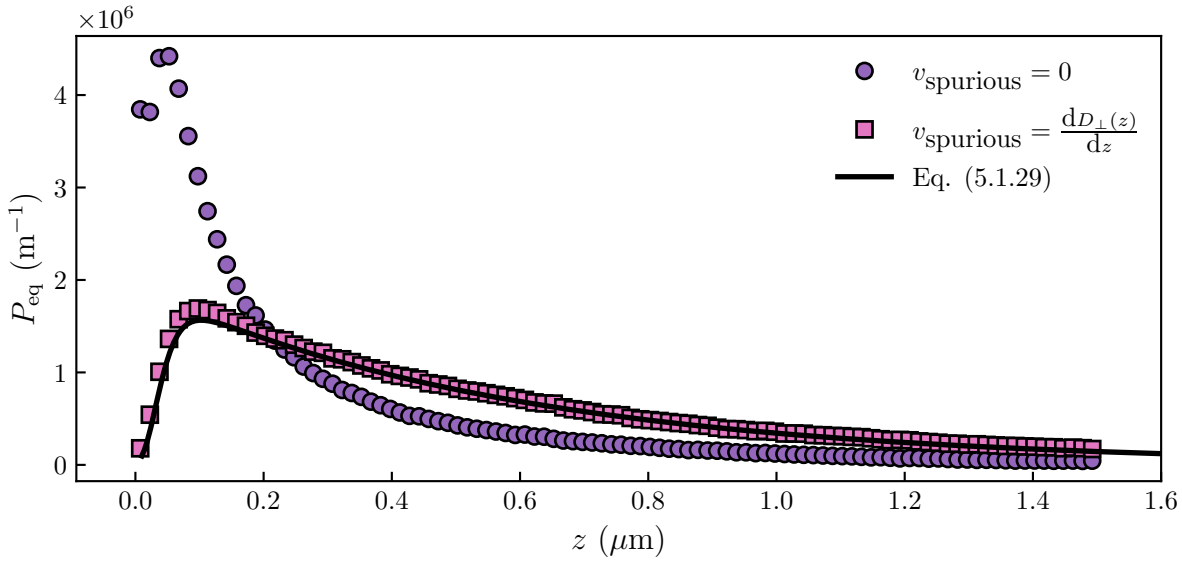


Figure 5.1.11: Simulated, Long-term Probability Density Function of the height of the particle with or without the spurious drift, as indicated. The solid black line represents the expected Gibbs-Boltzmann distribution. The physical parameters are: $a = 1.5$ μm, $\rho_{\text{p}} = 1050$ kg.m^{-3}, $\ell_{\text{D}} = 21$ nm, $B = 4.8$ and $\ell_{\text{B}} = 577$ nm.

5.2 Experimental study

Let us now analyze the experimental data acquired through the Mie tracking (see section 4.3). In the near-wall Brownian dynamic theory presented section 5.1, we considered distance z between the wall and the colloidal particle surface. However, it is not the height measured by the Mie tracking, since the latter measures the distance between the objective-lens focal plane and the particle center. Therefore, we measure the trajectory

up to an offset, *i.e.* the objective-lens focal plane to wall distance. To have the correct measured height, we suppose that the particle does approach very closely to the wall, such that the minimal measured distance is the focal plane to the wall distance. From that assumption, we set the minimal value of z in the trajectory to zero. This is repeated periodically and thus termed the moving-minimum method, and can be calculated using the following Python function.

```
def movmin(z, window):
  result = np.empty_like(z)
  start_pt = 0
  end_pt = int(np.ceil(window / 2))

  for i in range(len(z)):
    if i < int(np.ceil(window / 2)):
      start_pt = 0
    if i > len(z) - int(np.ceil(window / 2)):
      end_pt = len(z)

    result[i] = np.min(z[start_pt:end_pt])
    start_pt += 1
    end_pt += 1
  return result
```

In the above snippet, `window` represents the number of points used to compute the minimum. As an example, if one chooses `window = 100`, the first value of `result` is the minimum of the first 100 points of `z`. If there is enough data around the point where the minimum is calculated, the ensemble is centered, with a new window of size 100 (*i.e.* `result[100] = np.min(z[50:150])`. If there is not enough point around the n-th point, we then take the average value of the first (or the last) n points. The raw and shifted trajectories are shown in Fig. 5.2.1. Moreover, subtracting the moving minimum has a benefit. Indeed, it can remove some experimental drift due to the mechanical movement of the optical pieces of the microscope. Also, due to the approximation made to use the moving-minimum method, the exact location of the $z = 0$ origin is *a priori* undetermined we need to add to the physical parameters B, ℓ_{D} and ℓ_{B} a forth parameter: the height offset z_{off} that accounts for the correction of the wall position.

5.2.1 Equilibrium distribution

As we have done for the simulated trajectory, one can construct the equilibrium probability density function $P_{\mathrm{eq}}(z)$ of the position of the particle. As seen in Fig. 5.2.2, and explained

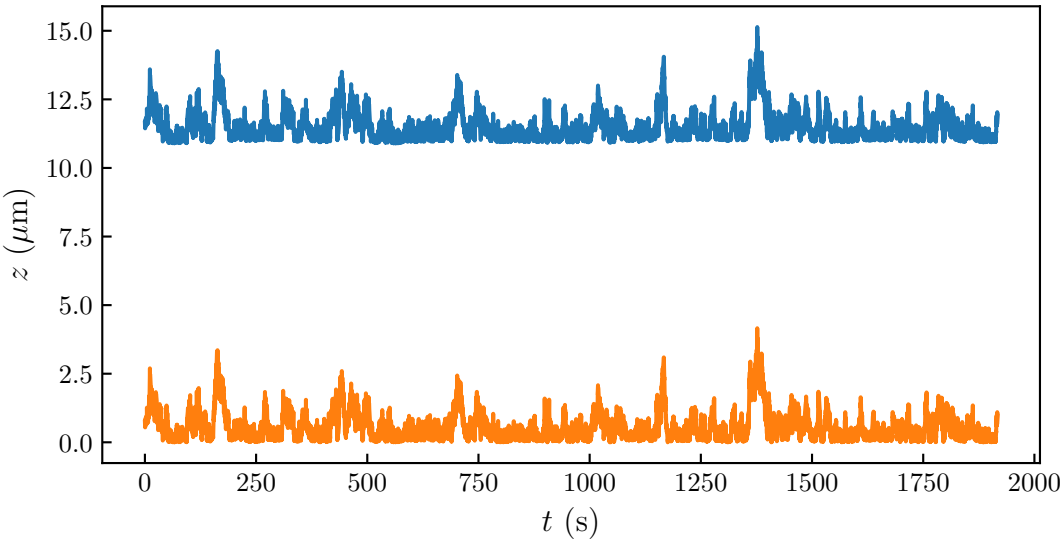


Figure 5.2.1: Raw trajectories measured using the Mie-tracking technique, and its shifted version (bottom, orange) using the moving-minimum method with a window of 10000 points.

in section 5.1.1, an exponential tail is observed at large distance, which is identified to the sedimentation contribution in Perrin's experiment [**perrin˙les˙2014**], but here with the probability density function of a single particle instead of the concentration field. In contrast, near the wall, we observe an abrupt depletion, indicating a repulsive electrostatic contribution. Additionally, we see that the Gibbs-Boltzmann distribution of Eq. (5.1.29) fits the data very well.

Moreover, as shown in Fig. 5.2.3, we recover the Debye relation, *i.e.* $\ell_\mathrm{D} = 0.304/\sqrt{[\mathrm{NaCl}]}$ (see Eq. (5.1.13)), with ℓ_D in nm, and where [NaCl] is the concentration of salt in mol/L, with a prefactor corresponding to a single monovalent salt in water at room temperature [**israelachvili˙intermolecular˙2015**]. Besides, we have verified, as shown in Fig. 5.2.3, that the dimensionless parameter B related to surface charges is constant in the studied salt-concentration range, thus excluding any nonlinear effect [**wang˙measurement˙2011**, **oberholzer˙grand˙1997**] in our case.

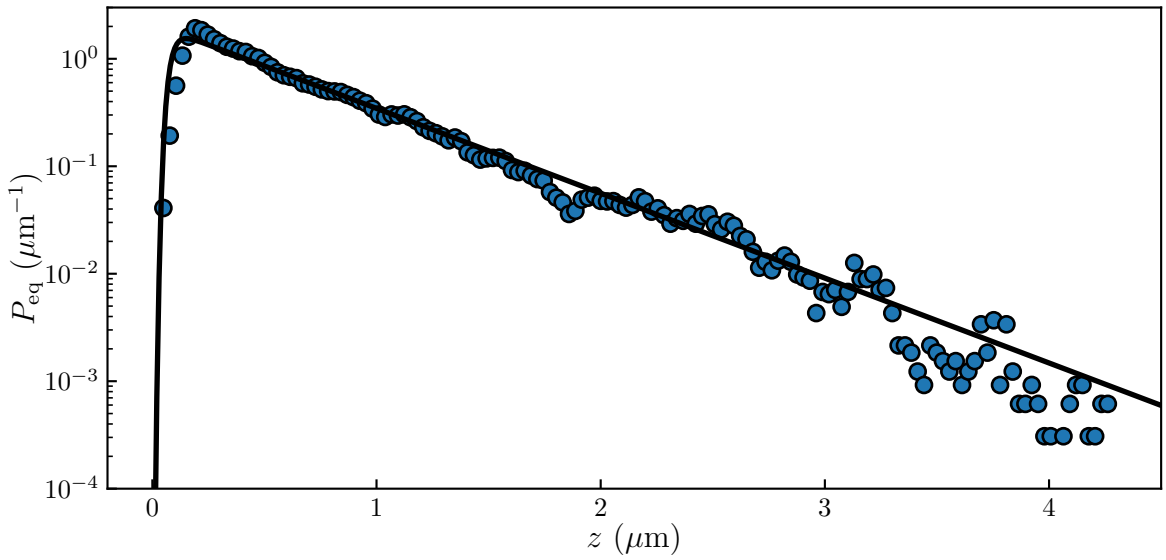


Figure 5.2.2: Measured equilibrium probability density function P_{eq} of the distance z between the particle and the wall. The solid line represents the best fit to the normalized Gibbs-Boltzmann distribution in position, using the total potential energy $U(z)$ of Eq. (5.1.28), with $B = 4.8$, $\ell_{\mathrm{D}} = 21$ nm, and $\ell_{\mathrm{B}} = 530$ nm.

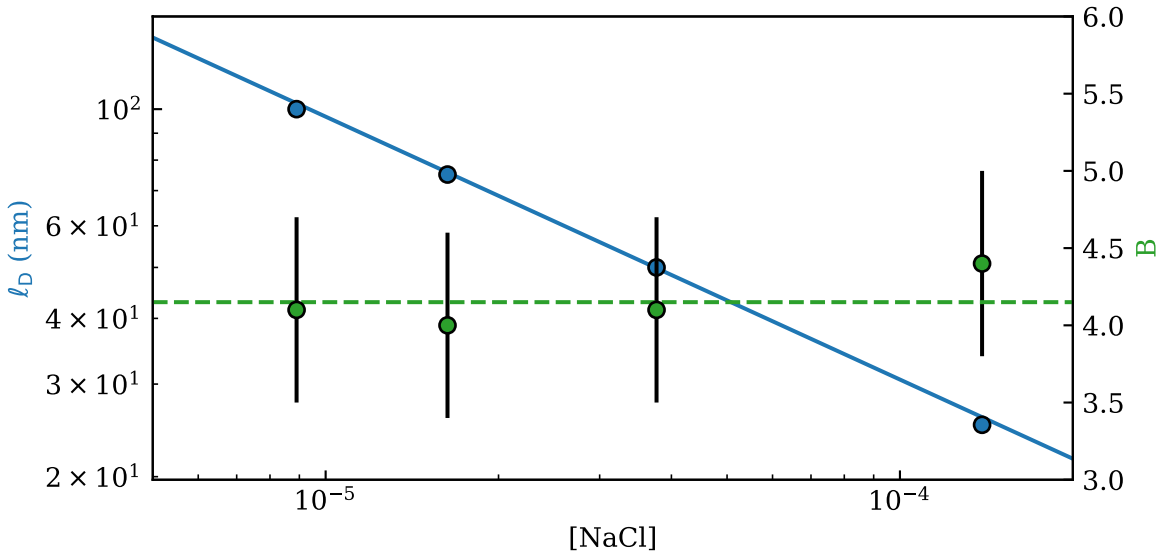


Figure 5.2.3: In blue, left axis, measured Debye length ℓ_{D} as a function of salt concentration [NaCl]. The solid line is the expected Debye relation $\ell_{\mathrm{D}} = 0.304/\sqrt{[\mathrm{NaCl}]}$, for a single monovalent salt in water at room temperature. In green, right axis, measured B as a function of salt concentration [NaCl]. The dashed line represents the mean value of the measured B values.

5.2.2 Mean Square Displacement

We now turn to dynamical aspects, by considering the mean-squared displacement (MSD). We recall that, for the three spatial directions, indexed by $i = x$, y, and z, corresponding to the coordinates $r_x = x$, $r_y = y$, and $r_z = z$, of the position $\vec{r}$, and for a given time increment Δt, the MSD is defined as:

$$\langle \Delta r_i(t)^2 \rangle_t = \langle [r_i(t + \Delta t) - r_i(t)]^2 \rangle_t \ , \tag{5.2.1}$$

where the average $\langle \rangle_t$ is performed over time t. For a free Brownian motion in the bulk, and in the absence of other forces than the dissipative and random ones, the MSD is linear in time, *i.e.* $\langle \Delta r_i(t)^2 \rangle_t = 2D_0 \Delta t$, where is the bulk diffusion coefficient (see Eq. (5.1.33)) given by the Stokes-Einstein relation [**einstein˙uber˙1905**]. Further including sedimentation restricts the validity of the linearity of the MSD along the z-axis to short times only, *i.e.* for $\Delta t \ll \ell_B^2/D_0$ such that the vertical diffusion is not yet affected by the gravitational drift.

As explain in section 5.1.2.1 The presence of a rigid wall at $z = 0$ adds a repulsive electrostatic force along z. It also decreases the mobilities nearby through hydrodynamic interactions (see section 5.1.3), where we recall that it leads to effective viscosities $\eta_{\parallel}(z) = \eta_x(z) = \eta_y(z)$, and $\eta_{\perp}(z) = \eta_z(z)$ (see Eq.5.1.45). Interestingly, despite the previous modifications, the temporal linearity of the MSD is not altered by the presence of the wall [**chubynsky˙diffusing˙2014**, **prieve˙measurement˙1999**] for x and y, as well as at short times for z. In such cases, the MSD reads:

$$\langle \Delta r_i(t)^2 \rangle_t = 2\langle D_i \rangle \Delta t \ , \tag{5.2.2}$$

where we introduced the average of the local diffusion coefficient:

$$\langle D_i(z) \rangle = \int_0^{\infty} \mathrm{d}z \, D_i(z) P_{\mathrm{eq}}(z) \ , \tag{5.2.3}$$

against the Gibbs-Boltzmann distribution in position. As shown in Fig. 5.2.4, the MSD measured along x or y is indeed linear in time. By fitting the data to Eq. (5.2.2), using Eqs. (5.1.28) and (5.1.47), we extract an average transverse diffusion coefficient $\langle D_{\parallel} \rangle = \langle D_x \rangle = \langle D_y \rangle = 0.52\, D_0$. In contrast, along z, we identify two different regimes: one at short times, where the MSD is still linear in time, with a similarly-obtained best-fit value of $\langle D_z \rangle = 0.24\, D_0$; and one at long times, where the MSD saturates to a plateau. This latter

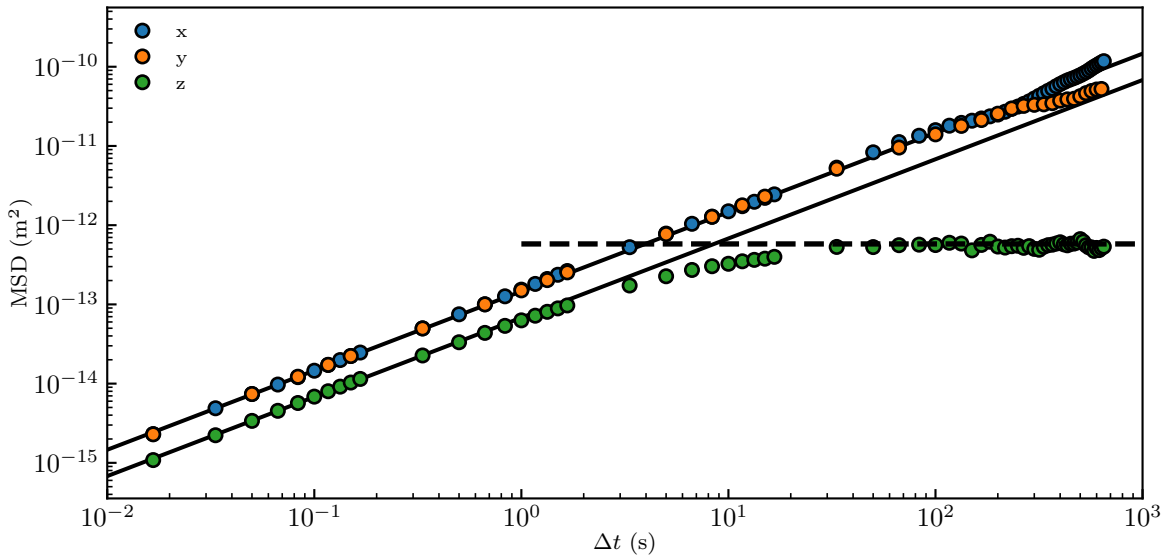


Figure 5.2.4: Measured mean-squared displacements (MSD, see Eq. (5.2.1)) as functions of the time increment Δt, for the three spatial directions, x, y, and z. The solid lines are best fits to Eq. (5.2.2), using Eqs. (5.1.28), (5.1.46) and (5.1.47), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm, providing the average diffusion coefficients $\langle D_\parallel \rangle = \langle D_x \rangle = \langle D_y \rangle = 0.52\, D_0$ and $\langle D_z \rangle = 0.24\, D_0$. The dashed line is the best fit to Eq. (5.2.9), using Eq. (5.1.28), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm.

behavior indicates that the equilibrium regime has been reached, with the particle having essentially explored all the relevant positions given by the Gibbs-Boltzmann distribution.

5.2.3 Non-Gaussian dynamics - displacement distribution

Having focused on the MSD, *i.e.* on the second moment only, we now turn to the full-Probability Density Function P_i of the displacement Δr_i. Since the diffusion coefficient $D_i(z)$ varies as a result of the variation of z along the particle trajectory, P_i exhibits a non-Gaussian behavior, as seen in Figs. 5.2.5-a,b,c,d). We even resolve the onset of a non-Gaussian behavior in P_x, by zooming on the large-$|\Delta x|$ wings. At short times, the diffusion coefficient D_i and the drift velocity $\bar{v}_d$, can be considered constant. By writing the initial condition of the particle-wall distance $\delta(z - z_0)$, the solution of Eq. (5.1.70) becomes:

$$\begin{aligned} P_z(z, z_0, \Delta t) &= \exp\left[\frac{\partial^2}{\partial z^2} D_\perp(z_0)\Delta t - \frac{\partial}{\partial z}\bar{v}_d(z_0)\Delta t\right] \frac{1}{2\pi}\int_{-\infty}^{\infty} du \exp(ju(z - z_0)) \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} du \exp\left[-u^2 D_\perp(z_0)\Delta t + ju(z - z_0) - ju\bar{v}_d(z_0)\Delta t\right] . \end{aligned} \tag{5.2.4}$$

The latter can be reduced to [**matse˙state-dependent˙nodate**, **risken˙fokker-planck˙2012**]:

$$P_z(\Delta z, z_0, \Delta t) = \frac{1}{\sqrt{4\pi D_i(z_0)\Delta t}} \exp\left[-\frac{(\Delta r_i - \bar{v}_\mathrm{d}\Delta t)^2}{4D_i(z_0)\Delta t}\right] , \tag{5.2.5}$$

which is a Gaussian distribution with a mean value $\langle \Delta z \rangle = \bar{v}_\mathrm{d}\Delta t$. The same calculus can be done for the x- and y-axes, by setting the drift velocity to zero and using $D_\perp$. Additionally, it has a standard deviation $\sigma_i(z_0) = \sqrt{2D_i(z_0)\Delta t}$. From Eq. (5.2.5), we can observe than the total drift $\bar{v}_\mathrm{d}$ induces an asymmetry on the displacement along the z-axis. However, in our experiment, as we have access to long-enough trajectories to reach equilibrium, the statistics are not conditioned by the initial position, but by the Gibbs-Boltzmann distribution of Eq. (5.1.29). At short times, P_i can thus be modeled by the averaged diffusion Green's function [**matse˙test˙2017**, **hapca˙anomalous˙2009**]:

$$\begin{aligned} P_i(\Delta r_i, \Delta t) &= \int_0^\infty \mathrm{d}z P_\mathrm{eq}(z) P(\Delta r_i, z, \Delta t) \\ &= \int_0^\infty \mathrm{d}z\, P_\mathrm{eq}(z) \frac{1}{\sqrt{4\pi D_i(z)\Delta t}} \mathrm{e}^{-\frac{\Delta r_i^2}{4D_i(z)\Delta t}} , \end{aligned} \tag{5.2.6}$$

against the Gibbs-Boltzmann distribution. The latter equation can alternatively be written as an integral over the diffusion coefficient such that:

$$P_i(\Delta r_i, \Delta t) = \int_0^\infty \mathrm{d}D_i P(D_i) \frac{1}{\sqrt{4\pi D_i \Delta t}} \mathrm{e}^{\frac{-\Delta r_i^2}{4D_i\Delta t}} \tag{5.2.7}$$

This equation can be evaluated using the following Python snippet.

```python
def P_D(B, ld, lb):
  # Computing the D PDF.
  z = np.linspace(1e-9, 15e-6, 1000)
  P_D = Dz(z) * P_b_off(z, offset, B, ld, lb)
  P_D = P_D / np.trapz(P_D, z)  # extra step to ensure PDF normalization
  return Dz, P_D


def _P_Dz_short_time(Dz, Dt, B, ld, lb):
  # Using the D PDF to compute P()
  D_z, P_D = P_D(B, ld, lb)
  P = P_D / np.sqrt(4 * np.pi * D_z * Dt) * np.exp(-(Dz ** 2) / (4 * D_z * Dt))
  P = np.trapz(P, D_z)
  return P


```

```
        # Creating a handy function for easier use with Dz numpy arrays
def P_Dz_short_time(Dz, Dt, B, ld, lb):
  P = np.array([_P_Dz_short_time(i, Dt, B, ld, lb) for i in Dz])
  P = P / np.trapz(P, Dz) # extra step to ensure PDF normalization
  return P
```

In this snippet, the evaluation is done for Δz. However, when computing $P_x(\Delta x)$ one should just change the `Dz(z)` function to $D_{\parallel}$. Since P is a PDF, it should be normalized such that $\int P = 1$. We added an extra step to ensure PDF normalization along the evaluation. At long enough time, since we have reached equilibrium, the averaged particle's drift should be equal to zero thus leading to a mean value $\langle \Delta r_i \rangle_t = 0$. As shown in Figs.5.2.5-a,c,b,d) Eq. (5.2.6) captures the early data very well. At long times, Eq. (5.2.6) remains valid only for P_x and P_y. Nevertheless, the equilibrium regime being reached, P_z only depends on the Gibbs-Boltzmann distribution. Indeed, in this regime P_z can be written as a convolution of two Gibbs-Boltzmann distribution as in a displacement $\Delta z = z(t + \Delta t) - z(t)$, $z(t + \Delta t)$ and $z(t)$ comes from independent draw from the Gibbs-Boltzmann distribution:

$$\lim_{\Delta t \to \infty} P_z(\Delta z, \Delta t) = \int_0^{\infty} \mathrm{d}z \, P_{\mathrm{eq}}(z + \Delta z) P_{\mathrm{eq}}(z) \ , \tag{5.2.8}$$

which contains in particular the second moment:

$$\lim_{\Delta t \to \infty} \langle \Delta z^2 \rangle = \int_{-\infty}^{+\infty} \mathrm{d}\Delta z \, \Delta z^2 \int_0^{\infty} \mathrm{d}z \, P_{\mathrm{eq}}(z + \Delta z) P_{\mathrm{eq}}(z) \ . \tag{5.2.9}$$

As shown in Fig. 5.2.5-e), Eq. (5.2.8) captures the long-term data along z very well. Additionally, Eq. (5.2.9) permits to fit the long-term plateau of the MSD shown in Fig. 5.2.4. Eq. (5.2.8) can be evaluated using the following Python function:

```
def _Pdeltaz_long(DZ, B, ld, lb):
  z = np.linspace(0, 20e-6, 1000)
  dP = P_eq(z, B, ld, lb) * P_eq(z + DZ, B, ld, lb)
  P = trapz(dP,z)
  return P

def Pdeltaz_long(DZ, B, ld, lb):
  pdf = np.array([_Pdeltaz_long(i,B, ld, lb) for i in DZ])
  pdf = pdf / trapz(pdf,DZ)
  return pdf

```

where the `P_eq` function has been described in section 5.1.2.

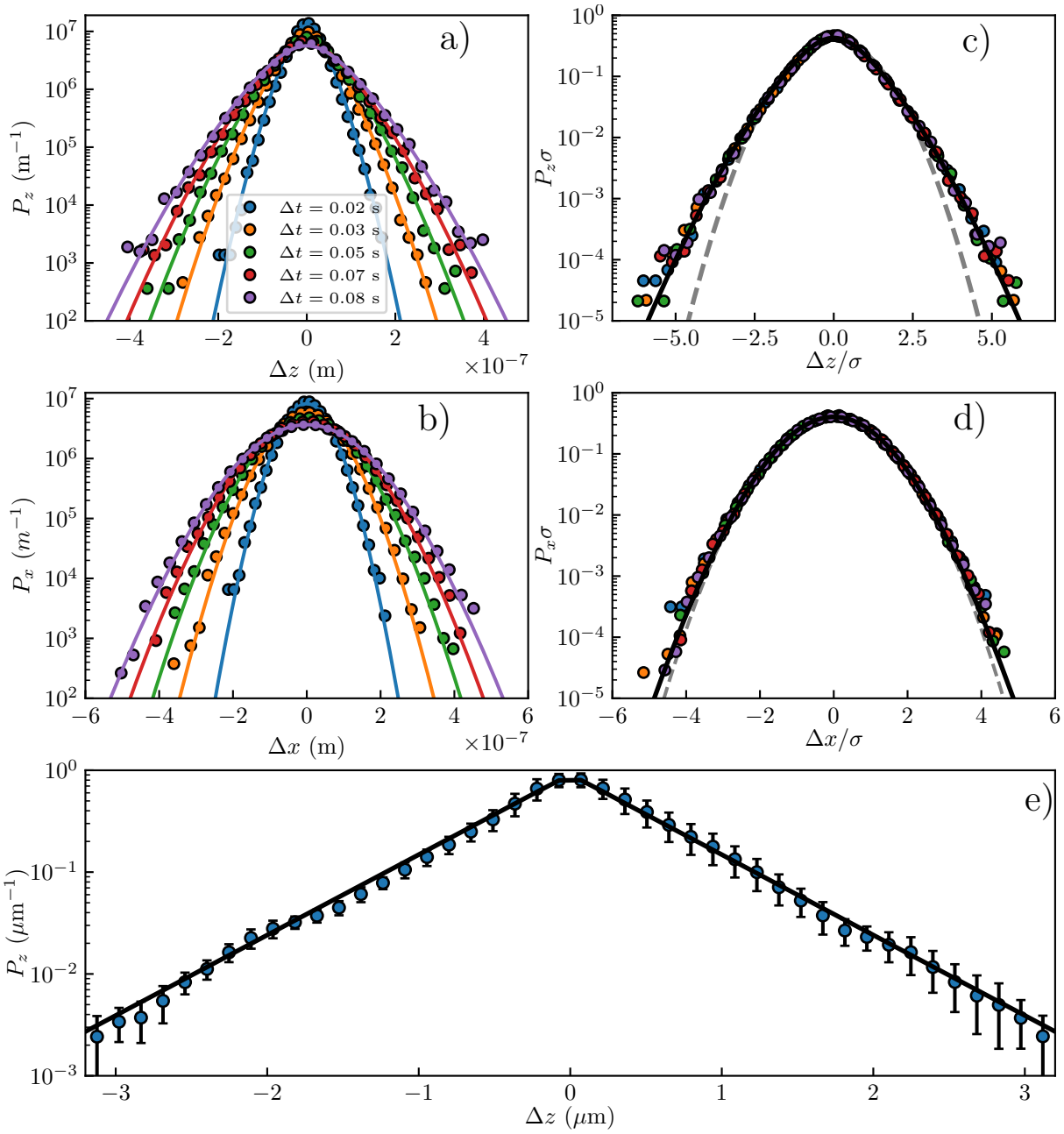


Figure 5.2.5: a, b) Probability density functions P_i of the displacements Δx and Δz, at short times. The solid lines are the best fits to Eq. (5.2.6), using Eqs. (5.1.29), (5.1.46), and (5.1.47), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm. c,d) Normalized probability density functions $P_i\,\sigma$ of the normalized displacements $\Delta x/\sigma$ and $\Delta z/\sigma$, at short times, with σ^2 the corresponding MSD (see Fig. 5.2.4), for different time increments Δt ranging from 0.0167 s to 0.083 s, as indicated with different colors. The solid lines are the best fits to Eq. (5.2.6), using Eqs. (5.1.28), (5.1.46), and (5.1.47), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm. For comparison, the gray dashed lines are normalized Gaussian distributions, with zero means and unit variances. e) Probability density function P_z of the displacement Δz, at long times, averaged over several values of Δt ranging between 25 s and 30 s. The solid line is the best fit to Eq. (5.2.8), using Eq. (5.1.28), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm.

5.2.4 Local diffusion coefficient

We now wish to go beyond the previous average $\langle D_i \rangle$ of Eq. (5.2.2), and resolve the local diffusion coefficient $D_i(z)$. To measure local viscosities from experimental trajectories, a binning method is generally employed [**friedrich˙approaching˙2011**]. This method computes the average $\langle (\Delta r_i^2)(z) \rangle_t$ locally over a z-binning grid such that the local diffusion coefficient writes:

$$D_i(z) = \frac{\langle \Delta r_i^2 \rangle_t(z)}{\Delta t} \ . \tag{5.2.10}$$

Using this method, Vestergard *et al.* [**vestergaard˙estimation˙2015**] discovered that the obturation time of the camera and the localization error σ_{r_i} plays an important role in the determination of the local diffusion coefficient such that Eq. 5.2.10 should be rewritten as:

$$D_i(z) = \frac{\langle \Delta r_i^2 \rangle_t(z) - 2\sigma_{\mathrm{r}_i}^2}{2(1-R)\Delta t} \ . \tag{5.2.11}$$

where R is a motion blur coefficient that depends on the aperture time of the camera. Let us write τ the time lapse between the capture of two images, and $\zeta(t)$ the state of the camera shutter during this time-lapse. $\zeta(t) > 0$ indicates that an open shutter while $\zeta(t) = 0$ indicates a close shutter. The scale $\zeta(t)$ is fixed by the normalization condition $\int_0^\tau \zeta(t)\mathrm{d}t$. The motion blur coefficient is given by:

$$R = \frac{1}{\tau} \int_0^\tau S(t)[1 - S(t)]\mathrm{d}t, \tag{5.2.12}$$

where $S(t) = \int_0^t \zeta(t')\mathrm{d}t'$. If the shutter is kept open for the whole duration of the time-lapse, one has $R = 1/6$. The localization error σ_{r_i} can be determined from a measured trajectory. Taking into account that Brownian motion should not be correlated, one can measure σ_{r_i} by calculating the autocorrelation $x_i(n)$ as a function of the number of time steps n of the particle position r_i, which we defined as:

$$x(n) = \langle (r_i(t) - \langle r_i(t) \rangle_t)(r_i(t + n\Delta t) - \langle r_i(t) \rangle_t) \rangle_t \ . \tag{5.2.13}$$

In [**vestergaard˙estimation˙2015**] the localization error (also called Vestergaard error)

is written as:

$$\sigma_{\mathrm{r}_i} = \sqrt{x(1) - \frac{2\langle \Delta r_i^2 \rangle_t}{1-R}} \tag{5.2.14}$$

Moreover, mechanical drifts (or evaporation driven-flow in the sample) can lead to correlation in the position time-series, such that σ_{r_i} increases in the presence of unwanted drifts. In our experience we had $\sigma_{\mathrm{r}_x} = \sigma_{\mathrm{r}_y} = 12$ nm and $\sigma_{\mathrm{r}_z} = 6$ nm.

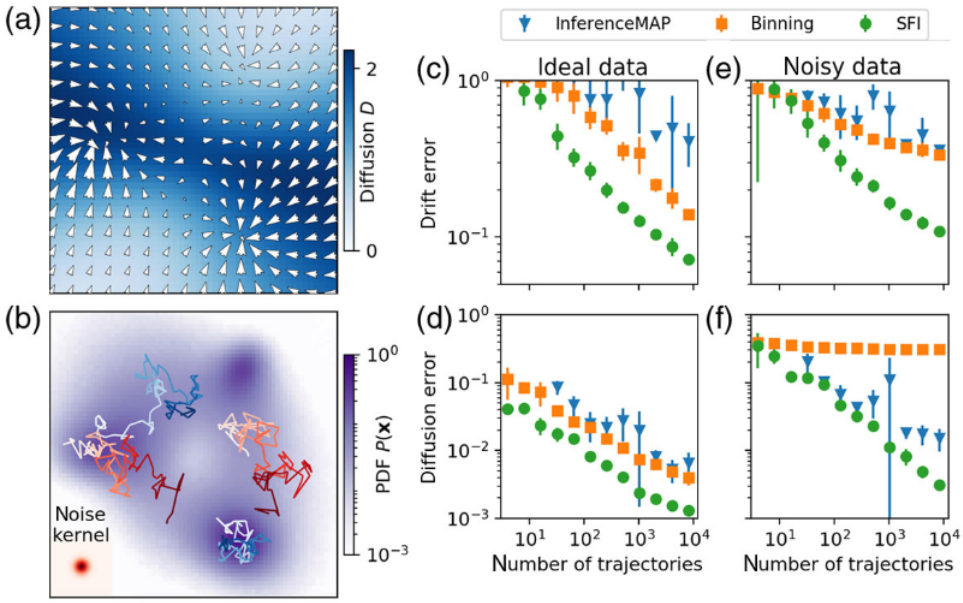


Figure 5.2.6: Figure from [**frishman˙learning˙2020**]. Quantitative comparison of Surface Force Inference (SFI with other methods on a simulated system mimicking 2D single-molecule trajectories in a complex environment with space-dependent isotropic diffusion. a) The diffusion field (blue gradient) and drift field (white arrows). b) The steady-state distribution function (PDF) of the process. The traces are representative trajectories of 100 time steps. c-f) Comparison of the performance of SFI and two widely used inference methods: InferenceMAP, a method for single-molecule inference (blue triangles) [**beheiry˙inferencemap˙2015**], and grid-based binning with maximum-likelihood estimation [**hoze˙heterogeneity˙2012**, **friedrich˙approaching˙2011**] (orange squares). They evaluated the performance of these methods on the approximation of the drift field (c),e)) and diffusion field (d)f)) as a function of the number N of single-molecule trajectories (similar to the ones in panel b)) used. With ideal data (c),d)) and in the presence of measurement noise(e),f)). The performance is evaluated as the average mean-squared error on the reconstructed field along trajectories. More information about the parameters of their simulation and analysis can be found in their work [**frishman˙learning˙2020**].

Although the binning method is well suited for drift measurements, it suffers from a lack of convergence and precision when second moments or local diffusion coefficients have to be extracted. In particular, the binning method did not allow us to measure specifically the local diffusion coefficient in the key interfacial region corresponding to

$z < 100$ nm. Indeed, as we can observe in Fig. 5.2.6-f) the diffusion error on noisy (such as experimental) data does saturate, and the binning method is outperformed by a robust method recently developed by Frishman and Ronceray [**frishman˙learning˙2020**]. This method uses Stochastic Force Inference (SFI), in order to evaluate spatially varying force fields and diffusion coefficients, from the information contained within the trajectories.

In practice the SFI method computes a local diffusion estimator:

$$\hat{d}(t_i) = \frac{[\Delta r_i(t_{i-1}) + \Delta r_i(t_i)]^2}{4\Delta t} + \frac{\Delta r_i(t_i)\Delta r_i(t_{i-1})}{2\Delta t}, \tag{5.2.15}$$

where the second term corresponds to Vestergaard error. Then, by approximating locally the diffusion coefficient by a polynomial function basis (such as $\sum_0^n a_n z^n$). One can use the latter defined estimator in order to fit locally the polynomial coefficient. Once the coefficients estimated, one can compute the diffusion coefficient for any height z in the range of the provided data r_i (*i.e.* in the range $[\min(r_i), \max(r_i)]$. Although, the mathematical details of the SFI method are beyond the scope of the presented work as it requires a great knowledge of information theory, we used the SFI method as a tool. We implemented this method, using a fourth-order polynomial base. To simplify the use of the method with our data, we developed a simple Python function which can infer the local diffusion coefficient by only one function call.

```
1 Dx, Dy, Dz, z_D = Compute_diffusion(pos)
```

where `pos`, is the 3D trajectory of a Brownian colloid. It allowed us to infer the local diffusion coefficients $D_i(z)$, down to $z = 10$ nm, as shown in Fig. 5.2.7. The results are in excellent agreement with the theoretical predictions, $D_{\parallel}(z)$ and $D_z(z)$, using Eqs. (5.1.46) and (5.1.46), thus validating the method.

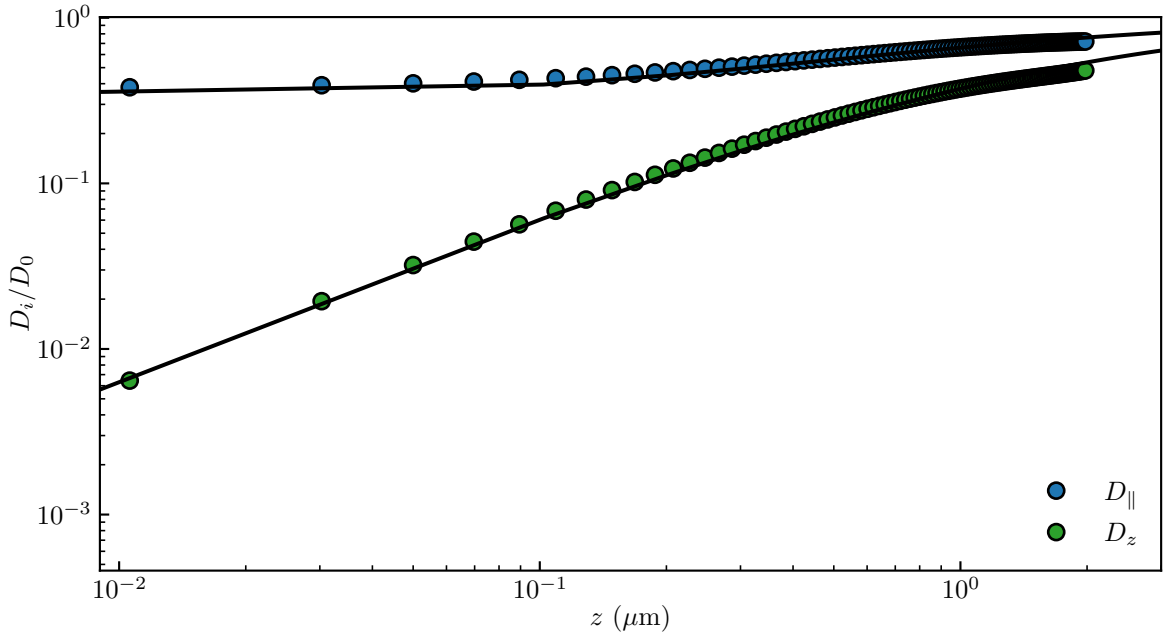


Figure 5.2.7: Measured local short-term diffusion coefficients D_i of the microparticle, normalized by the bulk value D_0, as functions of the distance z to the wall, along both a transverse direction x or y ($D_i = D_\parallel = D_x = D_y$, blue) and the normal direction z ($D_i = D_z$, green) to the wall. The solid lines are the theoretical predictions, $D_\parallel(z) = D_0\eta/\eta_\parallel(z)$ and $D_z(z) = D_0\eta/\eta_z(z)$, using the local effective viscosities $\eta_\perp(z)$ and $\eta_\parallel(z)$ of Eqs. (5.1.46) and (5.1.46), respectively.

5.2.5 Precise potential inference using multi-fitting technique

So far, through Figs. 5.2.2-5.2.7, we have successively presented the various measured statistical quantities of interest, as well as their fits to corresponding theoretical models. Therein, we have essentially three free physical parameters, B, ℓ_B, ℓ_D, describing the particle and its environment, as well as the *a priori* undetermined location of the $z = 0$ origin. These four parameters are actually redundant among the various theoretical models. Therefore, in order to measure them accurately, we in fact perform all the fits simultaneously, using a Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm that is well suited for unconstrained nonlinear optimization [**dai˙convergence˙2002**]. To do so, we construct a global minimizer:

$$\chi^2 = \sum_{n=1}^{N} \chi_n^2 \ , \tag{5.2.16}$$

where we introduce the minimizer χ_n^2 of each set n among the N sets of data, defined as:

$$\chi_n^2 = \sum_{i=1}^{M_n} \frac{[y_{ni} - f_n(x_{ni}, \mathbf{b})]^2}{f_n(x_{ni}\ , \mathbf{b})^2} \ , \tag{5.2.17}$$

with $\{x_{ni}, y_{ni}\}$ the experimental data of set n, M_n the number of experimental data points for set n, f_n the model for set n, and $\mathbf{b} = (b_1, b_2, ..., b_p)$ the p free parameters. In our case, $p = 4$, and $\{x_{ni}, y_{ni}\}$ represent all the experimental data shown in Figs. 5.2.2-5.2.7.

Due to strong dependence of the normal diffusion coefficient D_z with z, it is possible to find the wall position with a 10 nm resolution, thus overcoming a drawback of the Lorenz-Mie technique which only provides the axial distance relative to the focus of the objective lens. Besides, the three physical parameters globally extracted from the multifitting procedure are: $B = 4.8 \pm 0.6$, $\ell_\mathrm{D} = 21 \pm 1$ nm, and $\ell_\mathrm{B} = 530 \pm 2$ nm. Using the particle radius $a = 1.518 \pm 0.006$ μm calibrated from the preliminary fits of the interference patterns to the Lorenz-Mie scattering function (see section 4.7.3), and the $\rho_\mathrm{p} = 1050$ kg.m^{-3} tabulated bulk density of polystyrene, we would have expected $\ell_\mathrm{B} = 559$ nm instead, which corresponds to less than 2 % error, and might be attributed to nanometric offsets, such as *e.g.* the particle and/or wall rugosity.

5.2.6 Measuring external forces using the local drifts

Finally, we investigate the total conservative force $F_z(z)$ acting on the particle along z. The first way to measure it is to calculate the gradient of the potential U which is experimentally measured from the position PDF giving:

$$F_z^\mathrm{eq} = -\nabla U = k_\mathrm{B}T \frac{\mathrm{d}}{\mathrm{d}z} \ln(P_\mathrm{eq}) \ , \tag{5.2.18}$$

where one can use the experimentally measured P_eq (see Fig. 5.2.2). The results of this method are shown in Fig. 5.2.8. However, it can be interesting to measure the forces using the local drifts as for more complex systems, some non-conservative forces could arise. As the Eq. (5.2.18) takes only into account to the potential U, only conservative forces can be extracted from the measurement of P_eq. Non-conservative forces could be measured by the difference between forces obtained through P_eq and the local drifts.

Let us now explain the force measurement from drifts. By averaging the overdamped Langevin of Eq. (5.1.60) over a fine-enough z-binning grid and a short-enough time interval Δt, one gets in the Itô convention (corresponding to our definition of Δz):

$$F_z(z) = 6\pi\eta_z(z)a\frac{\langle \Delta z \rangle}{\Delta t} - k_\mathrm{B}T\frac{D_z'(z)}{D_z(z)} \ , \tag{5.2.19}$$

where the last term corresponds to the additional contribution due to the non-trivial inte-

gration of the multiplicative noise [**volpe˙influence˙2010**, **mannella˙comment˙2011**, **mannella˙ito˙2012**, **sancho˙brownian˙2011**] (see section 5.1.6), with the prime denoting the derivative with respect to z. From the averaged measured vertical drifts $\langle \Delta z \rangle$, and invoking Eqs. (5.1.45) and (5.1.48), one can reconstruct $F_z(z)$ from Eq. (5.2.19), as shown in Fig. 5.2.8. We stress that the statistical error on the force measurement is comparable to the thermal-noise limit [**liu˙subfemtonewton˙2016**]:

$$\Delta F = \sqrt{24\pi k_{\mathrm{B}} T \eta_z(z) a / \tau_{\mathrm{box}}(z)} \ , \tag{5.2.20}$$

where $\tau_{\mathrm{box}}(z)$ is the total time spent by the particle in the corresponding box of the z-binning grid. To corroborate these measurements, we invoke Eq. (5.1.28) and express the total conservative force $F_z(z) = -U'(z)$ acting on the particle along z:

$$F_z(z) = k_{\mathrm{B}} T \left(\frac{B}{\ell_{\mathrm{D}}} \mathrm{e}^{-\frac{z}{\ell_{\mathrm{D}}}} - \frac{1}{\ell_{\mathrm{B}}} \right) \ . \tag{5.2.21}$$

Using the physical parameters extracted from the above multifitting procedure, we plot Eq. (5.2.21) in Fig. 5.2.8. The agreement with the data is excellent, thus showing the robustness of the force measurement. In particular, we can measure forces down to a distance of 40 nm from the surface. Besides, far from the wall, we are able to resolve the actual buoyant weight $F_{\mathrm{g}} = -7 \pm 4$ fN of the particle. This demonstrates that we reach the femtoNewton resolution, and that this resolution is solely limited by thermal noise.

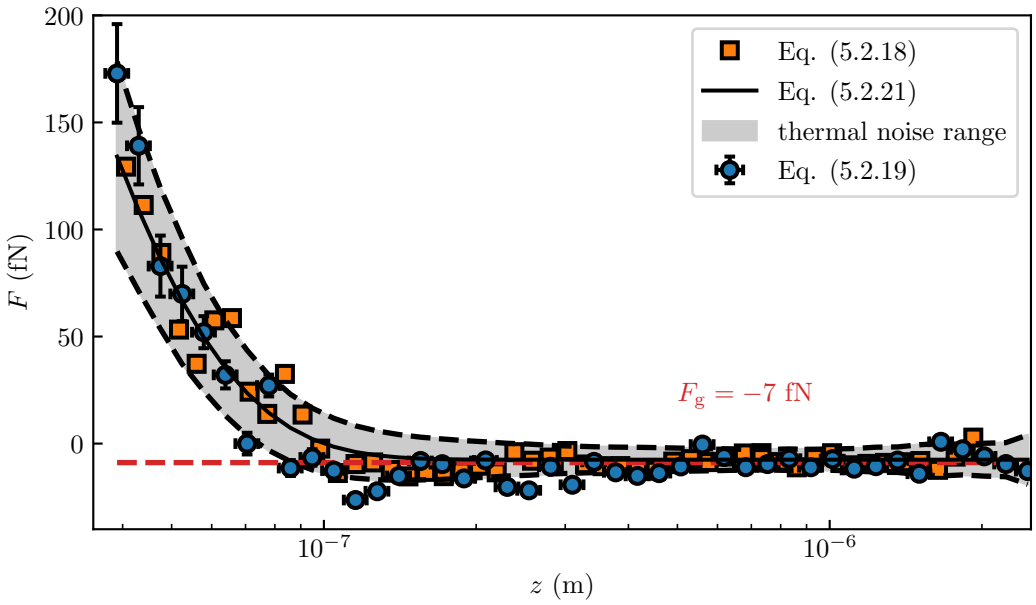


Figure 5.2.8: Total normal conservative force F_z exerted on the particle as a function of the distance z to the wall, reconstructed from Eq. (5.2.19), using Eq. (5.1.48) for the circles and Eq. (5.2.18) for the squares. The solid line corresponds to Eq. (5.2.21), with $B = 4.8$, $\ell_D = 21\,\text{nm}$ and $\ell_B = 530\,\text{nm}$. The black dashed lines and gray area indicate the amplitude of the thermal noise computed from Eq. (5.2.20). The horizontal red dashed line indicates the buoyant weight $F_g = -7$ fN of the particle.

5.3 Conclusion

In this section we have covered the physics we need to take into account when considering confined Brownian motion. We first detailed the gravitational and DLVO interactions to detail the Gibbs-Boltzmann distribution. Then, we detail the space-varying damping due to the hydrodynamic interactions between the wall and the particle. The damping which is now space-varying in the Langevin induces a non-trivial integration of the Langevin force, which induces the appearance of a spurious drift term in the Fokker-Planck equation. We then present a method that permits to estimate how the simulation time-step should be selected, then we show that the spurious drift term is needed to be taken into account to recover the correct equilibrium distribution.

In a second part, we present a multi-scale statistical analysis for the problem of freely diffusing individual colloids near a rigid wall. Combining the equilibrium distribution in position, time-dependent non-Gaussian statistics for the spatial displacements, a novel method to infer local diffusion coefficients, and a multifitting procedure, allowed us to reduce drastically the measurement uncertainties and reach the nanoscale and thermal-

noise-limited femtoNewton spatial and force resolutions, respectively.

6 Other applications of the method

In this chapter, I present ongoing work and preliminary observations about other topics, studied with the method presented previously.

6.1 Elastohydrodynamic lift near a soft wall

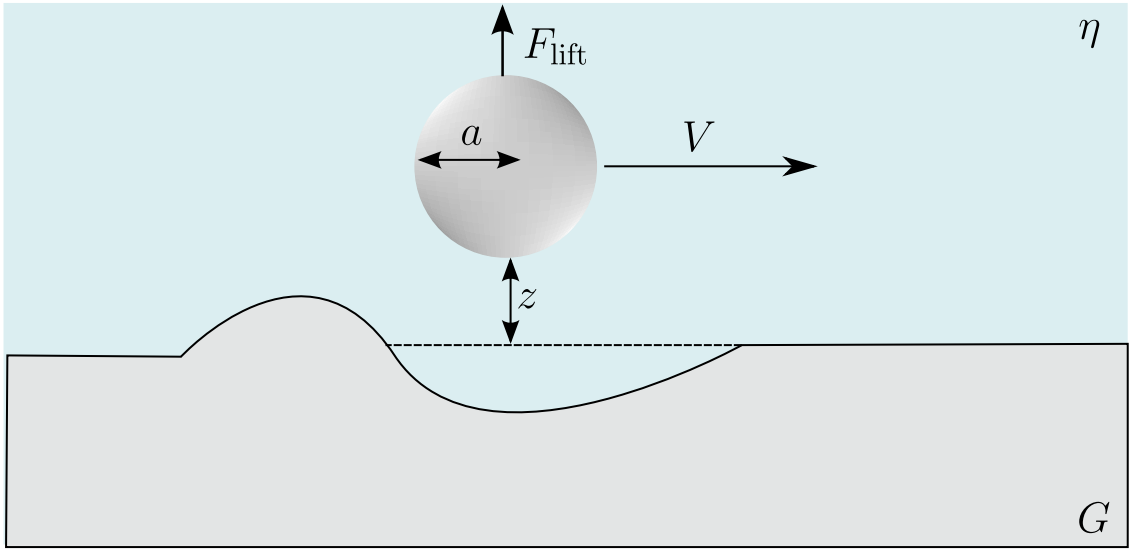


Figure 6.1.1: Schematic of a spherical colloid of radius a immersed in a fluid of viscosity η sliding at a velocity V above an incompressible and linear-elastic substrate of shear elastic modulus G. From the elastohydrodynamic interaction between the particle and the soft wall arises a net lift force F_{lift} (see Eq. (6.1.1)).

Elastohydrodynamics (EHD) is the field of mechanics that couples elasticity and hydrodynamics. We here focus particularly on EHD in the context of lubrication, which describes the relative motion of immersed objects in a near-contact regime. Lubrication EHD is present at many length and time scale. Examples of relevant phenomena and systems include, kilometrics landslides [**campbell˙self-lubrication˙1989**], roller bearings [**hamrock˙fundamentals˙2004**] and blood-cell motion in microfluidic devices [**byun˙characterizing˙2013**, **higgins˙sickle˙2007**, **cohen˙hydrodynamics˙2013**]. Recently, the problem of a free particle that can sediment, slide or roll near a soft surface has been treated [**sekimoto˙mechanism˙1993**, **skotheim˙soft˙2004**, **skotheim˙soft˙2005**, **beaucourt˙optimal˙2004**, **salez˙elastohydrodynamics˙2015**, **bertin˙soft-lubrication˙2021**]. As the particle slides near the surface, the hydrodynamic stresses deform the soft wall surface. This deformation induces a symmetry breaking of the contact geometry. Hence, a net normal force emerges and is applied to the objects. Furthermore, if a particle is

sliding due to its own weight, this lift force can be self-sustained. The first experimental quantitative observation of the lift effect has been done at the macroscopic scale using negatively-buoyant centimetric cylinders immersed in a viscous fluid, that were sliding down a tilted wall coated with an elastic layer. The authors show that the self-sustained EHD lift reduces the coefficient of dynamic friction by nearly an order of magnitude and suggest that this EHD force could partially explain phenomena such as reduced wear in animal joints and long-runout landslides. The EHD lift force has also recently been measured at the microscopic-scale, using micron-sized colloidal spheres in micro-channels under flow. However, in all these experiments, the motion is deterministic and induced by an external force or flow. In the context of my thesis, we further wonder whether spontaneous Brownian motion, and thus thermal fluctuations could trigger such an effect in some temporal range. In the lubrication "EHD" theory, considering a sphere of radius a moving at constant velocity V, in a solvent of viscosity η, and at a distance z from a thick (with respect to the particle radius), incompressible, linear-elastic substrate of shears elastic modulus G, the lift force F_{lift} reads [**skotheim˙soft˙2004**]:

$$F_{\text{lift}} \sim 0.41 \frac{\eta^2 V^2}{G} \frac{a^{5/2}}{z^{5/2}} \ , \tag{6.1.1}$$

where the 0.41 factor has been computed by Bertin *et al.* [**bertin˙soft-lubrication˙2021**] as well as other force and torque components. To incorporate fluctuations into this deterministic picture, a simple idea is to replace the velocity V in Eq. (6.1.1) by the typical horizontal thermal velocity $\sqrt{k_{\text{B}}T/m}$ obtained through the Maxwell-Boltzmann distribution , leading the following estimate of an hypothetical Brownian EHD lift force:

$$F_{\text{lift,Brown}} \sim 0.41 \frac{\eta^2 k_{\text{B}} T}{G \rho_{\text{p}} a^{1/2} z^{5/2}} \ . \tag{6.1.2}$$

From this equation, we can observe a counterintuitive effect: as the particle radius (and thus the surface of contact) decreases, the the EHD force $F_{\text{lift,Brown}}$ increases. Taking typical biophysical values such as $G \simeq 10$ kPa, $\rho_{\text{p}} = 1350$ kg.m^{-3} (proteins density) and $a = 100$ nm, we see that the Brownian EHD force is in the piconewton range. The latter range is comparable to other surface forces, which means that microscopic entities in biology and nanoscience may spontaneously trigger Brownian EHD couplings, notably their dynamics. However, it is important to note that it is only a simple estimate and which contains a high risk of conceptual failure associated, for example, to the lake of compensating drift at equilibrium.

6.1.1 PolyDimethylSiloxane

To do the soft coating experimentally, we use PolyDimethylSiloxane (PDMS) which is widly use for fabrication in microfluidics and also in shampoo [**im˙shampoo˙2012**] or food[10]. PDMS is a silicone-based organic polymer which chemical formula is:

$$CH_3[Si(CH_3)_2O]_nSi(CH3)_3 \ , \tag{6.1.3}$$

where n is the number of $Si(CH_3)_2O$ dimethyl groups. By mixing a solution of PDMS chains with a curring agent containing hydrosilane groups (SiH), bonds or crosslink between different PDMS chains are appearing, as shown in Fig. 6.1.2.

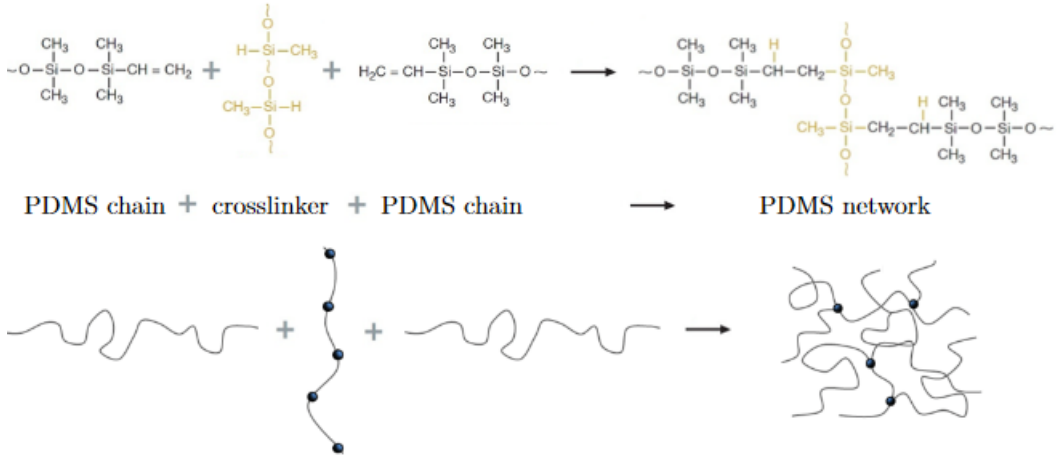


Figure 6.1.2: Figure taken from [**tucher˙analysis˙2016**]. Examples of crosslinking reaction between the PDMS chains and a curing agent containing hydrosilane groups.

Due to the crosslinker, the PDMS turns into an elastomer, modelled as an incompressible and linear-elastic solid. Some of its characteristics are to be hydrophobic and to exhibit strong gas permeability [**xia˙soft˙1998**]. The elastic modulus G of the crosslinked PDMS can be tuned by changing the mixing ratio of base polymer solutions and curing agent. For example, for one of the most used PDMS, which is Sylgard 184, a mixing ratio of 10 : 1 leads to an elastic modulus $G = 1.5$ MPa, and 35 : 1 leads to $G \simeq 100$ kPa [**wang˙crosslinking˙2014**]. To prepare experimental samples, it possible to spin coat the microscope slides with the base-agent mixture before it is cured in order to have a thick soft surface coating onto the slides. Moreover, after curing some uncrosslinked chains (or free chains) usually remains in samples which can play important role when working near

[10] PDMS is used as an antifoaming agent in food and is identified by the European food additive number E900.

the soft surfaces. Indeed, Hourlier-Fargette *et al.* [**hourlier-fargette˙role˙2017**] shown that these free chains are drawn by water drops, altering there sliding dynamics. To avoid any issue, Glover *et al.* [**glover˙extracting˙2020**] proposed an inexpensive method to clean PDMS. There method consists in putting a PDMS sheet at a water - organic solvent interface and wait that free chains migrate into the solvent. However, for simplicity, we first decided to use already prepared samples sold by Ibidi, these came as soft coated Petri dishes with a coverslip on the bottom that we can directly fit into our microscope.

6.1.2 Measuring non-conservative forces

To measure the non-conservative forces felt by a Brownian particle diffusing on top of a soft surface, we do the exact same experiment and data analysis as the one developed in section 5.2. As the EHD force does not derive from a potential, we need to extract the non-conservative forces F_{NC}, to do so, by combining Eqs. (5.2.18) and (5.2.21), the non-conservative force reads:

$$F_{\mathrm{NC}} = F_z(z) - F_z^{\mathrm{eq}}(z) \ . \tag{6.1.4}$$

In Fig. 6.1.3 are shown the measured F_{NC} as a function of particle-wall distance, for two different elastic moduli $G = 15$ and 28 kPa. These first experiments suggest that Brownian motion might indeed trigger some non-conservative EHD forces. This preliminary experiment will be completed in near future with additional and systematic experiments.

Specifically, we will vary three parameters, η, G and a to check if we can obtain a universal plot,*i.e.* where $F_{\mathrm{lift,Brown}}$ varies linearly with $\frac{\eta^2}{Ga^{1/2}}$, as shown in Fig. 6.1.4 with the first experiments.

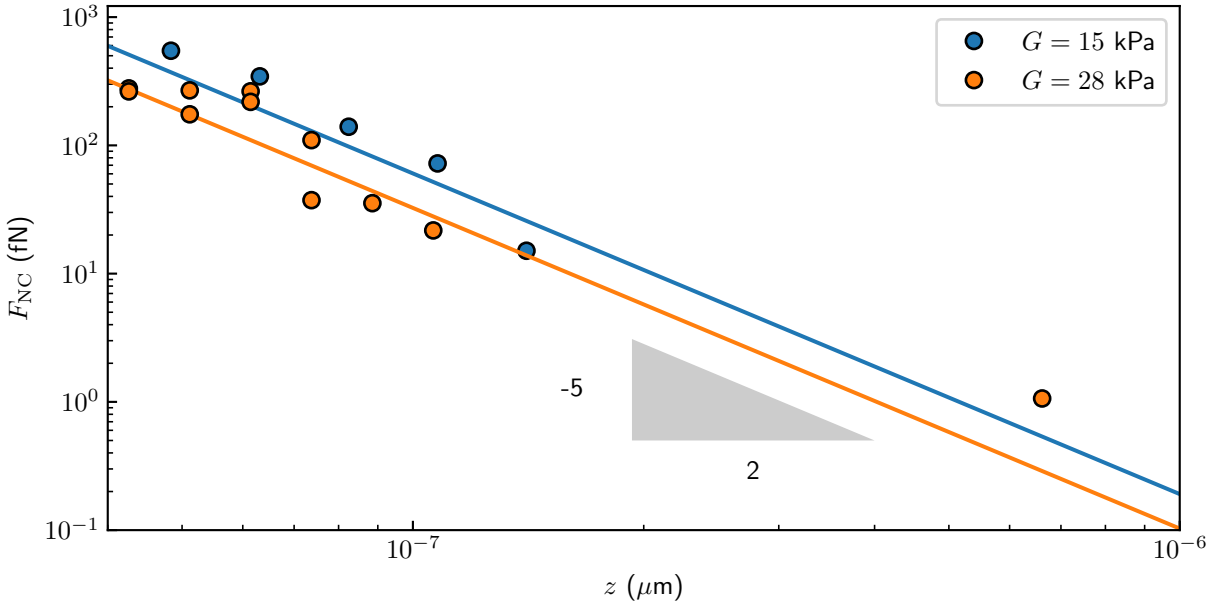


Figure 6.1.3: Non-conservative forces measured experimentally as a function of particle-wall distance, for colloidal particles of radius $a = 1.5$ μm diffusing in water above an incompressible and linear-elastic substrate of shear elastic moduli $G = 15$ and 28 kPa. Plain lines correspond to the Brownian EHD prediction $F_{\mathrm{lift,Brown}}$ (see Eq. (6.1.2)).

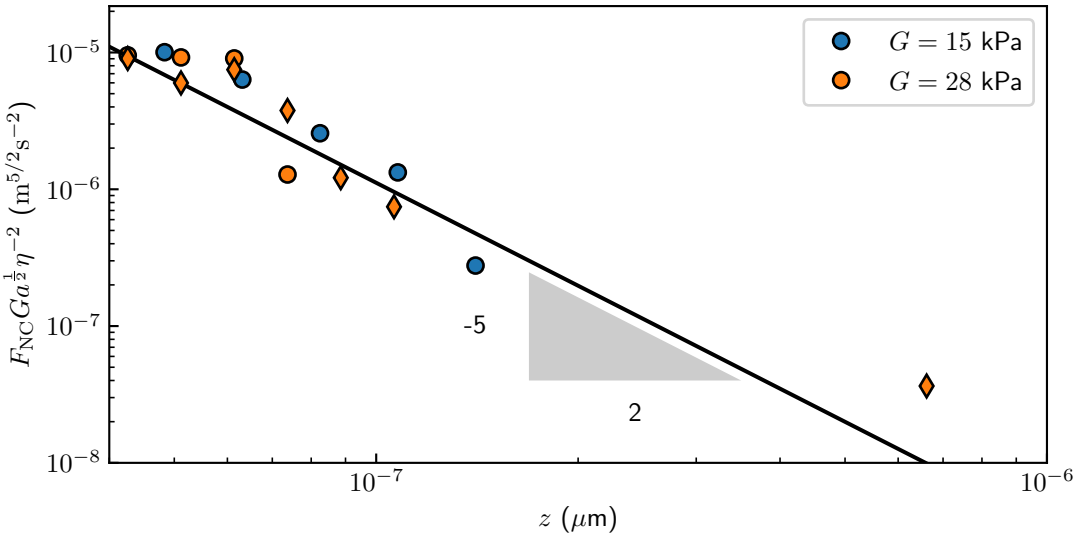


Figure 6.1.4: Non-conservative forces normalized by $Ga^{1/2}z\eta^{-2}$ by measured experimentally for colloidal particles of radius $a = 1.5$ μm diffusing in water above an incompressible and linear-elastic substrate of shear elastic moduli $G = 15$ and 28 kPa. The plain line corresponds to the Brownian EHD lift force $F_{\mathrm{lift,Brown}}$ (see Eq. (6.1.2)).

6.2 Close wall stuck motion

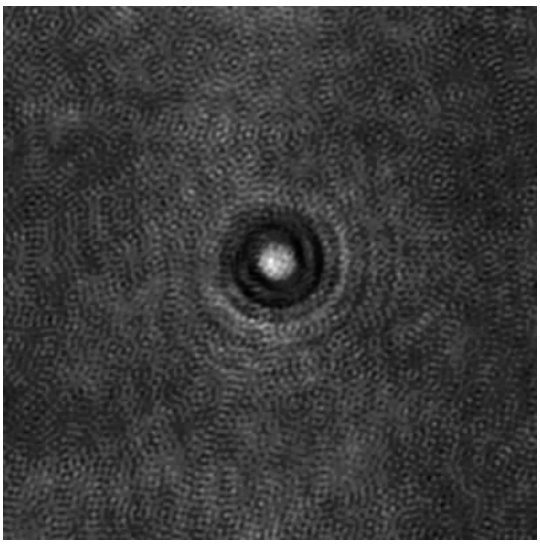


Figure 6.2.1: Median of the a movie of the hologram of a stuck yet moving particle. The median is calculated over 120 images taken every 30 s.

When performing the experiments near a rigid wall in order to measure the Debye length ℓ_D (see section 5.2.1) as a function of the concentration of NaCl, we observed that some particles were stuck on the surface. As we first expected for this stuck particles, we did not observe any motion of the particle. However, surprisingly, in some cases we observed that some particles were slightly diffusing. This slight diffusion can be observed directly from the raw data, by looking at the median of the images of the video capture from an hologram of a stuck yet moving particle. The latter median is shown in Fig. 6.2.1, where we observe a blurry hologram due to the particle motion. Moreover, as we cannot properly have the background in this experiment since the particle does not diffuse enough, the statistical error is increased. The measured trajectory is shown in Fig.6.2.2, where we observe that a mechanical drift happened during the experiment. This could be due to a shifting in time of the sample or the objective position, for example. The corresponding drift velocity is on the order of 2 μm.h^{-1} along the x- and y-axes and 6 μm.h^{-1} along the z-axis. In the following, we look at the short-time dynamics ($t < 1$ s), since the drift over such a time scale is of the ordert the nm.s$^-$1.

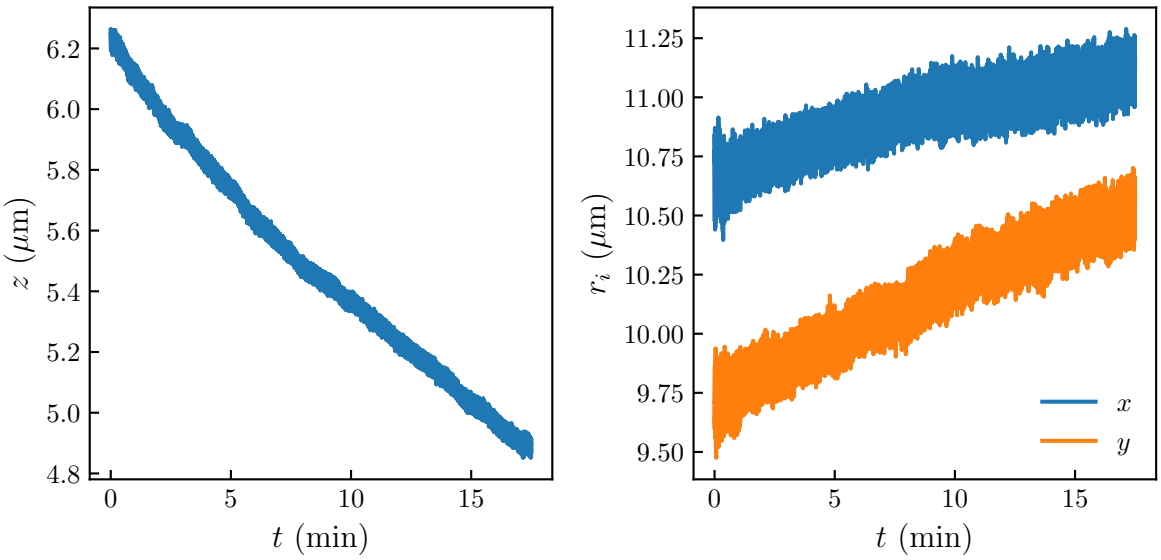


Figure 6.2.2: Raw trajectories measured using the Mie tracking technique for the x-, y- and z- axes, for a particle of radius $a = 1.5$ μm. The time between each frame is $\tau = 1/200$ s.

Let us focus on the MSDs along the x-, y- and z-axes which are shown in Fig. 6.2.3. We observe that the MSD along the z- axis is a constant, with $(\langle r_z^2 \rangle_t)^{1/2} = 10$ nm which is of the order of the tracking uncertainty (see section 4.3); hence, I will not physically comment the results obtained along the z-axis. However, on the x-axis, interestingly, we can observe two regimes, indicating that the particle is diffusing in a potential. In addition, the collapse with the corresponding data on the y-axis suggests that the potential is isotropic along the x- and y-axis, as if the particle was adhering to the surface with some rotational diffusion. The regime, at short time is linear with Δt showing a normal diffusion with average diffusion coefficients (see Eq. (5.2.2)) $\langle D_{\parallel} \rangle = \langle D_x \rangle = \langle D_y \rangle = 0.14\, D_0$. This latter value is lower than the minimum value of $D_{\parallel}$, See Eq. (5.1.47), which demonstrates that the associated motion is not a simple translational diffusive motion in the xy-plan.

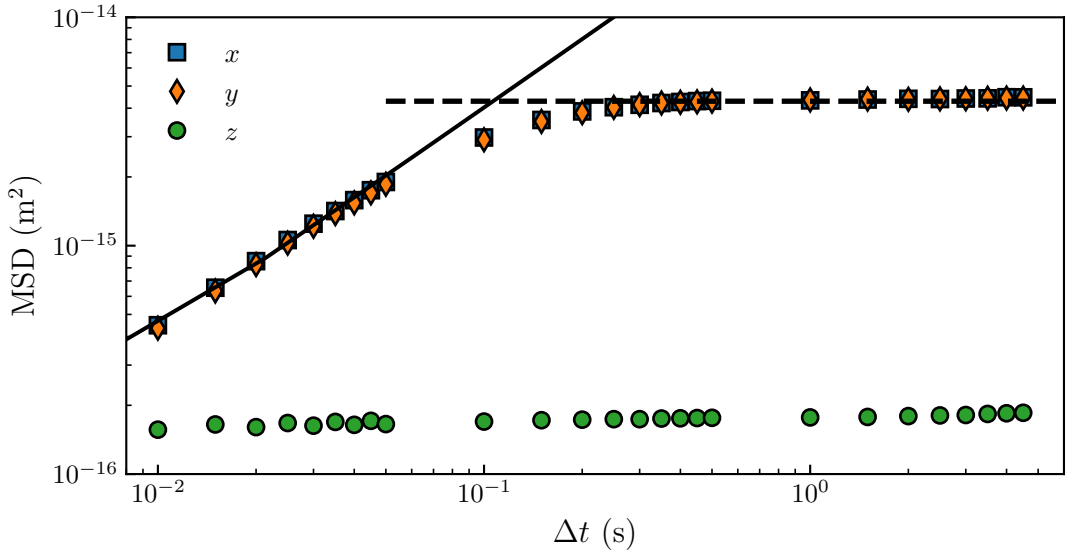


Figure 6.2.3: Measured mean-squared displacements (MSD, see Eq. (5.2.1)) of a particle stuck on the surface as functions of the time increment Δt, for the three spatial directions, x, y, and z. The solid line is the best fit to Eq. (5.2.2), having $\langle D_i \rangle$ as a free parameter, providing the average diffusion coefficient $\langle D_\parallel \rangle = \langle D_x \rangle = \langle D_y \rangle = 0.14\, D_0$. The dashed black line is the average value of the plateau of the MSD along the x- and y-axes, *i.e.* 4.3×10^{-15} m^2.

The plateau of the MSD along x and y has an average value 4.3×10^{-15} m^2. By supposing that the particle is in a harmonic potential, we can make an estimate of the spring constant k_H, using the relation:

$$k_\mathrm{H} = \frac{2k_\mathrm{B}T}{\lim\limits_{\Delta t \to \infty} \langle \Delta x^2 \rangle} = \frac{8 \times 10^{-21}}{4.3 \times 10^{-15}} \simeq 2\ \mu\mathrm{N.m}^{-1}\ . \tag{6.2.1}$$

Beyond further understanding the origin of such a stiffness, and possibly relating it to adhesion, it would be interesting to reproduce the same study using a soft surface and see if we can observe a change of k_H with the elastic modulus G. If the latter is true, this experiment could lead to a local determination of elastic moduli using Brownian colloids attached on soft surfaces. Additionally, we can look at the displacement PDFs P_i as shown in the Fig. 6.2.4. Contrary to the result we had for a freely diffusing particle (see Fig. 5.2.5) we do not observe a clear non-Gaussianity here.

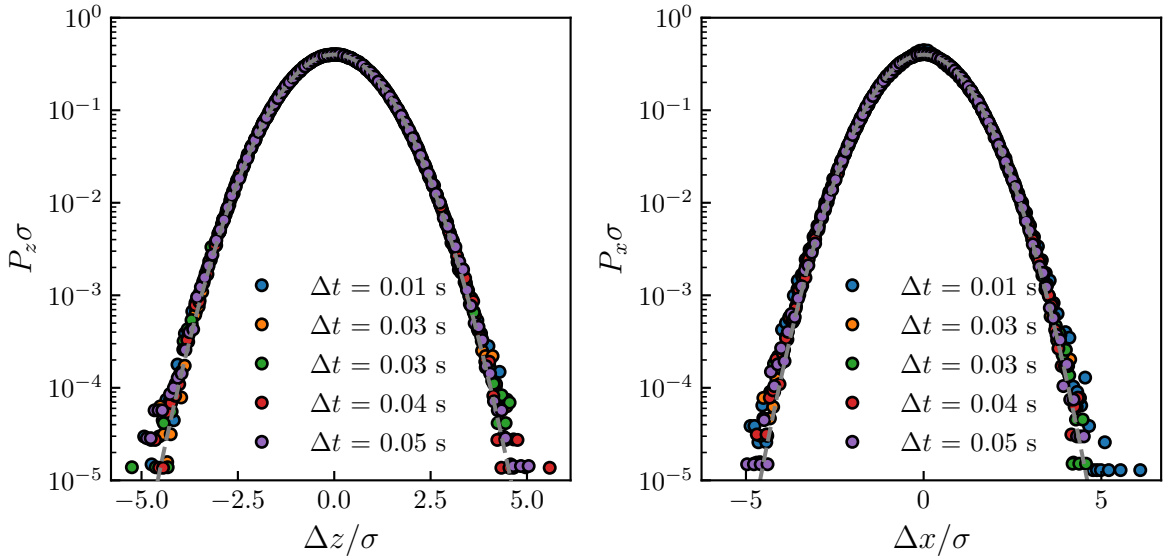


Figure 6.2.4: Normalized probability density functions $P_i\,\sigma$ of the normalized displacements $\Delta x/\sigma$ and $\Delta z/\sigma$, at short times, with σ^2 the corresponding MSD (see Fig. 6.2.3), for different time increments Δt ranging from 0.01 s to 0.05 s, as indicated with different colors. The gray dashed lines are normalized Gaussian distributions, with zero means and unit variances.

6.3 Long-time 4th cumulent

In this subsection we consider a cumulent of higher order than 2 (MSD). The 4th order cumulent readss:

$$C_4(\Delta t) = \frac{1}{4!}\left(\left\langle [x_i(t+\Delta t) - x_i(t)]^4 \right\rangle_t - 3\left\langle [x_i(t+\Delta t) - x_i(t)]^2 \right\rangle_t^2\right) \ . \tag{6.3.1}$$

For a Gaussian-distributed variable x , C_4 is given by the second order moment as $\langle \Delta x^4 \rangle_t = \langle \Delta x^2 \rangle_t^2$. Therefore, the literature often overlooks higher moments as they rarely give additional information. However, as it has been shown along this manuscript, confined Brownian motion exhibits non-Gaussian statistical properties, as shown in Fig. 5.2.5. Thus, it becomes interesting to study higher-order cumulents. We work on this project with David Dean, Thomas Guérin and Arthur Alexandre who are experts in stochastic theory. They found an interesting behavior for the 4th cumulent in parallel displacement to the wall, for which the Langevin equation along the x-axis is given by:

$$\mathrm{d}x = \sqrt{2D_{\parallel}(z)}\mathrm{d}B_t \tag{6.3.2}$$

In this context, Eq. (6.3.1) can be simplified in two regimes [unpublished work by Alexandre *et al.*], for $\Delta t \ll 1$, one has:

$$\lim_{\Delta t \to 0} C_4(\Delta t) = \frac{\Delta t^2}{2} \left(\langle D_\parallel^2 \rangle - \langle D_\parallel \rangle^2 \right) \ , \tag{6.3.3}$$

where the averages are done over the equilibrium distribution P_{eq}. The forth-cumulent thus varies as Δt^2 at short time. In the limit $\Delta t \to \infty$, Eq. (6.3.1) can be simplified to:

$$\lim_{\Delta t \to \infty} C_4(\Delta t) = C_4^0 \Delta t - C_4^1 \ , \tag{6.3.4}$$

where C_4^0 and C_4^1 are constants that depend on the equilibrium distribution P_{eq}, and can be written as functions of B, ℓ_{D} and ℓ_{B}. Unlike the first and second moments of the parallel displacement (the average displacement $\langle \Delta x \rangle_t$ and the MSD $\langle \Delta x^2 \rangle_t$ respectively), the 4th cumulent exhibits a peculiar regime at late times. This result is thus interesting as it gives a new statistical observable to characterize the long-term trajectory. If this prediction is verified, it could be added to our inference method (see section 5.2) in order to gain precision and robustness.

As we are only interested in the x- and y-axis displacements to show this result, we do not need 3D tracking. Thus, we only need to find the hologram centroids, and therefore we can use faster algorithms than the one employed in the full Mie framework. To do so, we use the Trackpy package [**allan˙soft-mattertrackpy˙2021**] which implements the Crocker–Grier algorithm [**crocker˙methods˙1996**]. The latter permits tracking the centroids of hundreds of colloids rapidly. Trajectories recorded this way are shown in Fig. 6.3.1.

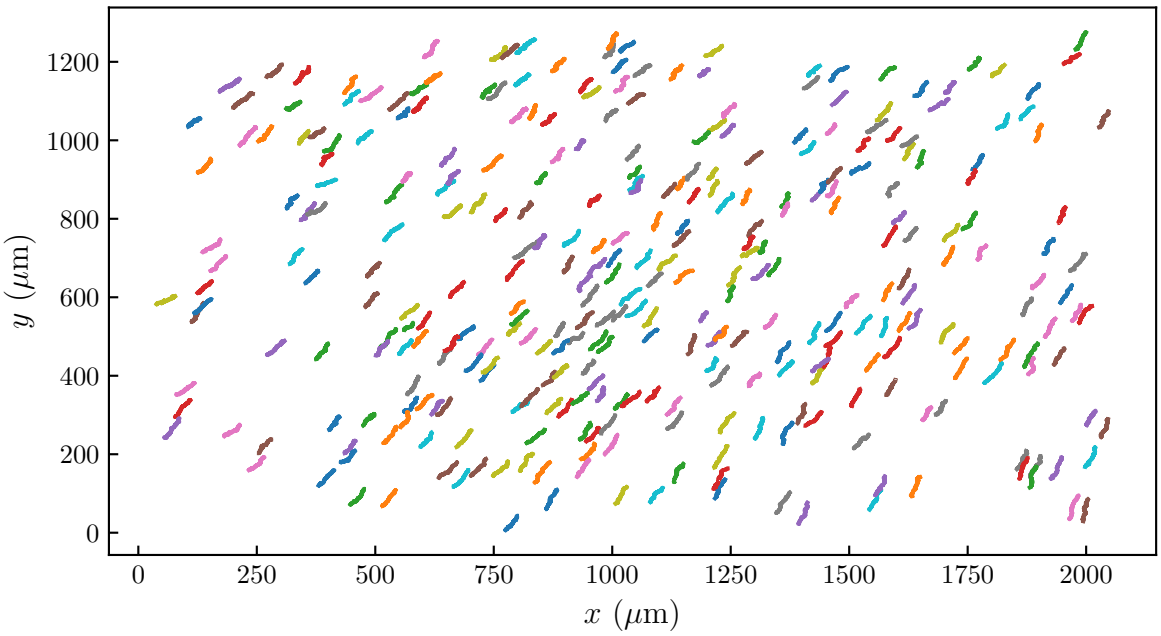


Figure 6.3.1: Trajectories of 300 particles of commercial radius $a = 2.5$ μm. The trajectories are composed of 10000 points with each, with time step $\tau = 0.05$ s.

From the data, we can compute the 4th cumulent using Eq. (6.3.1) and we do observe a linear regime as shown in Fig. 6.3.2. However, this experiment has only been done once so far: it is thus a preliminary result that needs further confirmation.

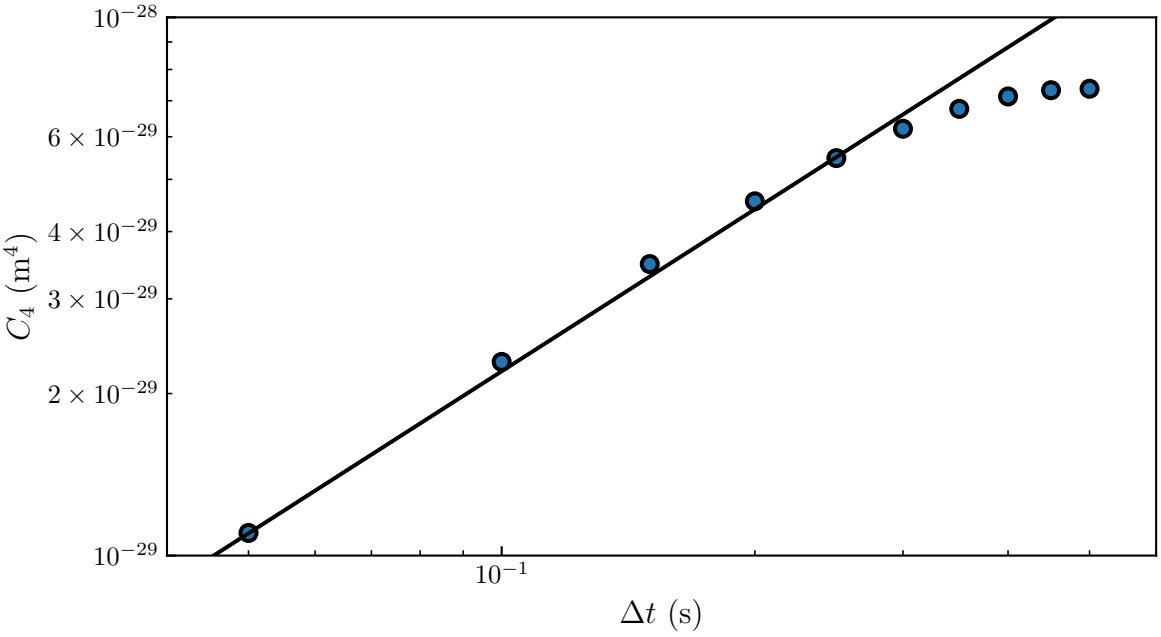


Figure 6.3.2: Experimentally measured 4th cumulent of the transverse displacement as a function of time increment. The plain line corresponds to a linear relation, as in Eq. (6.3.2)

6.4 Sample ageing

While doing the experiments on the soft surfaces, we observed that the images were becoming blurry with time. An example is shown in Fig. 6.4.1 where we can see microscope images separated by three hours.

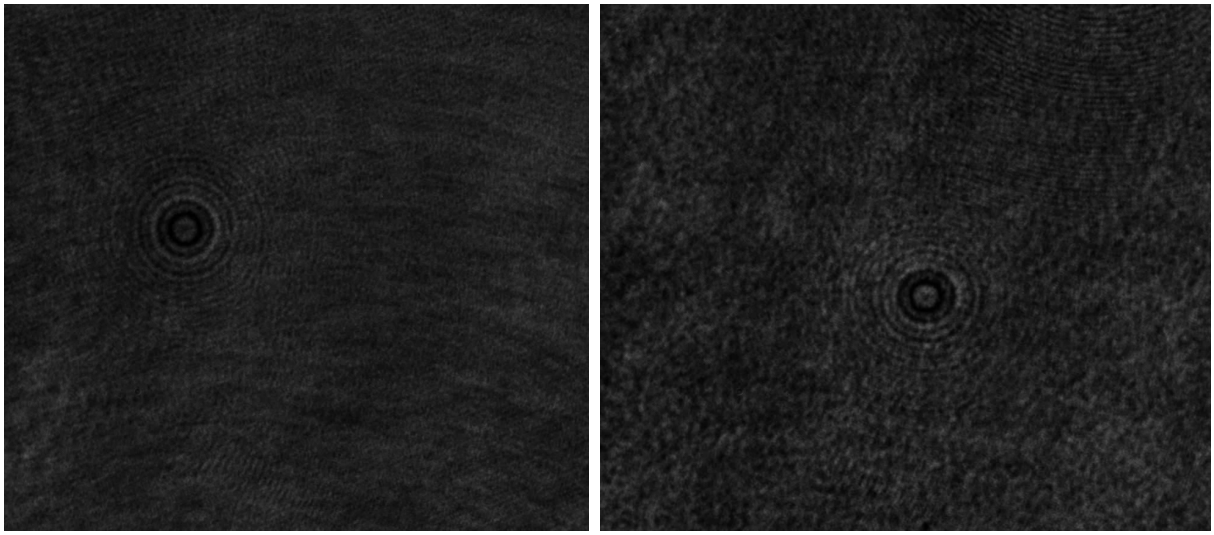


Figure 6.4.1: Hologram of a particle diffusing above a soft surface ($G = 1.5$ kPa). The image on the right has been taken 3 hours after the one on the left. The images are 45 μm wide and 50 μm tall.

Trying to find the origin of this effect, we focused on the interface between the soft PDMS layer and the glass substrate. Doing so, we observed a structure that looks like bubbles, as shown in Fig.6.4.2. We observe that this effect happens faster as the PDMS is softer (lower modulus) and the bubbles seem to be also bigger. One of the ideas we have to explain this phenomenon comes from the high gas permeability of PDMS that could lead to a gas accumulation between the PDMS and the glass. The origin of this gas would come from the naturally present gas molecules inside the colloid suspension. Other possible scenarios include swelling and osmocapillarity in the PDMS. This observation will be the subject of future work.

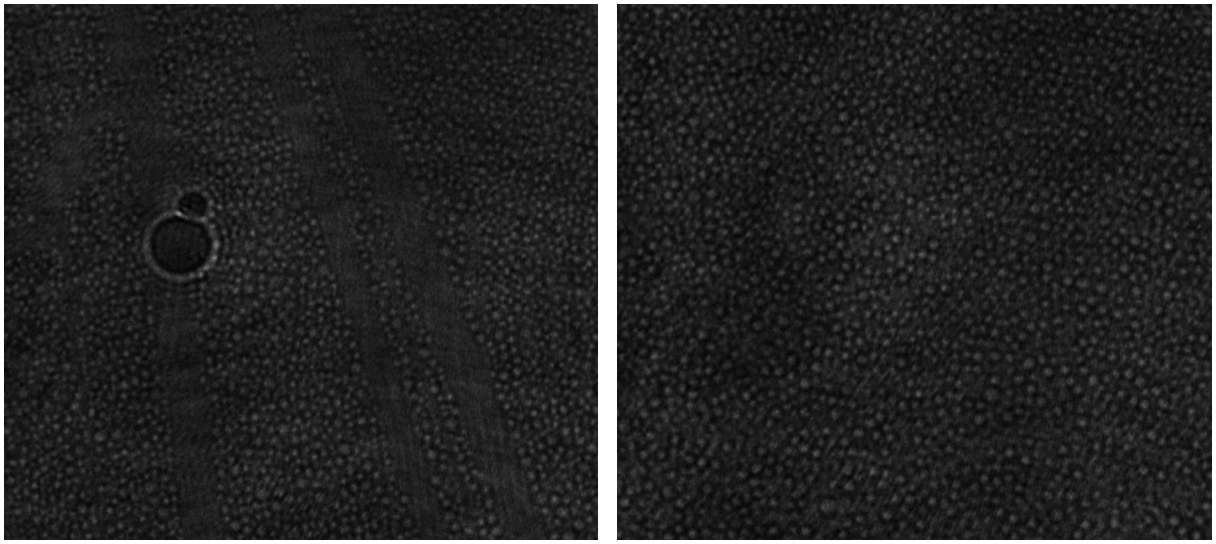


Figure 6.4.2: Images of the glass-PDMS ($G = 1.5$ kPa) interface, three hours after water has been introduced atop the sample. The images are 45 μm wide and 50 μm tall.

7 Conclusion

In this manuscript, we have addressed experimentally, numerically and theoretically several aspects related to Brownian motion. In chapter 3, we started with history by presenting the first observation in the XIXth century by Robert Brown and the subsequent mathematical description, and numerical simulation. We have then, in chapter 4, reviewed different techniques that permit tracking of individual microparticles. We mainly focused on the Lorenz-Mie framework that we used in order to study the free confined Brownian motion. Indeed, it requires no calibration by offering a direct measure of the radius and optical index of each tracked particle. We then showed the experimental setup, which is a custom-made inverted microscope that we optized over the years to capture Lorenz-Mie holograms. To retrieve the trajectory from the captured holograms we mainly used a tool developped at the Grier's Lab under the name of Pylorenzmie and a custom version, Wraplorenzmie, to automate the tracking across whole movies and simplify the use of MP4 files ..

The particles thus tracked and the trajectories retrieved, in chapter 5, we focused on the analysis of Brownian motion near a rigid and charged wall. We first detailed in section 5.1 how the physics changes due to the wall, from the DLVO interactions between the surface and the particles to hindered diffusion. Due the latter, a spurious drift appears in the overdamped Fokker-Planck equation which needs to be taken into account in numerical simulations, and also for force measurements.

Once the underlying theory has been explained,in section 5.2, we described the experimental results and how the trajectories were analyzed. We addressed static observables such as equilibrium distribution, and dynamic observables such as the Mean Squared Displacements (MSDs) and displacement distributions. In particular, for the motion perpendicular to the wall, the MSD exhibits a normal distribution at short time, and a plateau equilibrium value at long time. Furthermore, the short-time displacement distribution exhibits non-Gaussian properties which is a direct signature of the hindered mobility induced by the rigid boundary.

The statistical properties of a confined colloid understood and correctly measured, we then focused on measuring the hindered diffusion coefficient. This measure was done using a novel method developped by Frishman and Ronceray [**frishman˙learning˙2020**], using information theory to infere the local mobility onto a basis of fit functions, leading to high accuracy near the surface (where there is less data).

All the data, from the equilibrium distribution to the hindered mobility can be described

by the three parameters of the system B, ℓ_{D} and ℓ_{B}. We thus built a multi-fitting method, that permits to infere precisely these parameters by taking into account all the observables at once. From this method, we extracted precisely the equilibrium potential from which we computed the conservative forces. This force measuremement was successfully corroborated by an independent one based on the drifts of the trajectories.

All together, we are able to reach a thermal-noise-limited femtonewton force resolution as well as a spatial resolution at the nanoscale.

As shown in chapter 6, we are currently using our device and method in order to investigate and original coupling between soft lubrication theory and Brownian motion. Also, we are working on pushing the limits of our resolution to measure fine effects on higher-order cumulents. Finally, we believe that the ability to measure tiny surface forces, locally, and at equilibrium, as well as the possible application of the method towards non-conservative forces and out-of-equilibrium settings, open fascinating perspectives for nanophysics and biophysics.

8 Conclusion en français

Dans ce manuscrit, nous avons abordé expérimentalement, numériquement et théoriquement plusieurs aspects liés au mouvement Brownien. Dans le chapitre 3, nous avons commencé par l'histoire, en présentant la première observation au XIXème siècle par Robert Brown et la description mathématique ainsi que la simulation numérique qui ont suivi. Nous avons ensuite, dans le chapitre 4, passé en revue les différentes techniques qui permettent de suivre des microparticules individuelles. Nous nous sommes principalement concentrés sur le cadre Lorenz-Mie que nous avons utilisé afin d'étudier le mouvement brownien libre confiné. En effet, il ne nécessite aucune calibration en offrant une mesure directe du rayon et de l'indice optique de chaque particule suivie. Nous avons ensuite montré le dispositif expérimental, qui est un microscope inversé fait sur mesure que nous avons optimisé au fil des ans pour capturer des hologrammes de Lorenz-Mie. Pour extraire la trajectoire des hologrammes capturés, nous avons principalement utilisé un outil développé au laboratoire de Grier sous le nom de Pylorenzmie et une version personnalisée, Wraplorenzmie, pour automatiser le suivi sur des films entiers et simplifier l'utilisation des fichiers MP4 .

Les particules ainsi suivies et les trajectoires extraites, dans le chapitre 5, nous nous sommes intéressés à l'analyse du mouvement brownien à proximité d'une paroi rigide et chargée. Nous avons d'abord détaillé dans la section 5.1 comment la physique change à cause du mur, des interactions DLVO entre la surface et les particules à la diffusion entravée. En raison de cette dernière, une dérive parasite apparaît dans l'équation de Fokker-Planck suramortie qui doit être prise en compte dans les simulations numériques, ainsi que pour les mesures de force.

Une fois la théorie sous-jacente expliquée, dans la section 5.2, nous avons décrit les résultats expérimentaux et la manière dont les trajectoires ont été analysées. Nous avons abordé les observables statiques comme la distribution d'équilibre, et les observables dynamiques comme les déplacements quadratiques moyens (MSDs) et les distributions de déplacement. En particulier, pour le mouvement perpendiculaire au mur, le MSD présente une distribution normale à court terme, et une valeur d'équilibre en plateau à long terme. De plus, la distribution du déplacement à court terme présente des propriétés non gaussiennes, ce qui est une signature directe de la modification de la mobilité induite par la frontière rigide.

Les propriétés statistiques d'un colloïde confiné étant comprises et correctement mesurées, nous nous sommes ensuite attachés à mesurer le coefficient de diffusion le coefficient de diffusion. Cette mesure a été effectuée à l'aide d'une nouvelle méthode développée par

Frishman et Ronceray [**frishman˙learning˙2020**], utilisant la théorie de l'information pour déduire la mobilité locale sur une base de fonctions d'ajustement, conduisant à une grande précision près de la surface (où il y a moins de données).

Toutes les données, de la distribution à l'équilibre à la mobilité, peuvent être décrites par les trois paramètres du système B, ℓ_{D} et ℓ_{B}. Nous avons donc construit une méthode multi-fitting, qui permet de déduire précisément ces paramètres en prenant en compte toutes les observables à la fois. De cette méthode, nous avons extrait précisément le potentiel d'équilibre à partir duquel nous avons calculé les forces conservatives. Cette mesure de force a été corroborée avec succès par une mesure indépendante basée sur les dérives des trajectoires.

Au total, nous sommes capables d'atteindre une résolution de force femtonewton limitée par le bruit thermique ainsi qu'une résolution spatiale à l'échelle nanométrique.

Comme le montre le chapitre 6, nous utilisons actuellement notre dispositif et notre méthode afin d'étudier le couplage original entre la théorie de la lubrification douce et le mouvement brownien. Nous travaillons également à repousser les limites de notre résolution pour mesurer les effets fins sur les cumulants d'ordre supérieur. Enfin, nous pensons que la capacité à mesurer de minuscules forces de surface, localement et à l'équilibre, ainsi que l'application possible de la méthode aux forces non-conservatives et aux situations hors équilibre, ouvrent des perspectives fascinantes pour la nanophysique et la biophysique.

A Appendix

inertial_Brownian_motion

September 25, 2021

1 Inertial Brownian motion simulation

The Inertial Langevin equation for a particle of mass m and some damping γ writes:

$$m\ddot{x} = -\gamma\dot{x} + \sqrt{2k_{\mathrm{B}}T\gamma}\mathrm{d}B_t \tag{1}$$

Integrating the latter equation using the Euler method, one can replace $\dot{x}$ by:

$$\dot{x} \simeq \frac{x_i - x_{i-1}}{\tau}\,, \tag{2}$$

$\ddot{x}$ by:

$$\begin{aligned}\ddot{x} &\simeq \frac{\frac{x_i - x_{i-1}}{\tau} - \frac{x_{i-1} - x_{i-2}}{\tau}}{\tau} \\ &= \frac{x_i - 2x_{i-1} + x_{i-2}}{\tau^2}\,.\end{aligned} \tag{3}$$

and finally, $\mathrm{d}B_t$ by a Gaussian random number w_i with a zero mean value and a τ variance, on can write x_i as:

$$x_i = \frac{2 + \tau/\tau_{\mathrm{B}}}{1 + \tau/\tau_{\mathrm{B}}}x_{i-1} - \frac{1}{1 + \tau/\tau_{\mathrm{B}}}x_{i-2} + \frac{\sqrt{2k_{\mathrm{B}}T\gamma}}{m(1 + \tau/\tau_{\mathrm{B}})}\tau w_i\,, \tag{4}$$

In the following, we use Python to simulate such a movement and check the properties of the mean squared displacement. Then, I propose a Cython implementation that permits a 200x speed improvement on the simulation.

[1]:
```
# Import important libraries
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Just some matplotlib tweaks
import matplotlib as mpl

mpl.rcParams["xtick.direction"] = "in"
mpl.rcParams["ytick.direction"] = "in"
mpl.rcParams["lines.markeredgecolor"] = "k"
```

```
mpl.rcParams["lines.markeredgewidth"] = 1.5
mpl.rcParams["figure.dpi"] = 200
from matplotlib import rc

rc("font", family="serif")
rc("text", usetex=True)
rc("xtick", labelsize="medium")
rc("ytick", labelsize="medium")
rc("axes", labelsize="large")


def cm2inch(value):
    return value / 2.54
```

[3]:
```
N = 1000000  # number of time steps
tau = 0.01  # simulation time step
m = 1e-8  # particle mass
a = 1e-6  # radius of the particle
eta = 0.001  # viscosity (here water)
gamma = 6 * np.pi * eta * a
kbT = 4e-21
tauB = m / gamma
```

With such properties we have a characteristic diffusion time $\tau_B = 0.53$ s.

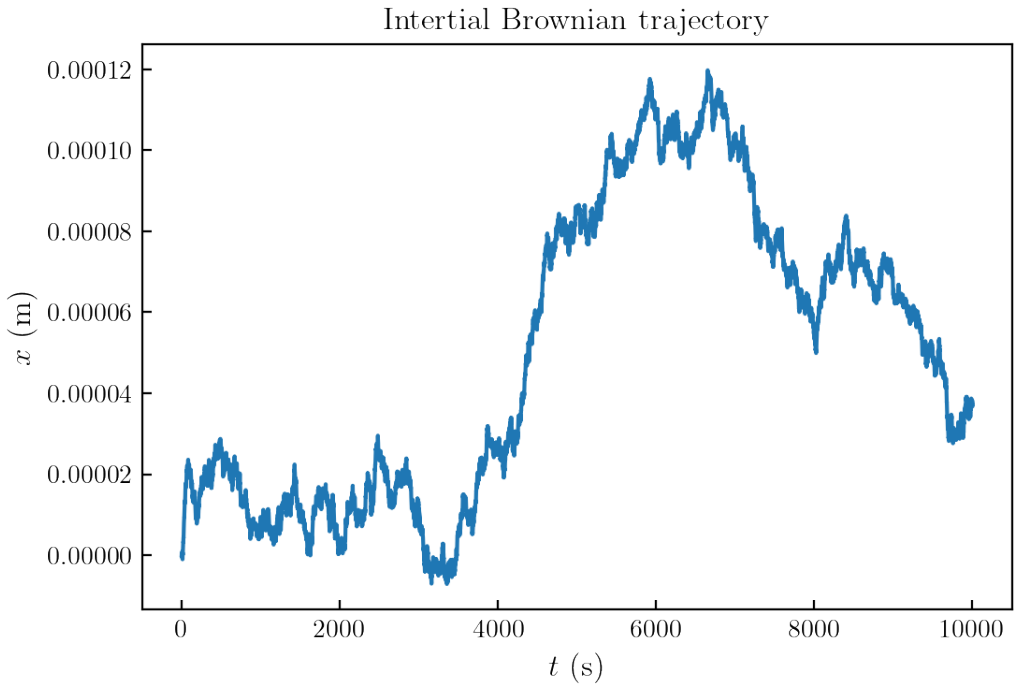
[4]:
```
def xi(xi1, xi2):
    """
    Function that compute the position of a particle using the full Langevin␣
 ↪Equation
    """
    t = tau / tauB
    wi = np.random.normal(0, np.sqrt(tau))
    return (
        (2 + t) / (1 + t) * xi1
        - 1 / (1 + t) * xi2
        + np.sqrt(2 * kbT * gamma) / (m * (1 + t)) * np.power(tau,1) * wi
    )
```

[5]:
```
def trajectory(N):
    """
    Function generating a trajectory of length N.
    """
    x = np.zeros(N)
    for i in range(2, len(x)):
        x[i] = xi(x[i - 1], x[i - 2])
    return x
```

Now that the functions are setup one can simply generate a trajectory of length N by simply calling the the function `trajectory()`

[6]:
```
# Generate a trajectory of 10e6 points.
x = trajectory(1000000)
```

[7]:
```
plt.plot(np.arange(len(x))*tau, x)
plt.title("Intertial Brownian trajectory")
plt.ylabel("$x$ (m)")
plt.xlabel("$t$ (s)")
plt.show()
```



1.1 Cross checking

We now check that the simulated trajectory gives us the correct MSD properties to ensure the simulation si done properly. The MSD given by:

$$\mathrm{MSD}(\Delta t) = \langle (x(t) - x(t+\Delta t)^2 \rangle \Big|_t \,, \tag{5}$$

with Δt a lag time. The MSD, can be computed using the function defined in the cell below. For a lag time $\Delta t \ll \tau_B$ we should have:

$$\mathrm{MSD}(\Delta t) = \frac{k_B T}{m} \Delta t^2 \,, \tag{6}$$

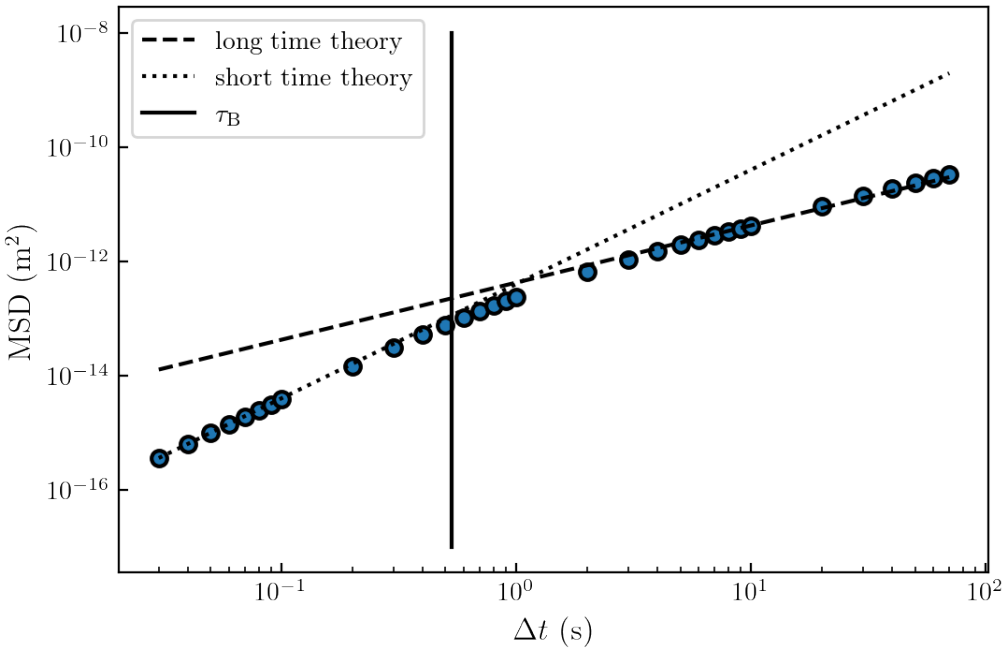
and for $\Delta t \gg \tau_B$:

$$\mathrm{MSD}(\tau) = 2D\Delta t\,, \tag{7}$$

with $D = k_\mathrm{B}T/(6\pi\eta a)$.

[8]:
```
t = np.array([*np.arange(3,10,1), *np.arange(10,100,10), *np.
 ↪arange(100,1000,100), *np.arange(1000,8000,1000)])
def msd(x,Dt):
    """Function that return the MSD for a list of time index t for a trajectory␣
 ↪x"""
    _msd = lambda x, t : np.mean((x[:-t] - x[t:])**2)
    return [_msd(x,i) for i in t]
MSD = msd(x,t)
```

[9]:
```
D = kbT/(6*np.pi*eta*a)
t_plot = t*tau
plt.loglog(t*tau,MSD, "o")
plt.plot(t*tau, (2*D*t_plot), "--", color = "k", label="long time theory")
plt.plot(t*tau, kbT/m * t_plot**2, ":", color = "k", label="short time theory")
plt.ylabel("MSD (m$^2$)")
plt.xlabel("$\Delta t$ (s)")
horiz_data = [1e-8, 1e-17]
t_horiz = [tauB, tauB]
plt.plot(t_horiz, horiz_data, "k", label="$\\tau_\mathrm{B}$")
plt.legend()
plt.show()
```



The simulations gives expected results. However, with the computer used, 6 seconds are needed to generate this trajectory. If someone wants to look at fine effects and need to generate millions of trajectories it is too long. In order to fasten the process, in the following I use Cython to generate the trajectory using C language.

1.2 Cython acceleration

[10]:
```
# Loading Cython library
%load_ext Cython
```

We now write the same functions as in the first part of the appendix. However, we now indicate the type of each variable.

[11]:
```
%%cython

import cython
cimport numpy as np
import numpy as np
from libc.math cimport sqrt
ctypedef np.float64_t dtype_t

cdef int N = 1000000 # length of the simulation

cdef dtype_t tau = 0.01 # simulation time step
cdef dtype_t m = 1e-8 # particle mass
cdef dtype_t a = 1e-6 # radius of the particle
cdef dtype_t eta = 0.001 # viscosity (here water)
cdef dtype_t gamma = 6 * 3.14 * eta * a
cdef dtype_t kbT = 4e-21
cdef dtype_t tauB = m/gamma
cdef dtype_t[:] x = np.zeros(N)

@cython.boundscheck(False)
@cython.wraparound(False)
@cython.nonecheck(False)
@cython.cdivision(True)
cdef dtype_t xi_cython( dtype_t xi1, dtype_t xi2, dtype_t wi):
    cdef dtype_t t = tau / tauB
    return (
        (2 + t) / (1 + t) * xi1
        - 1 / (1 + t) * xi2
        + sqrt(2 * kbT * gamma) / (m * (1 + t)) * tau * wi
    )

@cython.boundscheck(False)
```

```
@cython.wraparound(False)
@cython.nonecheck(False)
cdef dtype_t[:] _traj(dtype_t[:] x, dtype_t[:] wi):
    cdef int i
    for i in range(2, N):

        x[i] = xi_cython(x[i-1], x[i-2], wi[i])
    return x


def trajectory_cython():


    cdef dtype_t[:] wi = np.random.normal(0, np.sqrt(tau), N).astype('float64')


    return _traj(x, wi)
```

[12]:
```
%timeit trajectory(1000000)
```

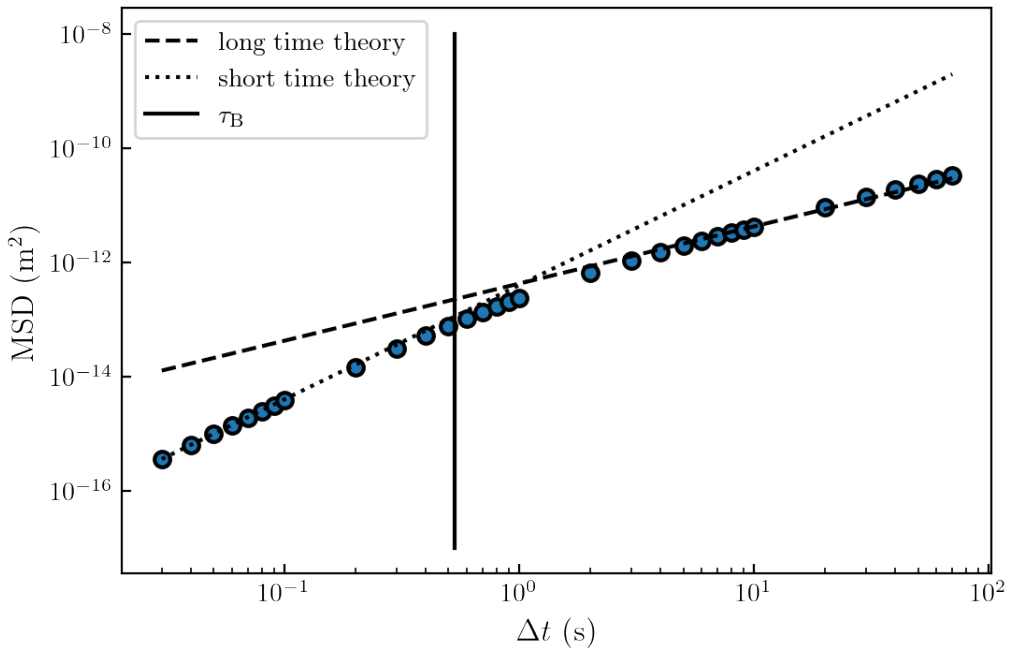
```
6.79 s ± 92.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[13]:
```
%timeit trajectory_cython()
```

```
30.6 ms ± 495 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Again, we check that the results given through the use of Cython gives the correct MSD

[14]:
```
x=np.asarray(trajectory_cython())
D = kbT/(6*np.pi*eta*a)
t_plot = t*tau
plt.loglog(t*tau,MSD, "o")
plt.plot(t*tau, (2*D*t_plot), "--", color = "k", label="long time theory")
plt.plot(t*tau, kbT/m * t_plot**2, ":", color = "k", label="short time theory")

horiz_data = [1e-8, 1e-17]
t_horiz = [tauB, tauB]
plt.plot(t_horiz, horiz_data, "k", label="$\\tau_\mathrm{B}$")
plt.xlabel("$\\Delta t$ (s)")
plt.ylabel("MSD (m$^2$)")
plt.legend()
plt.show()
```



1.2.1 Conclusion

Finally, one only needs $\simeq$ 30 ms to generate the trajectory instead of $\simeq$ 7 s which is a $\simeq 250\times$ improvement speed. The simulation si here bound to the time needed to generate the array of random numbers which is still done using numpy function. After further checking, Numpy random generation si as optimize as one could do so there is no benefit on cythonizing the random generation. For the sake of completness one could fine a Cython version to generate random numbers. Found thanks to Senderle on Stackoverflow. Tacking into account that, the time improvment on the actual computation of the trajectory **without** the random number generation is done with an $\simeq 1100\times$ improvement speed.

[15]:
```
%%cython
from libc.stdlib cimport rand, RAND_MAX
from libc.math cimport log, sqrt
import numpy as np
import cython

cdef double random_uniform():
    cdef double r = rand()
    return r / RAND_MAX

cdef double random_gaussian():
    cdef double x1, x2, w

    w = 2.0
```

```
    while (w >= 1.0):
        x1 = 2.0 * random_uniform() - 1.0
        x2 = 2.0 * random_uniform() - 1.0
        w = x1 * x1 + x2 * x2

    w = ((-2.0 * log(w)) / w) ** 0.5
    return x1 * w

@cython.boundscheck(False)
cdef void assign_random_gaussian_pair(double[:] out, int assign_ix):
    cdef double x1, x2, w

    w = 2.0
    while (w >= 1.0):
        x1 = 2.0 * random_uniform() - 1.0
        x2 = 2.0 * random_uniform() - 1.0
        w = x1 * x1 + x2 * x2

    w = sqrt((-2.0 * log(w)) / w)
    out[assign_ix] = x1 * w
    out[assign_ix + 1] = x2 * w

@cython.boundscheck(False)
def my_uniform(int n):
    cdef int i
    cdef double[:] result = np.zeros(n, dtype='f8', order='C')
    for i in range(n):
        result[i] = random_uniform()
    return result

@cython.boundscheck(False)
def my_gaussian(int n):
    cdef int i
    cdef double[:] result = np.zeros(n, dtype='f8', order='C')
    for i in range(n):
        result[i] = random_gaussian()
    return result

@cython.boundscheck(False)
def my_gaussian_fast(int n):
    cdef int i
    cdef double[:] result = np.zeros(n, dtype='f8', order='C')
    for i in range(n // 2):  # Int division ensures trailing index if n is odd.
        assign_random_gaussian_pair(result, i * 2)
    if n % 2 == 1:
        result[n - 1] = random_gaussian()
```

```
    return result
```

[16]:
```
%timeit my_gaussian_fast(1000000)
```

```
30.9 ms ± 941 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

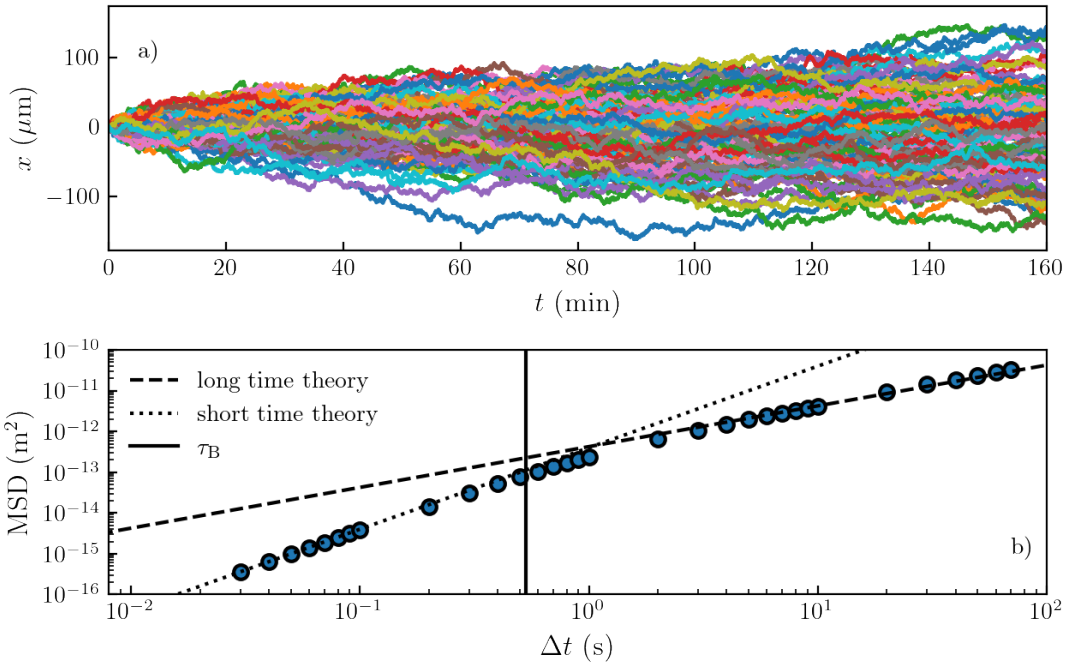
[17]:
```
%timeit np.random.normal(0,1,1000000)
```

```
26.4 ms ± 1.87 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

One can thus see, that even a pure C implementation can be slower than the Numpy one, thanks to a great optimization.

[18]:
```
fig = plt.figure(figsize = (cm2inch(16), cm2inch(10)))
gs = fig.add_gridspec(2, 1)
f_ax1 = fig.add_subplot(gs[0, 0])
for i in range(100):
    x = np.asarray(trajectory_cython())* 1e6
    plt.plot(np.arange(N)*tau / 60, x)

plt.ylabel("$x$ ($\mathrm{\mu m}$)")
plt.xlabel("$t$ (min)")
plt.text(5,100, "a)")
plt.xlim([0,160])
f_ax1 = fig.add_subplot(gs[1, 0])

x=np.asarray(trajectory_cython())
D = kbT/(6*np.pi*eta*a)
plt.loglog(t*tau,MSD, "o")
t_plot = np.linspace(0.5e-2,5e3,1000)
plt.plot(t_plot, (2*D*t_plot), "--", color = "k", label="long time theory")
plt.plot(t_plot, kbT/m * t_plot**2, ":", color = "k", label="short time theory")

horiz_data = [1e-7, 1e-18]
t_horiz = [tauB, tauB]
plt.plot(t_horiz, horiz_data, "k", label="$\\tau_\mathrm{B}$")
plt.ylabel("MSD (m$^2$)")
plt.xlabel("$\\Delta t$ (s)")
ax = plt.gca()
locmaj = mpl.ticker.LogLocator(base=10.0, subs=(1.0, ), numticks=100)
ax.yaxis.set_major_locator(locmaj)
locmin = mpl.ticker.LogLocator(base=10.0, subs=np.arange(2, 10) * .1,
                               numticks=100)
ax.yaxis.set_minor_locator(locmin)
ax.yaxis.set_minor_formatter(mpl.ticker.NullFormatter())
plt.legend(frameon=False)
plt.text(0.7e2,1e-15, "b)")
plt.xlim([0.8e-2,1e2])
plt.ylim([1e-16,1e-10])
```

```
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=1.0)

plt.savefig("intertial_langevin.pdf")
plt.show()
```



Fitting procedure using Pylorenzmie

1 Fitting procedure using Pylorenzmie

In order to fit an hologram, I used the pylorenzmie model which provides a set of python classes in order to analyse holographic microscopy data.

Pylorenzmie can be download on the David Grier's github repository: https://github.com/davidgrier/pylorenzmie.

What I actually get from the experiments are mp4 movies, in order to analyze them easily, I constructed a wrapper around the pylorenzmie module which can be found on my repository: https://github.com/eXpensia/wraplorenzmie.

This wrapper permits to do the following pipeline:

- Directly load the movies
- Compute the back ground.
- Use the first image in order to get the pre guesses
- Fit the 10 000 first images to determine precisely the radius and index of a particle.
- Use the later information in order to fit the whole movie (and save the data in the same time)

One that done, the trajectory be analyzed separately.

[1]:
```
# We first start by import the important modules

import wraplorenzmie.utilities.utilities as utilities
import wraplorenzmie.fits.fit as fit
import imageio
# For Plotting.
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
#sns.set(style='white', font_scale=2)
%matplotlib inline
import matplotlib as mpl
```

```
mpl.rcParams["figure.dpi"] = 200
from matplotlib import rc
rc('font', family='serif')
rc('text', usetex=True)
rc('xtick', labelsize='x-small')
rc('ytick', labelsize='x-small')

def cm2inch(value):
    return value/2.54
```

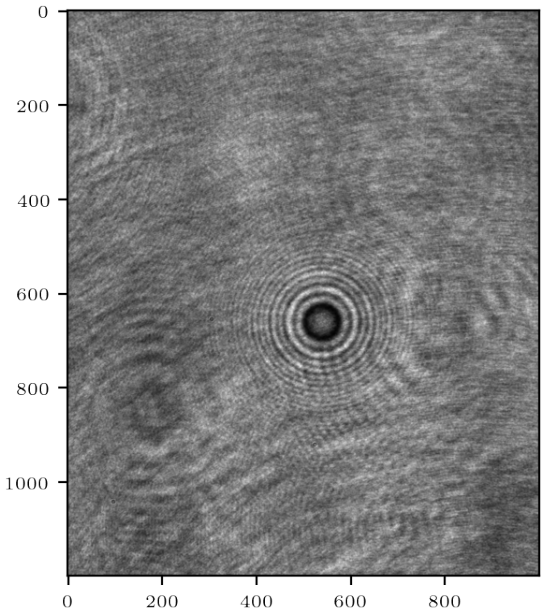
```
No module named 'pylorenzmie.fitting.cython.cminimizers'
```

[2]:
```
#We load the movie
vid = utilities.
 ↪video_reader("Basler_acA1920-155um__22392621__20200527_162231224.mp4")
```

[3]:
```
# A function that permits to compute de radial profile of an image this␣
 ↪will later be used in order to see if the fits are done correctly
def radial_profile(data, center=None):
    if center==None:
        center = np.array(np.shape(data)) / 2

    y, x = np.indices((data.shape))
    r = np.sqrt((x - center[0])**2 + (y - center[1])**2)
    r = r.astype(int)

    tbin = np.bincount(r.ravel(), data.ravel())
    nr = np.bincount(r.ravel())
    radialprofile = tbin / nr

    T = data.ravel()
    V = r.ravel()

    err = [np.std(T[V == u]) for u in np.unique(V)]

    return radialprofile, err
```

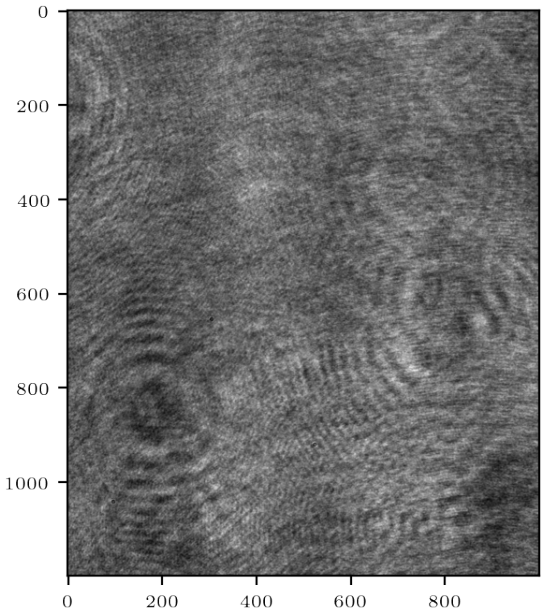
[4]:
```
# We take a look at the first image of the movie
image = vid.get_image(1)
plt.imshow(image,cmap="gray")
```

[4]:
```
<matplotlib.image.AxesImage at 0x1a70b337be0>
```



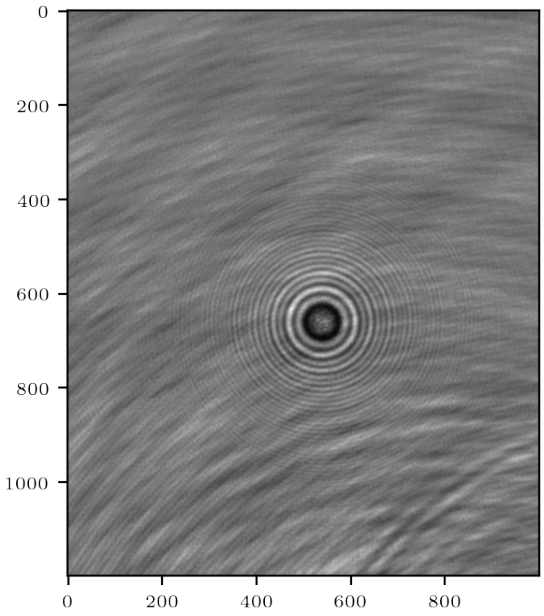
[5]:
```
# set the background image (it can also be computed using vid.
 ↪get_background method)
vid.number = 125000
vid.background = np.array(imageio.imread("background.tiff"))
#vid.background = vid.get_background(n=50) # n is the number of image to␣
 ↪use to compute the background
plt.imshow(vid.background,cmap="gray")
```

[5]: <matplotlib.image.AxesImage at 0x1a70bc56490>

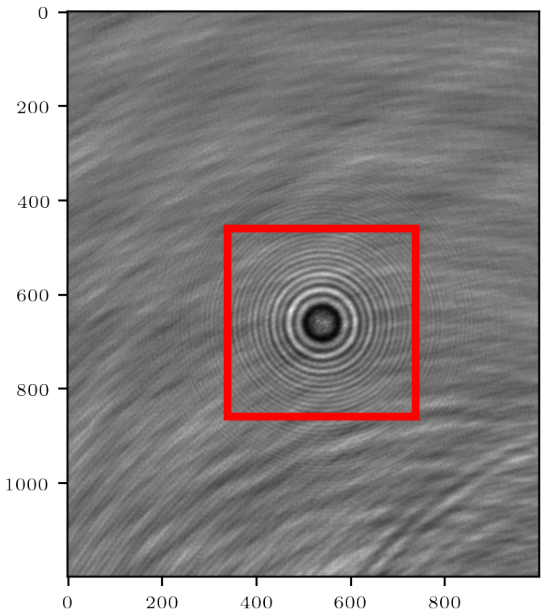


[6]:
```
imageio.imwrite("background.tiff",vid.background)# We save the background␣
↪for possible later use.
```

[7]:
```
# the normalized image, we can see that their is some movement in the␣
↪background.
# This could be avoided by computed the background as a function of the␣
↪time, if the particle diffuses enough.
normed_image = utilities.normalize(image,vid.background)
plt.imshow(normed_image,cmap="gray")
normed_image = normed_image
```



[8]:
```
# We found the possition of the particle
feature = utilities.center_find(image)[0]
utilities.plot_bounding(normed_image,feature)
```

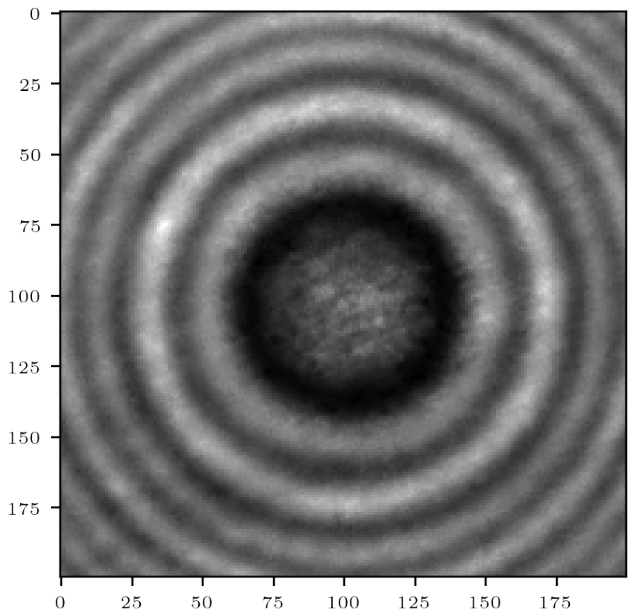


1.1 Fitting the first image

We fit the first image in order to get the preguess. We first start by croping the hologram.

[9]:
```
xc, yc, w, h = feature[0]
x_center = xc
y_center = yc
h=200
im_c = fit.crop(image, int(xc), int(yc), int(h))
bk_c = fit.crop(vid.background, int(xc), int(yc), int(h))
cropped = utilities.normalize(im_c,bk_c, dark_count = np.min(im_c))
cropped = cropped / np.mean(cropped)
plt.imshow(cropped,cmap = "gray")
```

[9]:
```
<matplotlib.image.AxesImage at 0x1a71d7e6d00>
```



[10]:
```
# We setup the fitting method.
fitter = fit.fitting(cropped,0.532,0.0513)
fitter.make_guess(1.50,1.59,12,alpha = 1,fit_r=True,␣
 ↪fit_n=True,fit_alpha=True)
```

[11]:
```
# We do the actual fit.
result = fitter.fit_single(cropped, method = "lm")
```

[12]:
```
zo = result.result["x"][2]*0.0513
print(result.result["x"][2]*0.0513)
print(result.redchi)
print(result.result["x"])
```

```
11.427616273713154
7.2459765196825305
[101.23514587 103.00299474 222.76055114   1.5310255    1.58239091
   1.00198476]
```

We can plot the result to see if the fit worked properly, and, for a more quantitative comparison we can compute the radial intensity profile of both hologram and compare them.

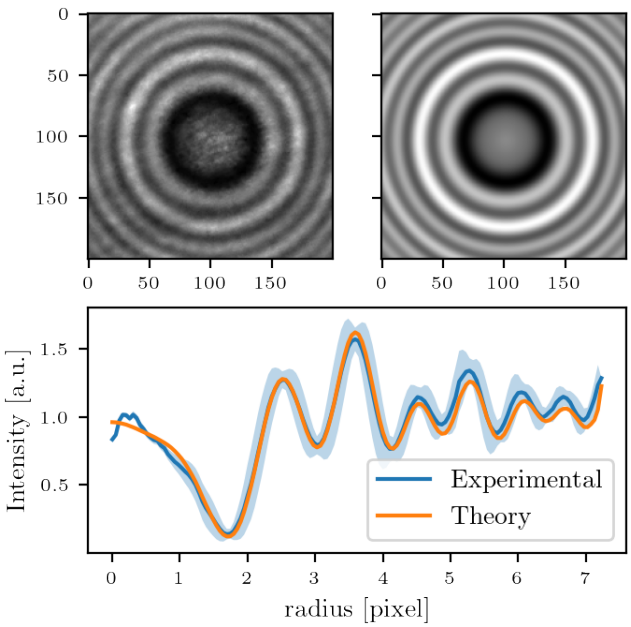
[13]:
```
center = np.array(np.shape(fitter.image))
```

[14]:
```
radial_exp, err = radial_profile(fitter.image)
theo_exp, err = radial_profile(fitter.fitter.model.hologram().
 ↪reshape(fitter.shape))
# computing first the holgram using the fit resutlt
```

[15]:
```
fit_data = {}
radius_radial = np.arange(len(radial_exp)) * 0.0513
plt.figure(figsize = (15,15))
fig = plt.figure(figsize=(cm2inch(8.6),1.65*cm2inch(8.6)))
fig.subplots_adjust(left=0.14, bottom=.12, right=.99, top=.98)

plt.subplot(2,2,1)
plt.imshow(fitter.image, cmap = "gray")
#plt.title('subplot(2,2,1)')

fit_data["exp_image"] = fitter.image

plt.subplot(2,2,2)
plt.imshow(fitter.fitter.model.hologram().reshape(fitter.shape), cmap =␣
 ↪"gray")
frame1 = plt.gca()
frame1.axes.yaxis.set_ticklabels([])

fit_data["th_image"] = fitter.fitter.model.hologram().reshape(fitter.
 ↪shape)


#plt.title('subplot(2,2,2)')

plt.subplot(2,2,(3,4))
```

```
plt.plot(radius_radial, radial_exp, label="Experimental")
plt.fill_between(radius_radial, radial_exp - err, radial_exp + err, alpha␣
↪= 0.3)
plt.plot(radius_radial, theo_exp, label="Theory")
plt.legend()
plt.xlabel("radius [pixel]")
plt.ylabel("Intensity [a.u.]")

fit_data["I_r_exp"] = radial_exp
fit_data["I_errr_exp"] = err

fit_data["theo_exp"] = theo_exp
fit_data["I_radius"] = radius_radial

fig.set_size_inches(cm2inch(8.6), cm2inch(1.6 * 8.6/1.618))
plt.savefig("fit_fig.pdf")
```

```
<Figure size 3000x3000 with 0 Axes>
```



[16]:
```
fitter.fit_video(vid =␣
↪vid,savefile="find_nrfit_result_dur_27052020_n_r_fix_0p0513_wav532.
↪dat",xc = x ,yc= y, h = 200, n_end=10000,method = "lm")
```

```
100% 9999/9999 [12:39<00:00, 13.17it/s]
```

[17]:
```
# Since the measurement or not saved into the ram we need to load it
n_r = np.fromfile('find_nr_exame.dat', dtype=np.float64)
n_r = n_r.reshape(len(n_r)//10,10)
r = n_r[:,3]
n = n_r[:,4]
```

1.2 Fitting the n, r distributiton using a KDE estimator

To find the most probable couple of r/n we use a kde estimator using `seaborn`

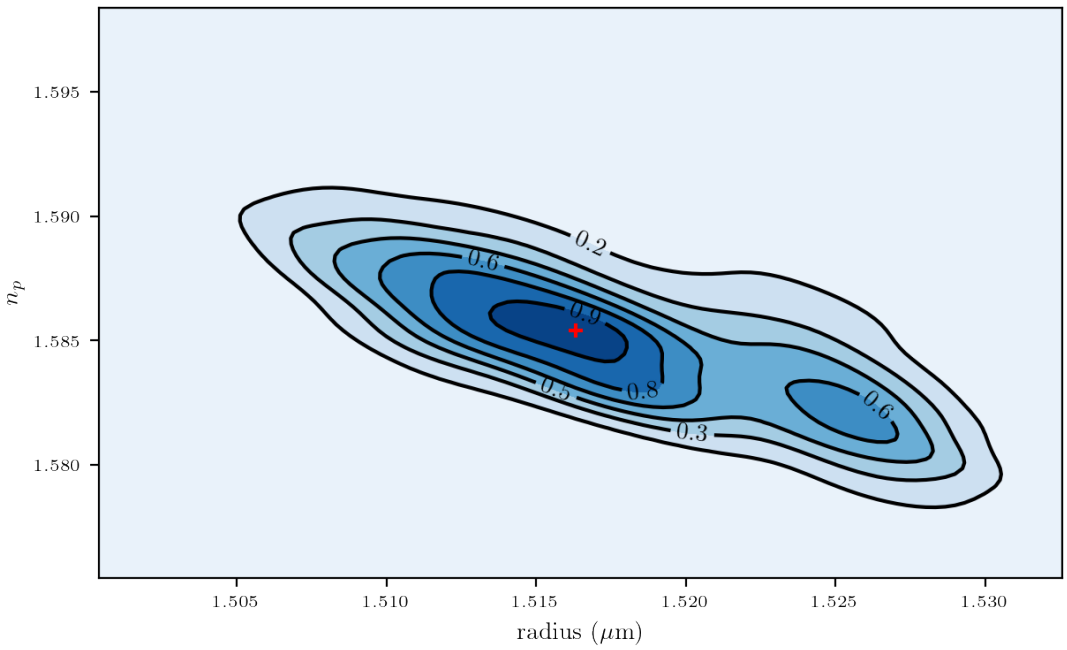
[18]:
```
import numpy as np
import scipy.stats as st
import matplotlib.ticker as ticker

data = np.random.multivariate_normal((0, 0), [[0.8, 0.05], [0.05, 0.7]],␣
↪100)
x = r[(r>1.5) & (r<1.555)]
y = n[(r>1.5) & (r<1.555)]
xmin, xmax = np.min(x), np.max(x)
ymin, ymax = np.min(y), np.max(y)

# Peform the kernel density estimate
xx, yy = np.mgrid[xmin:xmax:100j, ymin:ymax:100j]
positions = np.vstack([xx.ravel(), yy.ravel()])
values = np.vstack([x, y])
kernel = st.gaussian_kde(values)
f = np.reshape(kernel(positions).T, xx.shape)
f = f/np.max(f)
```

[19]:
```
np.round(np.max(f))
```

[19]:
```
1.0
```

[20]:
```
fig = plt.figure()
fig.subplots_adjust(left=0.16, bottom=.20, right=.99, top=.99)
ax = fig.gca()
#ax.set_xlim(1.505, 1.53)
#ax.set_ylim(1.575, 1.6)
# Contourf plot
cfset = ax.contourf(xx, yy, f, cmap='Blues')
## Or kernel density estimate plot instead of the contourf plot
#ax.imshow(np.rot90(f), cmap='Blues', extent=[xmin, xmax, ymin, ymax])
# Contour plot
cset = ax.contour(xx, yy, f, colors='k', levels=6)
```

```
# Label plot
ax.xaxis.set_major_formatter(ticker.FormatStrFormatter('%1.3f'))
ax.yaxis.set_major_formatter(ticker.FormatStrFormatter('%1.3f'))
ax.clabel(cset, inline=1, fontsize=10, fmt="%1.1f")
plt.scatter(xx[np.where(f == 1)],yy[np.where(f == 1)], color = "red",␣
 ↪marker="+")
ax.set_xlabel("radius ($\mathrm{\mu m}$)")
ax.set_ylabel("$n_p$")
#plt.title("KDE r n")
fig.set_size_inches(cm2inch(16), cm2inch(9.9))

plt.tight_layout()
fig.savefig('KDErn.pdf')
#pl.show()
```



[21]:
```
print(" n determined with : mu={0}, sigma={1}".format(np.mean(yy[np.
 ↪where(f > 0.1)]), np.std(yy[np.where(f > 0.1)])))
print(" r determined with : mu={0}, sigma={1}".format(np.mean(xx[np.
 ↪where(f > 0.1)]), np.std(xx[np.where(f > 0.1)])))
```

```
n determined with : mu=1.5851200393768743, sigma=0.003267685282504072
r determined with : mu=1.5181266656310368, sigma=0.00682411690457934
```

[22]:
```
(mu_n, mu_r) = np.mean(yy[np.where(f > 0.1)]) ,  np.mean(xx[np.where(f >␣
↪0.1)])
```

1.3 Fitting the whole movie

Now that the measurement of n and r is one we can move on the measurement of the whole trajectory by simply using `fitter.fit_video`. For demonstration purposes, I only fit here at $\simeq$ 22 image per seconds, if can goes up to at least 60 with recent GPU.

[23]:
```
del fitter
fitter = fit.fitting(cropped,0.532,0.0513)
fitter.make_guess(mu_r,mu_n,zo,alpha = 1,fit_r=False,␣
↪fit_n=False,fit_alpha=False)
#result = fitter.fit_single(cropped, method = "lm")
fitter.fit_video(vid =␣
↪vid,savefile="fit_result_1p5kPa_18122019_n_r_fix_0p0883_wav_532_ex1.
↪dat",xc = xc ,yc= yc, h = 200, n_end=10000,method = "lm")
```

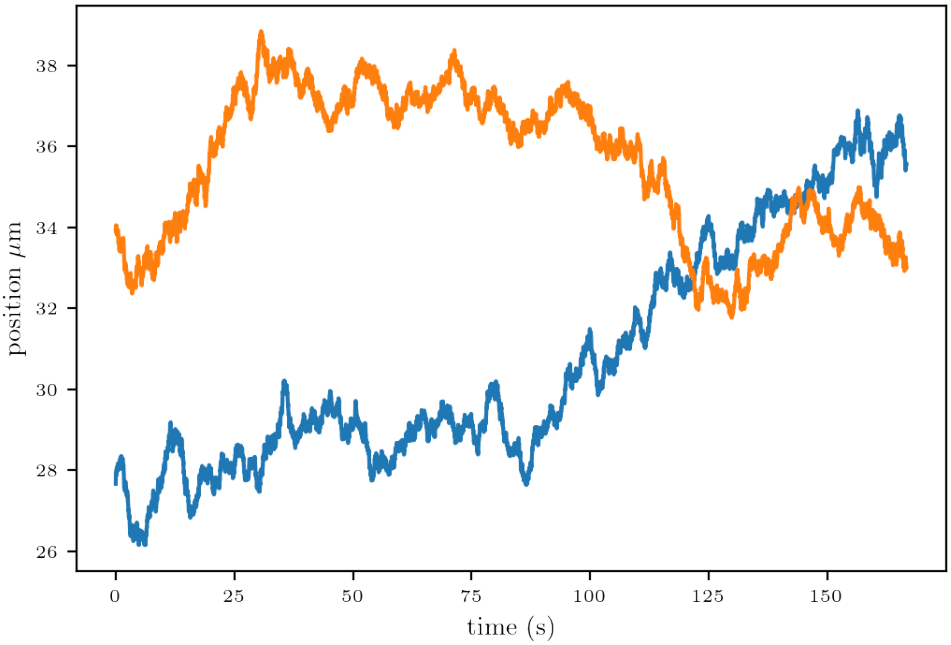
```
100% 9999/9999 [07:24<00:00, 22.47it/s]
```

[24]:
```
import numpy as np
data = np.fromfile('fit_result_1p5kPa_18122019_n_r_fix_0p0883_wav_532_ex1.
↪dat', dtype=np.float64)
data = data.reshape(len(data)//10,10)
x = data[:,0]*0.0513
y = data[:,1]*0.0513
z = data[:,2]*0.0513
```

1.4 Plot the trajectory

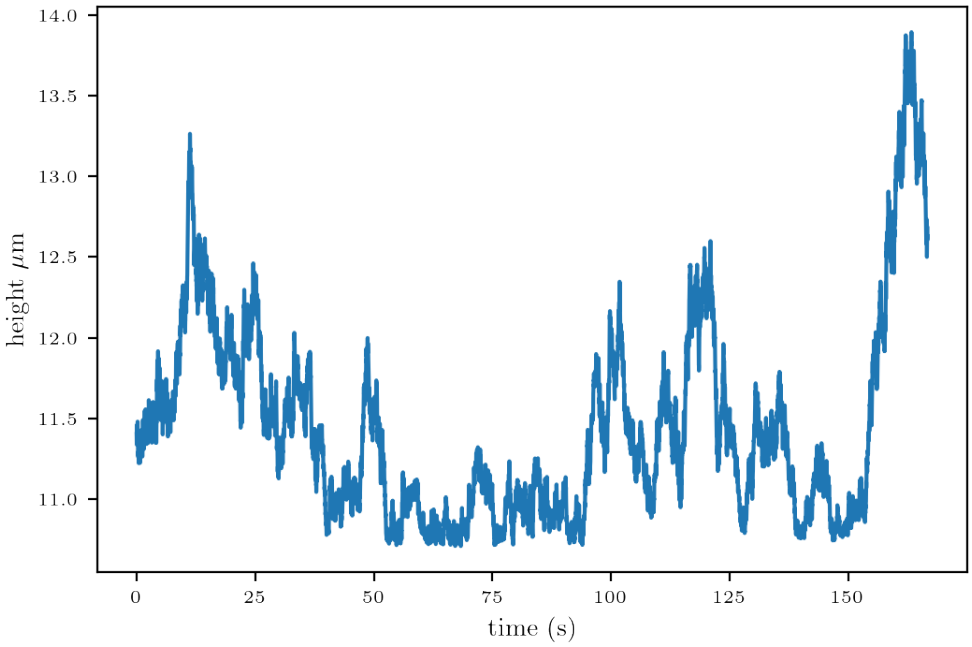
[25]:
```
plt.plot(np.arange(len(z))/60, x)
plt.plot(np.arange(len(z))/60, y)
plt.ylabel("position $\mathrm{\mu m}$")
plt.xlabel("time (s)")
```

[25]:
```
Text(0.5, 0, 'time (s)')
```



[26]:
```
plt.plot(np.arange(len(z))/60, z)
plt.ylabel("height $\mathrm{\mu m}$")
plt.xlabel("time (s)")
```

[26]: `Text(0.5, 0, 'time (s)')`



PHYSICAL REVIEW RESEARCH **3**, L032011 (2021)

Letter

Stochastic inference of surface-induced effects using Brownian motion

Maxime Lavaud,[1] Thomas Salez,[1,2,*] Yann Louyer,[1] and Yacine Amarouchene[1,†]

[1]*Univ. Bordeaux, CNRS, LOMA, UMR 5798, F-33405 Talence, France*

[2]*Global Station for Soft Matter, Global Institution for Collaborative Research and Education, Hokkaido University, Sapporo, Hokkaido 060-0808, Japan*

(Received 8 December 2020; revised 28 April 2021; accepted 2 June 2021; published 8 July 2021)

Brownian motion in confinement and at interfaces is a canonical situation, encountered from fundamental biophysics to nanoscale engineering. Using the Lorenz-Mie framework, we optically record the thermally induced tridimensional trajectories of individual microparticles, within salty aqueous solutions, in the vicinity of a rigid wall, and in the presence of surface charges. We construct the time-dependent position and displacement probability density functions, and study the non-Gaussian character of the latter which is a direct signature of the hindered mobility near the wall. Based on these distributions, we implement a robust and self-calibrated multifitting method, allowing for the thermal-noise-limited inference of diffusion coefficients spatially resolved at the nanoscale, equilibrium potentials, and forces at the femtonewton resolution.

DOI: 10.1103/PhysRevResearch.3.L032011

Brownian motion is a central paradigm in modern science. It has implications in fundamental physics, biology, and even finance, to name a few. By understanding that the apparent erratic motion of colloids is a direct consequence of the thermal motion of surrounding fluid molecules, pioneers like Einstein and Perrin provided decisive evidence for the existence of atoms [1,2]. Specifically, free Brownian motion in the bulk is characterized by a typical spatial extent evolving as the square root of time, as well as Gaussian displacements.

At a time of miniaturization and interfacial science, and moving beyond the idealized bulk picture, it is relevant to consider the added roles of boundaries to the above context. Indeed, Brownian motion at interfaces and in confinement is a widespread practical situation in microbiology and nanofluidics. In such a case, surface effects become dominant and alter drastically the Brownian statistics, with key implications towards (i) the understanding and smart control of the interfacial dynamics of microscale entities and (ii) high-resolution measurements of surface forces at equilibrium. Interestingly, a confined colloid will exhibit non-Gaussian statistics in displacements, due to the presence of multiplicative noises induced by the hindered mobility near the wall [3–5]. Additionally, the particle can be subjected to electrostatic or van der Waals forces [6] exerted by the interface, and might experience slippage too [7,8]. Considering the two-body problem, the nearby boundary can also induce some effective interaction [9]. Previous studies have designed novel methods to measure the diffusion coefficient of confined colloids [10–16], or to infer surface forces [17–22]. However, such a statistical inference is still an experimental challenge, and a precise calibration-free method taking simultaneously into account the whole ensemble of relevant properties, over broad spatial and time ranges, is currently lacking.

In this Rapid Communication, we aim at filling the previously identified gap by implementing a method of statistical inference on a set of trajectories of individual microparticles recorded by holographic microscopy. The buoyant particles are free to evolve within salty aqueous solutions, near a rigid substrate, and in the presence of surface charges. We primarily reconstruct the equilibrium probability distribution function of the position, as well as the time-resolved probability distribution functions of the displacements in directions transverse and normal to the wall, including in particular the mean-squared displacements (MSDs). Special attention is dedicated to the non-Gaussian statistics, for time scales broadly ranging from tens of milliseconds to several tens of minutes. Furthermore, we implement the advanced inference method recently proposed [23]. Additionally, an optimization scheme is used in order to determine precisely all the free physical parameters and the actual distance to the wall, at once. All together, this procedure leads to the robust calibration-free inference of the two central quantities of the problem: (i) the space-dependent short-term diffusion coefficients, with a nanoscale spatial resolution; and (ii) the total force experienced by the particle, at the thermal-noise limited femtonewton resolution. These main results are summarized in Fig. 1, the goal of this Rapid Communication being the detailed obtention of which.

The experimental setup is schematized in Fig. 2(a). A sample consists of a parallelepipedic chamber ($1.5\,\mathrm{cm} \times 1.5\,\mathrm{cm} \times 150\,\mu\mathrm{m}$), made from two glass covers and a parafilm spacer and sealed with vacuum grease, containing a dilute suspension of spherical polystyrene beads (Sigma Aldrich) with nominal radii $a = 1.5 \pm 0.035\,\mu\mathrm{m}$, at room

*thomas.salez@u-bordeaux.fr

†yacine.amarouchene@u-bordeaux.fr

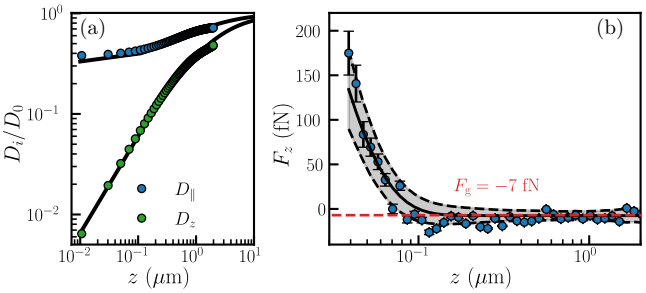


FIG. 1. (a) Measured local short-term diffusion coefficients D_i of the microparticle, normalized by the bulk value D_0, as functions of the distance z to the wall [see Fig. 2(c)], along both a transverse direction x or y ($D_i = D_\parallel = D_x = D_y$, blue) and the normal direction z ($D_i = D_z$, green) to the wall. The solid lines are the theoretical predictions, $D_\parallel(z) = D_0\eta/\eta_\parallel(z)$ and $D_z(z) = D_0\eta/\eta_z(z)$, using the local effective viscosities $\eta_\parallel(z)$ and $\eta_z(z)$ of Eqs. (3) and (4), respectively. (b) Total normal conservative force F_z exerted on the particle as a function of the distance z to the wall, reconstructed from Eq. (11), using Eq. (4). The solid line corresponds to Eq. (13), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm. The black dashed lines and gray area indicate the amplitude of the thermal noise computed from Eq. (12). The horizontal red dashed line indicates the buoyant weight $F_g = -7$ fN of the particle.

temperature T, in distilled water (type 1, MilliQ device) of viscosity $\eta = 1$ mPa s. The sample is illuminated by a collimated laser beam with a 532-μm wavelength. The light scattered by one colloidal particle at a given time t interferes with the incident beam. An oil-immersion objective lens (x60 magnification, 1.30 numerical aperture) collects the resulting instantaneous interference pattern, and relays it to a camera

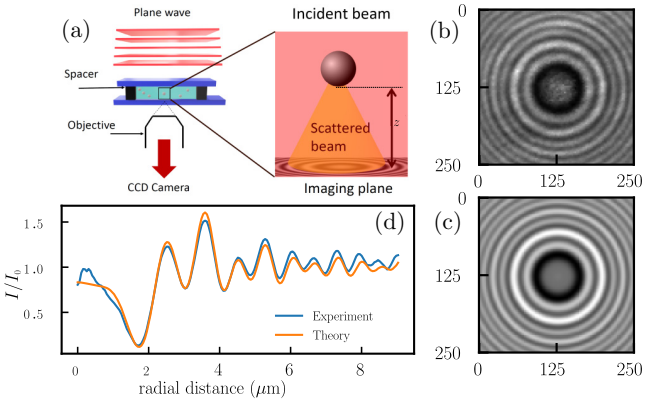


FIG. 2. (a) Schematic of the experimental setup. A laser plane wave of intensity I_0 illuminates the chamber containing a dilute suspension of microspheres in water. The light scattered by a particle interferes with the incident beam onto the focal plane of an objective lens, that magnifies the interference pattern and relays it to a camera. (b) Typical experimental interference pattern produced by one particle. (c) Corresponding best-fit Lorenz-Mie interference pattern [24–28], providing a distance $z = 11.24 \pm 0.2\,\mu$m to the wall, as well as the radius $a = 1.518 \pm 0.006\,\mu$m and refractive index $n = 1.584 \pm 0.006$ of the particle. (d) Angular averages of the intensities I (normalized by I_0) from the experimental and theoretical interference patterns, as functions of the radial distance to the z axis.

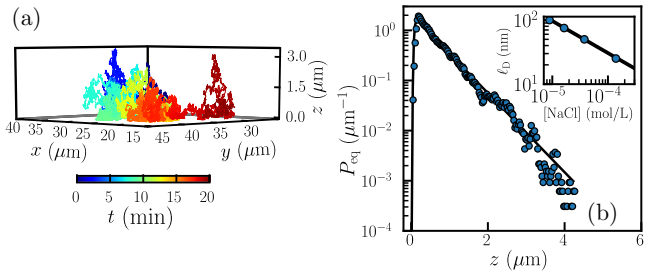


FIG. 3. (a) Typical measured tridimensional trajectory $\mathbf{r}(\mathbf{t}) = [\mathbf{x}(\mathbf{t}), \mathbf{y}(\mathbf{t}), \mathbf{z}(\mathbf{t})]$ of the microparticle near the wall ($z = 0$). (b) Measured equilibrium probability density function P_{eq} of the distance z between the particle and the wall. The solid line represents the best fit to the normalized Gibbs-Boltzmann distribution in position, using the total potential energy $U(z)$ of Eq. (1), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm. The inset shows the measured Debye length ℓ_D as a function of salt concentration [NaCl]. The solid line is the expected Debye relation $\ell_D = 0.304/\sqrt{[\mathrm{NaCl}]}$, for a single monovalent salt in water at room temperature.

with a 51.6-nm/pixel resolution [see Fig. 2(b)]. The exposure time for each frame is fixed to 3 ms to avoid motion-induced blurring of the image. The angular average of the intensity profile from each time frame is then fitted [see Figs. 2(c) and 2(d)] to the Lorenz-Mie scattering function [24–28], which provides the particle radius a, its refractive index n, and its instantaneous tridimensional position $\mathbf{r} = (\mathbf{x}, \mathbf{y}, \mathbf{z})$. To reduce the uncertainty on the position measurement, we first calibrate $a = 1.518 \pm 0.006\,\mu$m and $n = 1.584 \pm 0.006$ separately from the first 10^5 time frames. The obtained refractive index is consistent with the one reported in [16]. Then, for each subsequent time frame, the only remaining fitted quantity is $\mathbf{r}$, which allows us to reconstruct the trajectory $\mathbf{r}(t)$ with a nanometric spatial resolution, as shown in Fig. 3(a).

Using the trajectory of the particle, one can then construct the equilibrium probability density function $P_{eq}(\mathbf{r})$ of the position of the particle. We find that it does not depend on x and y, but only on the distance z between the particle and the wall. As seen in Fig. 3(b), an exponential tail is observed at large distance, which is identified to the sedimentation contribution in Perrin's experiment [2], but here with the probability density function of a single particle instead of the concentration field. In contrast, near the wall, we observe an abrupt depletion, indicating a repulsive electrostatic contribution. Indeed, when immersed in water, both the glass substrate and the polystyrene bead are negatively charged. All together, the total potential energy $U(z)$ thus reads

$$\frac{U(z)}{k_B T} = \begin{cases} B\, e^{-\frac{z}{\ell_D}} + \dfrac{z}{\ell_B}, & \text{for } z > 0 \\ +\infty, & \text{for } z \leqslant 0 \end{cases}, \qquad (1)$$

where k_B is the Boltzmann constant, B is a dimensionless number related to the surface electrostatic potentials of the particle and the wall [17], ℓ_D is the Debye length, $\ell_B = k_B T/(g\Delta m)$ is the Boltzmann length, g is the gravitational acceleration, and Δm is the (positive) buoyant mass of the particle. From this total potential energy, one can then construct the Gibbs-Boltzmann distribution $P_{eq}(z) = A \exp[-U(z)/(k_B T)]$ in position, where A is a normalization

constant, that fits the data very well, as shown in Fig. 3(b). Moreover, as shown in the inset of Fig. 3(b), we verified that we recover the Debye relation $\ell_D = 0.304/\sqrt{[\mathrm{NaCl}]}$, with ℓ_D in nm, and where [NaCl] is the concentration of salt in mol/L, with a prefactor corresponding to a single monovalent salt in water at room temperature [29]. Besides, we have verified (not shown) that the dimensionless parameter $B = 4.8$ related to surface charges is constant in the studied salt-concentration range, thus excluding any nonlinear effect [21,30] in our case.

We now turn to dynamical aspects, by considering the MSD. For the three spatial directions, indexed by $i = x, y$, and z, corresponding to the coordinates $r_x = x$, $r_y = y$, and $r_z = z$, of the position $\mathbf{r}$, and for a given time increment Δt, the MSD is defined as

$$\langle \Delta r_i(t)^2 \rangle_t = \langle [r_i(t + \Delta t) - r_i(t)]^2 \rangle_t, \tag{2}$$

where the average $\langle\rangle_t$ is performed over time t. For a free Brownian motion in the bulk, and in the absence of other forces than the dissipative and random ones, the MSD is linear in time, i.e., $\langle \Delta r_i(t)^2 \rangle_t = 2D_0 \Delta t$, where $D_0 = k_B T/(6\pi\eta a)$ is the bulk diffusion coefficient given by the Stokes-Einstein relation [1], and η is the liquid viscosity. Further including sedimentation restricts the validity of the previous result along z to short times only, i.e., for $\Delta t \ll \ell_B^2/D_0$ such that the vertical diffusion is not yet affected by the gravitational drift.

The presence of a rigid wall at $z = 0$ adds a repulsive electrostatic force along z. It also decreases the mobilities nearby through hydrodynamic interactions, leading to effective viscosities $\eta_\parallel(z) = \eta_x(z) = \eta_y(z)$, and $\eta_z(z)$. The latter are [31]

$$\eta_\parallel = \frac{\eta}{1 - \frac{9}{16}\xi + \frac{1}{8}\xi^3 - \frac{45}{256}\xi^4 - \frac{1}{16}\xi^5}, \tag{3}$$

where $\xi = a/(z + a)$, and

$$\eta_z = \eta \frac{6z^2 + 9az + 2a^2}{6z^2 + 2az}, \tag{4}$$

which is Padé approximated within 1% accuracy [32].

Interestingly, despite the previous modifications, the temporal linearity of the MSD is not altered by the presence of the wall [17,33] for x and y, as well as at short times for z. In such cases, the MSD reads

$$\langle \Delta r_i(t)^2 \rangle_t = 2\langle D_i \rangle \Delta t, \tag{5}$$

where for each spatial direction we introduced the local diffusion coefficient $D_i(z) = D_0\eta/\eta_i(z)$, and its average $\langle D_i(z)\rangle = \int_0^\infty dz\, D_i(z) P_{\mathrm{eq}}(z)$ against the Gibbs-Boltzmann distribution in position. As shown in Fig. 4(a), the MSD measured along x or y is indeed linear in time. By fitting to Eq. (5), using Eqs. (1) and (3), we extract an average transverse diffusion coefficient $\langle D_\parallel \rangle = \langle D_x \rangle = \langle D_y \rangle = 0.52\, D_0$. In contrast, along z, we identify two different regimes: one at short times, where the MSD is still linear in time, with a similarly obtained best-fit value of $\langle D_z \rangle = 0.24\, D_0$; and one at long times, where the MSD saturates to a plateau. This latter behavior indicates that the equilibrium regime has been reached, with the particle having essentially explored all the relevant positions given by the Gibbs-Boltzmann distribution.

Having focused on the MSD, i.e., on the second moment only, we now turn to the full probability density function

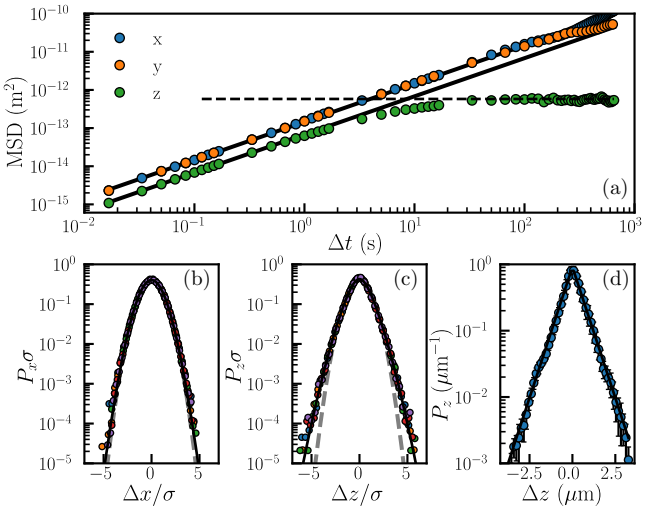


FIG. 4. (a) Measured mean-squared displacements [MSD, see Eq. (2)] as functions of the time increment Δt, for the three spatial directions, x, y, and z. The solid lines are best fits to Eq. (5), using Eqs. (1), (3), and (4), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm, providing the average diffusion coefficients $\langle D_\parallel \rangle = \langle D_x \rangle = \langle D_y \rangle = 0.52\, D_0$ and $\langle D_z \rangle = 0.24\, D_0$. The dashed line is the best fit to Eq. (8), using Eq. (1), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm. (b, c) Normalized probability density functions $P_i\sigma$ of the normalized displacements $\Delta x/\sigma$ and $\Delta z/\sigma$, at short times, with σ^2 the corresponding MSD [see panel (a)], for different time increments Δt ranging from 0.0167 to 0.083 s, as indicated with different colors. The solid lines are the best fits to Eq. (6), using Eqs. (1), (3), and (4), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm. For comparison, the gray dashed lines are normalized Gaussian distributions, with zero means and unit variances. (d) Probability density function P_z of the displacement Δz, at long times, averaged over several values of Δt ranging between 25 and 30 s. The solid line is the best fit to Eq. (7), using Eq. (1), with $B = 4.8$, $\ell_D = 21$ nm, and $\ell_B = 530$ nm.

P_i of the displacement Δr_i. Since, the diffusion coefficient $D_i(z)$ varies as a result of the variation of z along the particle trajectory, P_i exhibits a non-Gaussian behavior, as seen in Figs. 4(b)–4(d). We stress that we even resolve the onset of a non-Gaussian behavior in P_x, by zooming on the large-$|\Delta x|$ wings (not shown). At short times, P_i can be modeled by the averaged diffusion Green's function [16,34]:

$$P_i(\Delta r_i) = \int_0^\infty dz\, P_{\mathrm{eq}}(z) \frac{1}{\sqrt{4\pi D_i(z)\Delta t}} e^{-\frac{\Delta r_i^2}{4D_i(z)\Delta t}}, \tag{6}$$

against the Gibbs-Boltzmann distribution. As shown in Figs. 4(b) and 4(c), Eq. (6) captures the early data very well. At long times, Eq. (6) remains valid only for P_x and P_y. Nevertheless, the equilibrium regime being reached, P_z can eventually be written as

$$\lim_{\Delta t \to \infty} P_z(\Delta z) = \int_0^\infty dz\, P_{\mathrm{eq}}(z + \Delta z) P_{\mathrm{eq}}(z), \tag{7}$$

which contains in particular the second moment:

$$\lim_{\Delta t \to \infty} \langle \Delta z^2 \rangle = \int_{-\infty}^{+\infty} d\Delta z\, \Delta z^2 \int_0^\infty dz\, P_{\mathrm{eq}}(z + \Delta z) P_{\mathrm{eq}}(z). \tag{8}$$

As shown in Fig. 4(d), Eq. (8) captures the long-term data along z very well.

We now wish to go beyond the previous average $\langle D_i \rangle$ of Eq. (5), and resolve the local diffusion coefficient $D_i(z)$. To measure local viscosities from experimental trajectories, a binning method is generally employed [35]. Although this technique is well suited for drift measurements, it suffers from a lack of convergence and precision when second moments or local diffusion coefficients have to be extracted [23]. In particular, the binning method did not allow us to measure specifically the local diffusion coefficient in the key interfacial region corresponding to $z < 100$ nm. Additionally, Frishman and Ronceray have recently developed a robust numerical method using stochastic force inference, in order to evaluate spatially varying force fields and diffusion coefficients, from the information contained within the trajectories [23]. In practice, this is done by projecting the diffusion tensor onto a finite set of basis functions. We implemented this method, using fourth-order polynomials in our case. It allowed us to infer the local diffusion coefficients $D_i(z)$, down to $z = 10$ nm, as shown in Fig. 1(a). The results are in excellent agreement with the theoretical predictions, $D_\parallel(z) = D_0\eta/\eta_\parallel(z)$ and $D_z(z) = D_0\eta/\eta_z(z)$, using the effective viscosities of Eqs. (3) and (4), thus validating the method.

So far, through Figs. 1(a), 3(b), and 4, we have successively presented the various measured statistical quantities of interest, as well as their fits to corresponding theoretical models. Therein, we have essentially three free physical parameters, B, ℓ_B, and ℓ_D, describing the particle and its environment, as well as the *a priori* undetermined location of the $z = 0$ origin. These four parameters are actually redundant among the various theoretical models. Therefore, in order to measure them accurately, we in fact perform all the fits simultaneously, using a Broyden-Fletcher-Goldfarb-Shanno algorithm that is well suited for unconstrained nonlinear optimization [36]. To do so, we construct a global minimizer:

$$\chi^2 = \sum_{n=1}^{N} \chi_n^2, \tag{9}$$

where we introduce the minimizer χ_n^2 of each set n among the N sets of data, defined as

$$\chi_n^2 = \sum_{i=1}^{M_n} \frac{[y_{ni} - f_n(x_{ni}, \mathbf{b})]^2}{f_n(x_{ni}, \mathbf{b})^2}, \tag{10}$$

with $\{x_{ni}, y_{ni}\}$ the experimental data of set n, M_n the number of experimental data points for set n, f_n the model for set n, and $\mathbf{b} = (b_1, b_2, ..., b_p)$ the p free parameters. In our case, $p = 4$, and $\{x_{ni}, y_{ni}\}$ represent all the experimental data shown in Figs. 1(a), 3(b), and 4.

Due to strong dependence of the normal diffusion coefficient D_z with z, it is possible to find the wall position with a 10-nm resolution, thus overcoming a drawback of the Lorenz-Mie technique which only provides the axial distance relative to the focus of the objective lens. Additionally, the three physical parameters globally extracted from the multifitting procedure are $B = 4.8 \pm 0.6$, $\ell_D = 21 \pm 1$ nm, and $\ell_B = 530 \pm 2$ nm. Using the particle radius $a = 1.518 \pm 0.006\,\mu$m calibrated from the preliminary fits of the interference patterns to the Lorenz-Mie scattering function [see Figs. 2(c) and 2(d)], and the 1050 kg m^{-3} tabulated bulk density of polystyrene, we would have expected $\ell_B = 559$ nm instead, which corresponds to less than 2% error, and might be attributed to nanometric offsets, such as, e.g., the particle and/or wall rugosities.

Finally, we investigate the total conservative force $F_z(z)$ acting on the particle along z. By averaging the overdamped Langevin equation over a fine-enough z-binning grid and short enough time interval Δt, one gets the Itō convention (corresponding to our definition of Δz):

$$F_z(z) = 6\pi\eta_z(z)a\frac{\langle \Delta z \rangle}{\Delta t} - k_B T \frac{D_z'(z)}{D_z(z)}, \tag{11}$$

where the last term corresponds to the additional contribution due to the nontrivial integration of the multiplicative noise [20,37–39], with the prime denoting the derivative with respect to z. From the averaged measured vertical drifts $\langle \Delta z \rangle$, and invoking Eq. (4), one can reconstruct $F_z(z)$ from Eq. (11), as shown in Fig. 1(b). We stress that the statistical error on the force measurement is comparable to the thermal-noise limit [40]:

$$\Delta F = \sqrt{24\pi k_B T \eta_z(z) a/\tau_{box}(z)}, \tag{12}$$

where $\tau_{box}(z)$ is the total time spent by the particle in the corresponding box of the z-binning grid. To corroborate these measurements, we invoke Eq. (1) and express the total conservative force $F_z(z) = -U'(z)$ acting on the particle along z:

$$F_z(z) = k_B T \left(\frac{B}{\ell_D} e^{-\frac{z}{\ell_D}} - \frac{1}{\ell_B} \right). \tag{13}$$

Using the physical parameters extracted from the above multifitting procedure, we plot Eq. (13) in Fig. 1(b). The agreement with the data is excellent, thus showing the robustness of the force measurement. In particular, we can measure forces down to a distance of 40 nm from the surface. Additionally, far from the wall, we are able to resolve the actual buoyant weight $F_g = -7 \pm 4$ fN of the particle. This demonstrates that we reach the femtonewton resolution, and that this resolution is solely limited by thermal noise.

To conclude, we have successfully built a multiscale statistical analysis for the problem of freely diffusing individual colloids near a rigid wall. Combining the equilibrium distribution in position, time-dependent non-Gaussian statistics for the spatial displacements, a method to infer local diffusion coefficients, and a multifitting procedure allowed us to reduce drastically the measurement uncertainties and reach the nanoscale and thermal-noise-limited femtonewton spatial and force resolutions, respectively. The ability to measure tiny surface forces, locally, and at equilibrium, as well as the possible extension of the method to nonconservative forces and out-of-equilibrium settings [41,42], opens fascinating perspectives for nanophysics and biophysics.

We thank Elodie Millan, Louis Bellando de Castro, Julien Burgin, Bernard Trégon, Abdelhamid Maali, David Dean, and Mathias Perrin for interesting discussions. We acknowledge funding from the Bordeaux IdEx program LAPHIA (Grant No. ANR-10IDEX-03-02), Arts et Science (Sonotact

Grant No. 2017-2018), Région Nouvelle Aquitaine (Grant No. 2018-1R50304), and from the Agence Nationale de la Recherche (Grant No. ANR-21-ERCC-0010-01 EMet-Brown).

[1] A. Einstein, Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen, Ann. Phys. **322**, 549 (1905).
[2] J. Perrin, *Les Atomes* (Flammarion, Paris, 2014).
[3] B. U. Felderhof, Effect of the wall on the velocity autocorrelation function and long-time tail of Brownian motion, J. Phys. Chem. B **109**, 21406 (2005).
[4] B. Wang, S. M. Anthony, S. C. Bae, and S. Granick, Anomalous yet Brownian, Proc. Natl. Acad. Sci. USA **106**, 15160 (2009).
[5] A. V. Chechkin, F. Seno, R. Metzler, and I. M. Sokolov, Brownian yet Non-Gaussian Diffusion: From Superstatistics to Subordination of Diffusing Diffusivities, Phys. Rev. X **7**, 021002 (2017).
[6] C. I. Bouzigues, P. Tabeling, and L. Bocquet, Nanofluidics in the Debye Layer at Hydrophilic and Hydrophobic Surfaces, Phys. Rev. Lett. **101**, 114503 (2008).
[7] L. Joly, C. Ybert, and L. Bocquet, Probing the Nanohydrodynamics at Liquid-Solid Interfaces Using Thermal Motion, Phys. Rev. Lett. **96**, 046101 (2006).
[8] J. Mo, A. Simha, and M. G. Raizen, Brownian motion as a new probe of wettability, J. Chem. Phys. **146**, 134707 (2017).
[9] E. R. Dufresne, T. M. Squires, M. P. Brenner, and D. G. Grier, Hydrodynamic Coupling of Two Brownian Spheres to a Planar Surface, Phys. Rev. Lett. **85**, 3317 (2000).
[10] L. P. Faucheux and A. J. Libchaber, Confined Brownian motion, Phys. Rev. E **49**, 5158 (1994).
[11] E. R. Dufresne, D. Altman, and D. G. Grier, Brownian dynamics of a sphere between parallel walls, Europhys. Lett. **53**, 264 (2001).
[12] R. Lopez-Fernandez, M. D. Carbajal-Tinoco, and J. L. Arauz-Lara, Asymmetry in Colloidal Diffusion Near a Rigid Wall, Phys. Rev. Lett. **99**, 138303 (2007).
[13] H. B. Eral, J. M. Oh, D. van den Ende, F. Mugele, and M. H. G. Duits, Anisotropic and hindered diffusion of colloidal particles in a closed cylinder, Langmuir **26**, 16722 (2010).
[14] P. Sharma, S. Ghosh, and S. Bhattacharya, A high-precision study of hindered diffusion near a wall, Appl. Phys. Lett. **97**, 104101 (2010).
[15] J. Mo, A. Simha, and M. G. Raizen, Broadband boundary effects on Brownian motion, Phys. Rev. E **92**, 062106 (2015).
[16] M. Matse, M. V. Chubynsky, and J. Bechhoefer, Test of the diffusing-diffusivity mechanism using near-wall colloidal dynamics, Phys. Rev. E **96**, 042604 (2017).
[17] D. C. Prieve, Measurement of colloidal forces with TIRM, Adv. Colloid Interface Sci. **82**, 93 (1999).
[18] A. Banerjee and K. D. Kihm, Experimental verification of near-wall hindered diffusion for the Brownian motion of nanoparticles using evanescent wave microscopy, Phys. Rev. E **72**, 042101 (2005).
[19] S. K. Sainis, V. Germain, and E. R. Dufresne, Statistics of Particle Trajectories at Short Time Intervals Reveal fN-Scale Colloidal Forces, Phys. Rev. Lett. **99**, 018303 (2007).
[20] G. Volpe, L. Helden, T. Brettschneider, J. Wehr, and C. Bechinger, Influence of Noise on Force Measurements, Phys. Rev. Lett. **104**, 170602 (2010).
[21] W. Wang, J. S. Guasto, and P. Huang, Measurement bias in evanescent wave nano-velocimetry due to tracer size variations, Exp. Fluids **51**, 1685 (2011).
[22] M. Li, O. Sentissi, S. Azzini, G. Schnoering, A. Canaguier-Durand, and C. Genet, Subfemtonewton force fields measured with ergodic Brownian ensembles, Phys. Rev. A **100**, 063816 (2019).
[23] A. Frishman and P. Ronceray, Learning Force Fields from Stochastic Trajectories, Phys. Rev. X **10**, 021009 (2020).
[24] C. F. Bohren and D. R. Huffman, *Absorption and Scattering of Light by Small Particles* (Wiley, New York, 1998).
[25] M. Mishchenko, *Scattering, Absorption, and Emission of Light by Small Particles* (Cambridge University, Cambridge, England, 2002).
[26] S.-H. Lee, Y. Roichman, G.-R. Yi, S.-H. Kim, S.-M. Yang, A. van Blaaderen, P. van Oostrum, and D. G. Grier, Characterizing and tracking single colloidal particles with video holographic microscopy, Opt. Express **15**, 18275 (2007).
[27] P. D. J. van Oostrum, Using light scattering to track, characterize and manipulate colloids, Ph.D. thesis, University of Amsterdam, 2011.
[28] B. J. Krishnatreya, Precision measurements of colloidal interactions and dynamics, Ph.D. thesis, New York University, 2014.
[29] J. Israelachvili, *Intermolecular and Surface Forces* (Academic, New York, 2011).
[30] M. R. Oberholzer, N. J. Wagner, and A. M. Lenhoff, Grand canonical Brownian dynamics simulation of colloidal adsorption, J. Chem. Phys. **107**, 9157 (1997).
[31] H. Brenner, The slow motion of a sphere through a viscous fluid towards a plane surface, Chem. Eng. Sci. **16**, 242 (1961).
[32] M. A. Bevan and D. C. Prieve, Hindered diffusion of colloidal particles very near to a wall: Revisited, J. Chem. Phys. **113**, 1228 (2000).
[33] M. V. Chubynsky and G. W. Slater, Diffusing Diffusivity: A Model for Anomalous, yet Brownian, Diffusion, Phys. Rev. Lett. **113**, 098302 (2014).
[34] S. Hapca, J. W. Crawford, and I. M. Young, Anomalous diffusion of heterogeneous populations characterized by normal diffusion at the individual level, J. R. Soc., Interface **6**, 111 (2009).
[35] R. Friedrich, J. Peinke, M. Sahimi, and M. Reza Rahimi Tabar, Approaching complexity by stochastic methods: From biological systems to turbulence, Phys. Rep. **506**, 87 (2011).
[36] Y.-H. Dai, Convergence properties of the BFGS algorithm, SIAM J. Optim. **13**, 693 (2002).
[37] R. Mannella and P. V. E. McClintock, Comment on "Influence of Noise on Force Measurements," Phys. Rev. Lett. **107**, 078901 (2011).
[38] G. Volpe, L. Helden, T. Brettschneider, Jan Wehr, and C. Bechinger, Reply to Mannella and McClintock, Phys. Rev. Lett. **107**, 078902 (2011).

[39] R. Mannella and P. V. E. McClintock, Ito versus Stratonovich: 30 years later, Fluct. Noise Lett. **11**, 1240010 (2012).

[40] L. Liu, S. Kheifets, V. Ginis, and F. Capasso, Subfemtonewton Force Spectroscopy at the Thermal Limit in Liquids, Phys. Rev. Lett. **116**, 228001 (2016).

[41] Y. Amarouchene, M. Mangeat, B. V. Montes, L. Ondic, T. Guérin, D. S. Dean, and Y. Louyer, Nonequilibrium Dynamics Induced by Scattering Forces for Optically Trapped Nanoparticles in Strongly Inertial Regimes, Phys. Rev. Lett. **122**, 183901 (2019).

[42] M. Mangeat, Y. Amarouchene, Y. Louyer, T. Guérin, and D. S. Dean, Role of nonconservative scattering forces and damping on Brownian particles in optical traps, Phys. Rev. E **99**, 052107 (2019).

Data Analysis procedure

[1]:
```
%load_ext lab_black
# Import important tools
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from scipy.io import loadmat, savemat
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import curve_fit, minimize, least_squares
from scipy.integrate import trapz
from scipy.stats import norm, kurtosis
from matplotlib.ticker import ScalarFormatter
from matplotlib import rc
```

[ ]:
```
```

[2]:
```
mpl.rcParams["xtick.direction"] = "in"
mpl.rcParams["ytick.direction"] = "in"
mpl.rcParams["lines.markeredgecolor"] = "k"
mpl.rcParams["lines.markeredgewidth"] = 1
mpl.rcParams["figure.dpi"] = 130
rc("font", family="serif")
rc("text", usetex=True)
rc("xtick", labelsize="x-small")
rc("ytick", labelsize="x-small")


def cm2inch(value):
    return value / 2.54
```

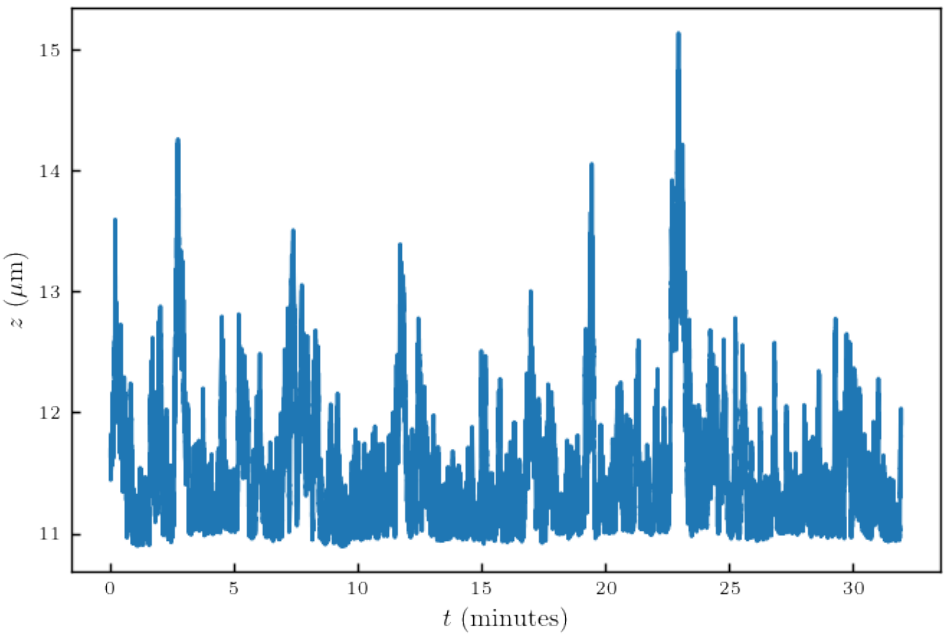
We load the data

[3]:
```
raw_data = loadmat(
    "fit_result_dur_27052020_n_r_fix_0p0513_wav_532_r_1p516_n_1.597.mat"
)["data"][:, 0:3]
r = 1.516 * 1e-6
n_part = 1.597
fps = 60
time = np.arange(0, np.shape(raw_data)[0]) / fps
```

```
dataset = {}
dataset["r"] = r
dataset["n"] = n_part
dataset["fps"] = fps
dataset["time"] = time
```

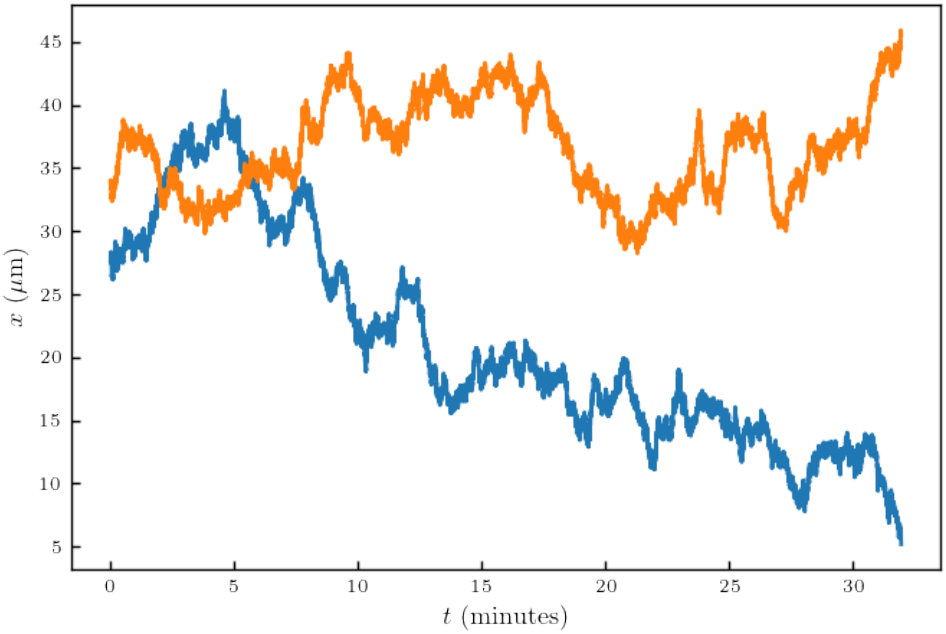
1 Data exploration

[4]:
```
# We put everything in microns
raw_data_m = raw_data
raw_data_m[:, 0:3] = raw_data_m[:, 0:3] * 0.0513
plt.plot(time / fps, raw_data_m[:, 2])
x = raw_data_m[:, 0]
y = raw_data_m[:, 1]
z = raw_data_m[:, 2]

plt.xlabel("$t$ (minutes)")
plt.ylabel("$z$ ($\mathrm{\mu m}$)")
plt.show()
```



[5]:
```
plt.plot(time / fps, raw_data_m[:, 0], label="x")
plt.plot(time / fps, raw_data_m[:, 1], label="y")
plt.xlabel("$t$ (minutes)")
plt.ylabel("$x$ ($\mathrm{\mu m}$)")
```

```
plt.show()
```



2 MSD

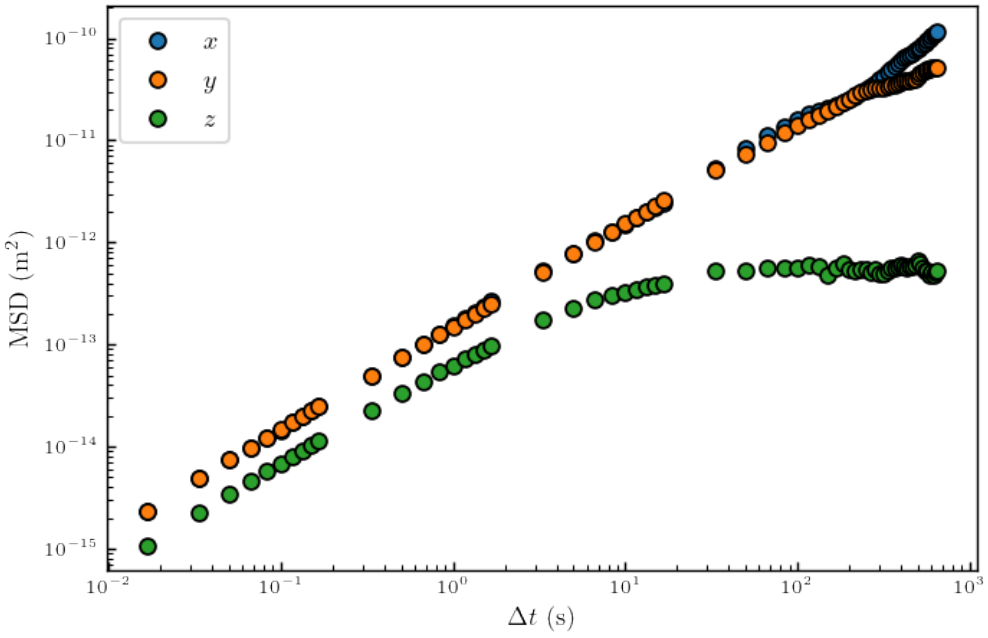
We compute the MSD using the formula:

$$\langle \Delta r_i(t)^2 \rangle_t = \langle [r_i(t + \Delta t) - r_i(t)]^2 \rangle_t \, . \tag{1}$$

[6]:
```
def MSD(x, t):
    MSD = np.zeros(len(t))
    for n, i in enumerate(t):
        MSD[n] = np.nanmean((x[0:-i] - x[i:]) ** 2)
    return MSD
```

[7]:
```
t = np.array(
    [
        *np.arange(1, 10, 1),
        *np.arange(10, 100, 10),
        *np.arange(100, 1000, 100),
        *np.arange(1000, 40000, 1000),
    ]
)
MSD_x = MSD(x * 1e-6, t)  # m² conversion
MSD_y = MSD(y * 1e-6, t)
```

```
MSD_z = MSD(z * 1e-6, t)


plt.loglog(time[t], MSD_x, "o", label="$x$")
plt.plot(time[t], MSD_y, "o", label="$y$")
plt.plot(time[t], MSD_z, "o", label="$z$")
plt.ylabel("MSD ($\mathrm{m^2}$)")
plt.xlabel("$\Delta t$ (s)")

plt.legend()

dataset["MSD_x_tot"] = MSD_x
dataset["MSD_y_tot"] = MSD_y
dataset["MSD_z_tot"] = MSD_z
dataset["MSD_time_tot"] = time[t]
```



We fit the short time MSD with and average diffusion coefficient such as:

$$\langle \Delta r_i(t)^2 \rangle_t = 2\langle D_i \rangle \Delta t \,, \tag{2}$$

[8]:
```
Do = 4e-21 / (6 * np.pi * 0.001 * r)
f = lambda x, a, noiselevel: 2 * Do * a * x + (noiselevel * 1e-9) ** 2
popt_1, pcov_1 = curve_fit(f, time[t[0:5]], MSD_x[0:5], p0=[1, 30])
popt_2, pcov_1 = curve_fit(f, time[t[0:5]], MSD_y[0:5], p0=[1, 30])
popt_3, pcov_1 = curve_fit(f, time[t[0:5]], MSD_z[0:5], p0=[1, 30])
```

```
dataset["x_MSD_fit"] = time[t[0:5]]

dataset["MSD_x"] = MSD_x[0:5]
dataset["MSD_y"] = MSD_y[0:5]
dataset["MSD_z"] = MSD_z[0:5]
```

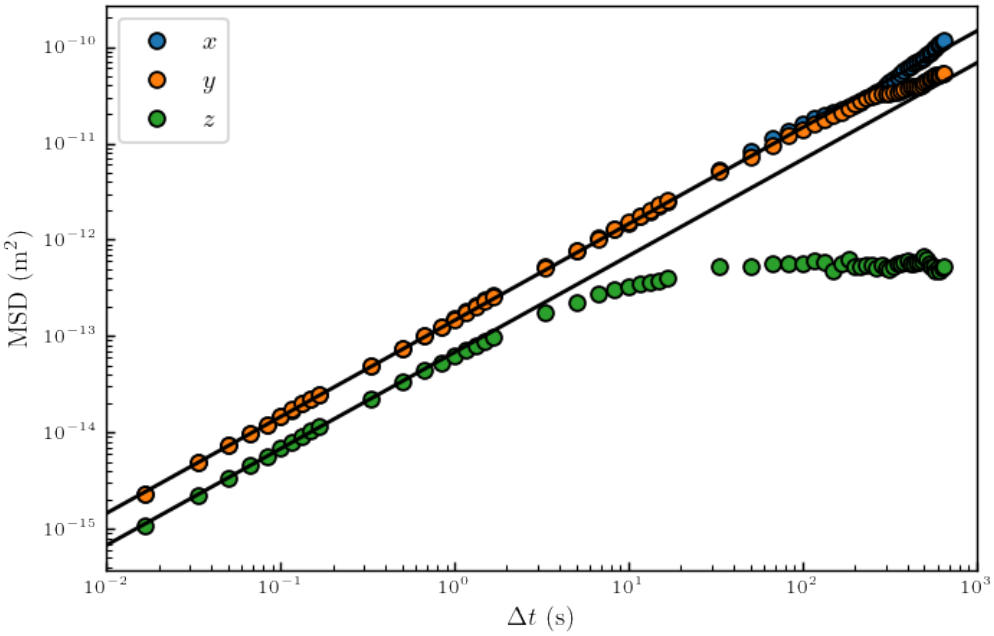
```
C:\Users\m.lavaud\.conda\envs\analyse\lib\site-
packages\scipy\optimize\minpack.py:828: OptimizeWarning: Covariance of the
parameters could not be estimated
  warnings.warn('Covariance of the parameters could not be estimated',
```

[9]:
```
print(
    "We measure a reduced mean diffusion coefficient of {:.3f} for the␣
 ↪perpendicular motion and of {:.3f} for the parallel motion".format(
        (popt_1[0] + popt_2[0]) / 2, popt_3[0]
    )
)
```

```
We measure a reduced mean diffusion coefficient of 0.522 for the perpendicular
motion and of 0.243 for the parallel motion
```

[10]:
```
plt.loglog(time[t], MSD_x, "o", label="$x$")
plt.plot(time[t], MSD_y, "o", label="$y$")
plt.plot(time[t], MSD_z, "o", label="$z$")
plt.ylabel("MSD ($\mathrm{m^2}$)")
plt.xlabel("$\Delta t$ (s)")
tt = np.linspace(1e-2, 1e3)
plt.plot(tt, f(tt, *popt_1), color="k")
plt.plot(tt, f(tt, *popt_3), color="k")

plt.xlim((1e-2, 1e3))
plt.legend()
```

[10]: <matplotlib.legend.Legend at 0x27a80c0b700>



3 Displacement distributions

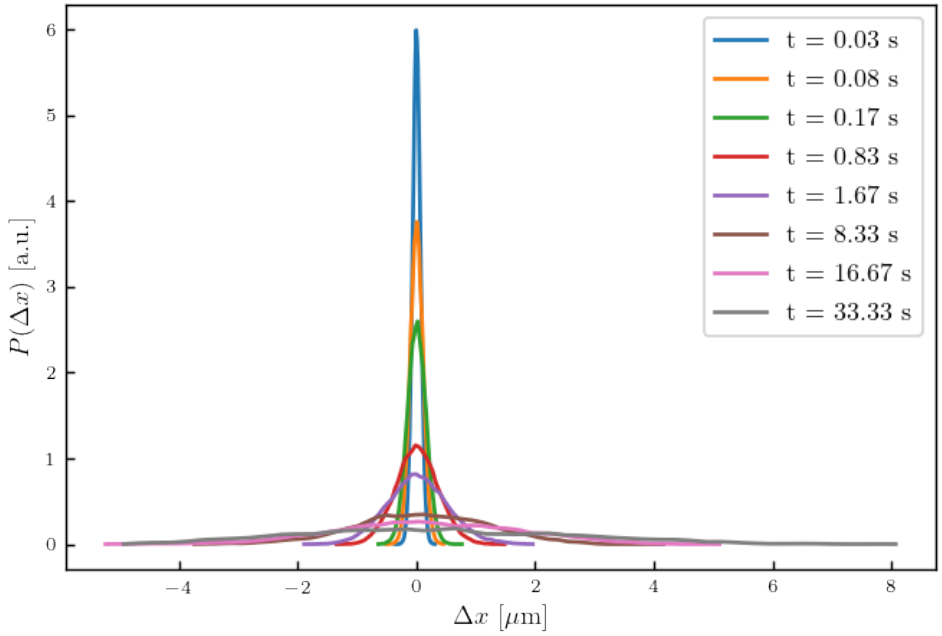
3.1 Δx distributions

[11]:
```
def pdf(data, bins=10, density=True):
    """
    function to automatize the computations of experimental probability density␣
 ↪functions.
    """

    pdf, bins_edge = np.histogram(data, bins=bins, density=density)
    bins_center = (bins_edge[0:-1] + bins_edge[1:]) / 2

    return pdf, bins_center
```

[12]:
```
I = [2, 5, 10, 50, 100, 500, 1000, 2000]

for i in I:

    Dezs = x[0:-i] - x[i:]
    hist, bins_center = pdf(Dezs, bins=50)

    plt.plot(bins_center, hist, label=" t = {:.2f} s".format(time[i]))
```

```
plt.legend()
plt.ylabel("$P(\Delta x)$ [a.u.]")
plt.xlabel("$\Delta x$ [$\mathrm{\mu m}$]")
```

[12]: `Text(0.5, 0, '$\\Delta x$ [$\\mathrm{\\mu m}$]')`



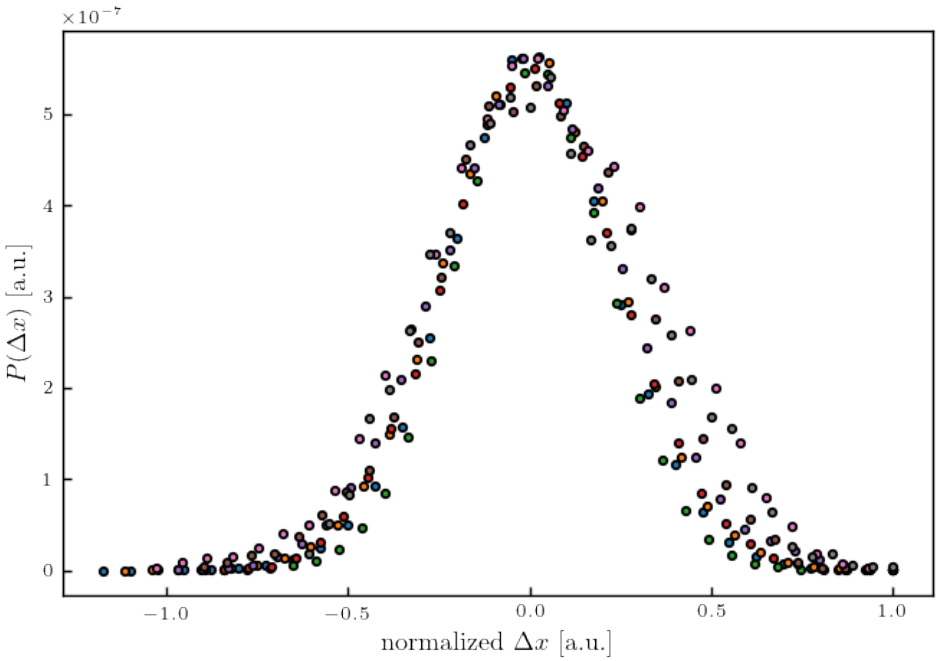
If we now normalize by the standard deviation

[13]:
```
def gauss_function(x, a, x0, sigma):
    return a * np.exp(-((x - x0) ** 2) / (2 * sigma ** 2))
```

[14]:
```
for n, i in enumerate(I):

    Dezs = x[0:-i] - x[i:]
    Dezs = Dezs / np.sqrt(2 * Do * time[i])
    hist, bins_center = pdf(Dezs, bins=30)

    # if i == I[0]:
    #     popt, pcov = curve_fit(gauss_function, bins_center/np.max(bins_center),␣
↪hist, p0 = [1, np.mean(hist), np.std(hist)])
    #     plt.plot(bins_center/np.max(bins_center), gauss_function(bins_center,␣
↪*popt), label = "fit at t = {:.2f} s".format(time[i]))
    #     plt.plot(bins_center/np.max(bins_center), hist, "x",label = " t = {:.
↪2f} s".format(time[i]),color = "tab:blue")
    #     continue
```

```
    plt.plot(
        bins_center / np.max(bins_center),
        hist,
        ".",
        label=" $Delta$t = {:.2f} s".format(time[i]),
    )


plt.ylabel("$P(\Delta x)$ [a.u.]")
plt.xlabel("normalized $\Delta x$ [a.u.]")
```

[14]: `Text(0.5, 0, 'normalized $\\Delta x$ [a.u.]')`



[15]:
```
(3.5e-22) ** (1 / 3)
```

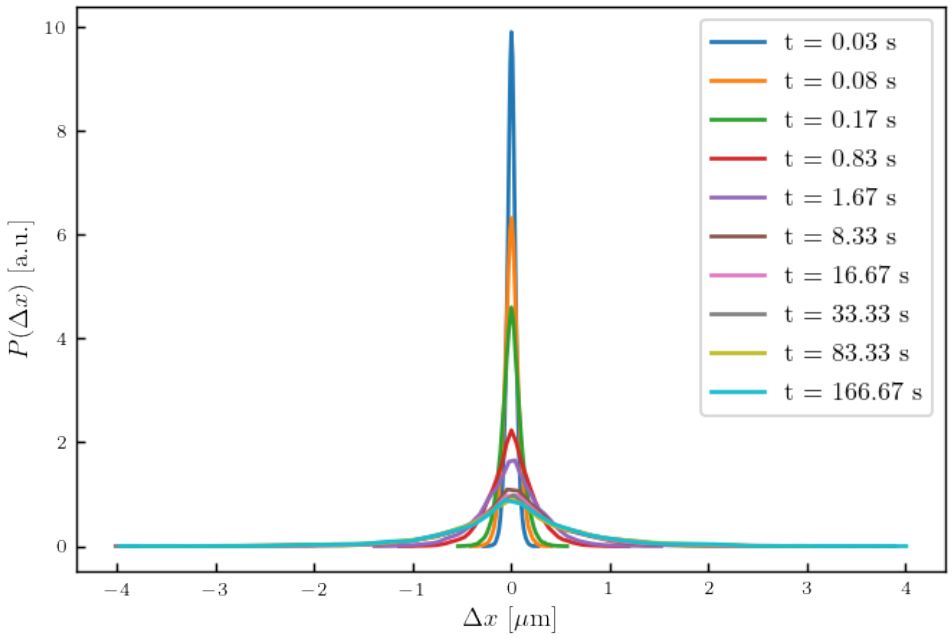
[15]: `7.047298732064899e-08`

We can see a clear change but we would need to average on different trajectectories to have consitant results.

3.2 Δz distributions

[16]:
```
I = [2, 5, 10, 50, 100, 500, 1000, 2000, 5000, 10000]

for i in I:

    Dezs = z[0:-i] - z[i:]
    hist, bins_center = pdf(Dezs[~np.isnan(Dezs)], bins=50)

    plt.plot(bins_center, hist, label=" t = {:.2f} s".format(time[i]))

plt.legend()
plt.ylabel("$P(\Delta x)$ [a.u.]")
plt.xlabel("$\Delta x$ [$\mathrm{\mu m}$]")
```

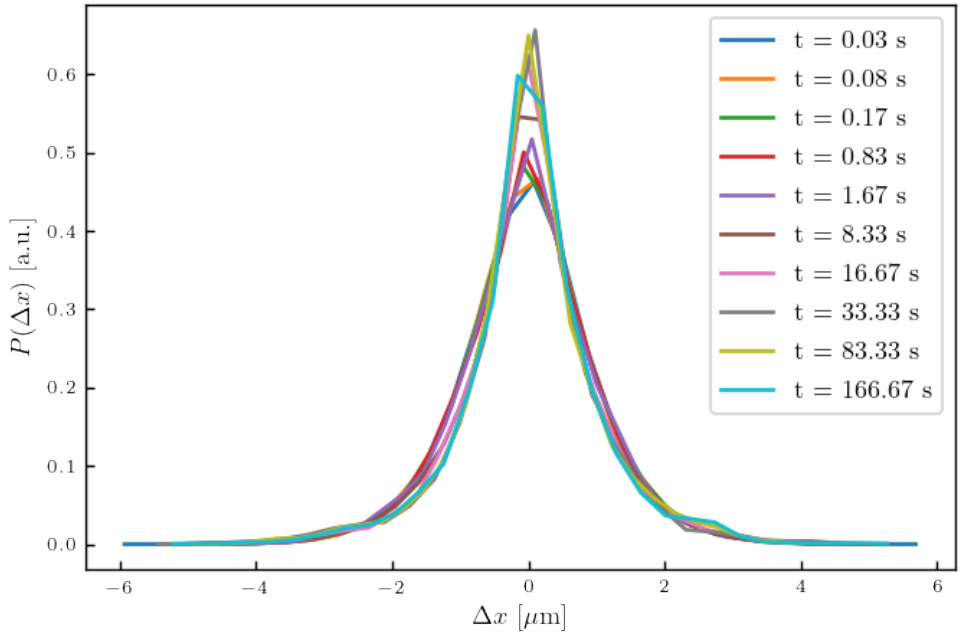
[16]:
```
Text(0.5, 0, '$\\Delta x$ [$\\mathrm{\\mu m}$]')
```



[17]:
```
for i in I:

    Dezs = z[0:-i] - z[i:]
    Dezs = Dezs / np.nanstd(Dezs)
    hist, bins_center = pdf(Dezs[~np.isnan(Dezs)], bins=30)

    plt.plot(bins_center, hist, label=" t = {:.2f} s".format(time[i]))
```

```
plt.legend()
plt.ylabel("$P(\Delta x)$ [a.u.]")
plt.xlabel("$\Delta x$ [$\mathrm{\mu m}$]")
```

[17]: `Text(0.5, 0, '$\\Delta x$ [$\\mathrm{\\mu m}$]')`

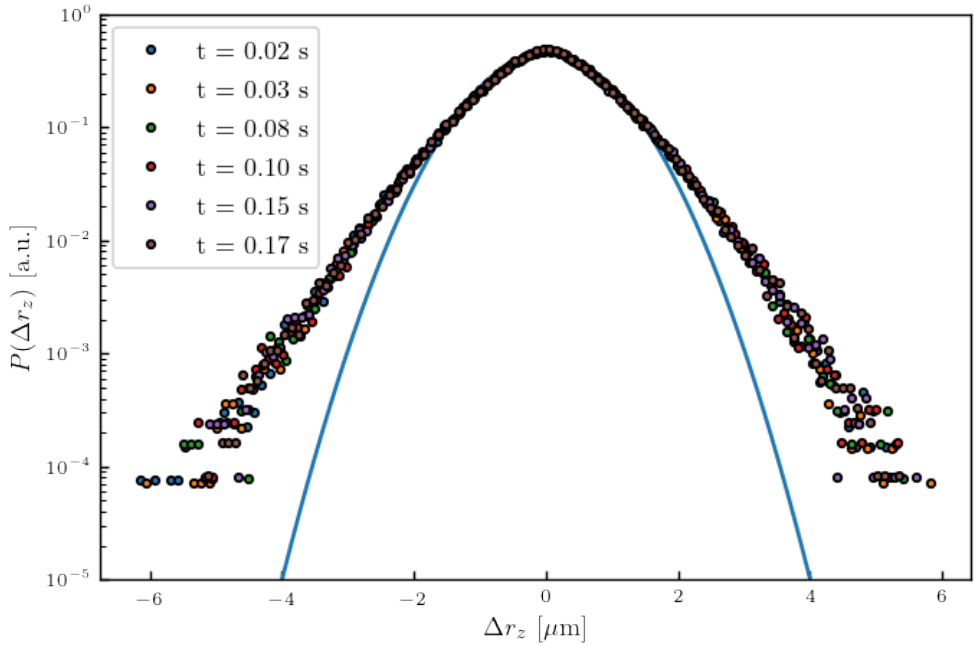


3.2.1 Short time distributions

[18]:
```
I = [1, 2, 5, 6, 9, 10]

for i in I:

    Dezs = z[0:-i] - z[i:]
    Dezs = Dezs / np.std(Dezs)
    hist, bins_center = pdf(Dezs[~np.isnan(Dezs)], bins=100)

    if i == I[0]:
        popt, pcov = curve_fit(
            gauss_function, bins_center, hist, p0=[1, np.mean(hist), np.
 ↪std(hist)]
        )
        plt.plot(bins_center, gauss_function(bins_center, *popt))
        plt.plot(
            bins_center,
            hist,
```

```
            ".",
            label=" t = {:.2f} s".format(time[i]),
            color="tab:blue",
        )
        continue
    plt.semilogy(bins_center, hist, ".", label=" t = {:.2f} s".format(time[i]))

plt.legend()
plt.ylabel("$P(\Delta r_z)$ [a.u.]")
plt.xlabel("$\Delta r_z$ [$\mathrm{\mu m}$]")
axes = plt.gca()
axes.set_ylim([1e-5, 1])
```

[18]: (1e-05, 1)



The non-Gaussianity is due to the hindered mobility. Taking into account the hindered mobility the PDF of displacement writes:

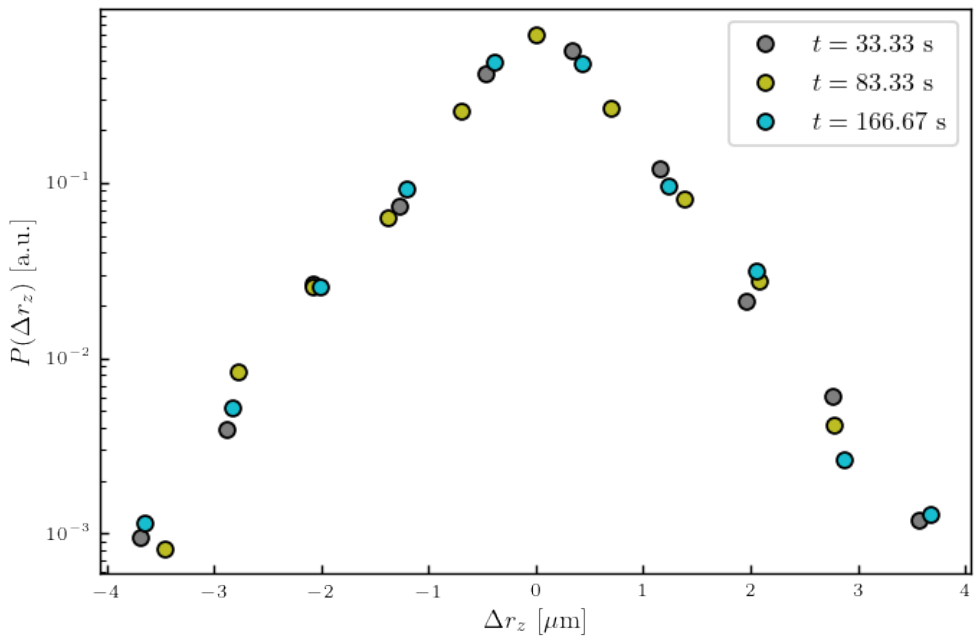
$$P(\Delta r_i, \Delta t) = \int_0^\infty \mathrm{d}D P(D_i) \frac{1}{\sqrt{4\pi D_i \Delta t}} \exp\left[\frac{-\Delta_i r_i^2}{4 D_i \Delta t}\right] . \tag{3}$$

This non-Gaussianity can be fitted as done at the end of this appendix and shown in the manuscript.

3.3 Long time distributions

[19]:
```
I = [2000, 5000, 10000]

color_long_time = ["tab:gray", "tab:olive", "tab:cyan"]
for n, i in enumerate(I):

    Dezs = z[0:-i] - z[i:]
    hist, bins_center = pdf(Dezs[~np.isnan(Dezs)], bins=10)

    plt.semilogy(
        bins_center,
        hist,
        "o",
        label=" $t = {:.2f}$ s".format(time[i]),
        color=color_long_time[n],
    )

plt.legend()

plt.ylabel("$P(\Delta r_z)$ [a.u.]")
plt.xlabel("$\Delta r_z$ [$\mathrm{\mu m}$]")
```

[19]:
```
Text(0.5, 0, '$\\Delta r_z$ [$\\mathrm{\\mu m}$]')
```



Indeed at long time it becomes exponential and it's no longer dependent on Δt At very long time

intervals Δt each position measurement can be seen as random measurement on the Boltzmann distribution. Thus, one can write the probability distribution as a convolution of two PDF:

$$P(\Delta z) = \int_{-\infty}^{\infty} dz P_B(z) P_B(z + \Delta z), \tag{4}$$

with :

$$P_B(z) = Ae^{\left(Bexp\left(-\frac{z}{l_d}\right) - \frac{z}{l_b}\right)} \tag{5}$$

Also, $P_B(z < 0)$ giving at long time step :

$$P(\Delta z) = A' exp\left[Bexp\left[-\frac{z}{l_d}\right](1 + exp[-\frac{\Delta z}{l_d}]) - \frac{2z + \Delta z}{l_b}\right] \tag{6}$$

3.4 Analysis of pdf of the Δz at large time step

To have a better measurement we average the PDF of displacement Δr_z over different time-step Δt . But, first of all, we need to get rid of the drifts at long time. We do that by taking a moving minimum.

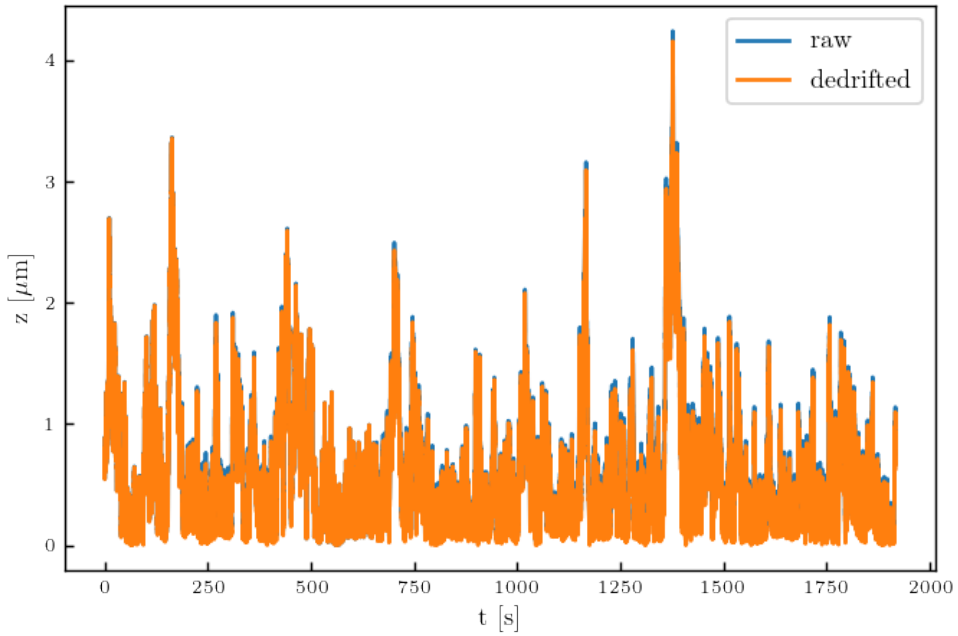
3.5 Dedrifting the z trajectory

[20]:
```
def movmin(datas, k):
    result = np.empty_like(datas)
    start_pt = 0
    end_pt = int(np.ceil(k / 2))

    for i in range(len(datas)):
        if i < int(np.ceil(k / 2)):
            start_pt = 0
        if i > len(datas) - int(np.ceil(k / 2)):
            end_pt = len(datas)
        result[i] = np.min(datas[start_pt:end_pt])
        start_pt += 1
        end_pt += 1

    return result
```

[21]:
```
z_dedrift = z - movmin(z, 10000)
```

[22]:
```
# Fig for comparing the two

plt.plot(time, z - np.min(z), label="raw")
plt.plot(time, z_dedrift, label="dedrifted")
plt.legend()

plt.xlabel("t [s]")
plt.ylabel("z [$\mathrm{\mu m}$]")
```

[22]:
```
Text(0, 0.5, 'z [$\\mathrm{\\mu m}$]')
```

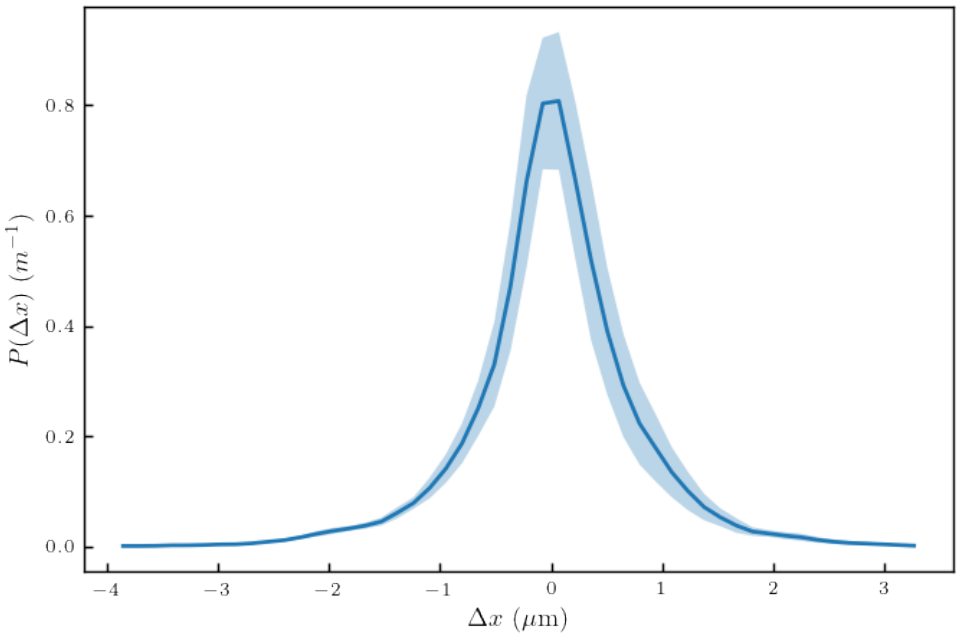


3.5.1 Measuring pdf at large Δt with the dedrifted trajectory and analysing it

[23]:
```
t_start = 25
t_end = 30
I = np.arange(t_start * fps, t_end * fps)
bins = 50

hists = np.zeros((bins, len(I)))
bins_centers = np.zeros((bins, len(I)))

for n, i in enumerate(I):

    Dezs = z_dedrift[0:-i] - z_dedrift[i:]
    hist, bins_center = pdf(Dezs[~np.isnan(Dezs)], bins=bins)

    hists[:, n] = hist
    bins_centers[:, n] = bins_center


pdf_long_t = np.mean(hists, axis=1)
bins_centers_long_t = np.mean(bins_centers, axis=1)
err_long_t = np.std(hists, axis=1)
err_bins_centers = np.std(bins_centers, axis=1)
```

[24]:
```
plt.plot(bins_centers_long_t, pdf_long_t)
plt.fill_between(
    bins_centers_long_t, pdf_long_t - err_long_t, pdf_long_t + err_long_t,␣
 ↪alpha=0.3
)
plt.ylabel("$P(\Delta x)$ ($m^{-1}$)")
plt.xlabel("$\Delta x$ ($\mathrm{\mu m}$)")
```

[24]:
```
Text(0.5, 0, '$\\Delta x$ ($\\mathrm{\\mu m}$)')
```



We are now going to code the function

$$P(\Delta z) = \int_{-\infty}^{\infty} A' exp\left[Bexp\left[-\frac{z}{l_d}\right](1 + exp[-\frac{\Delta z}{l_d}]) - \frac{2z + \Delta z}{l_b}\right] \tag{7}$$

Noting that coding the form :

$$P(\Delta z) = \int_{-\infty}^{\infty} dz P_B(z) P_B(z + \Delta z), \tag{8}$$

Will be easier and P_B will be reused later on. Also since $P_B(z < 0) = 0$:

$$P(\Delta z) = \int_{0}^{\infty} dz P_B(z) P_B(z + \Delta z), \tag{9}$$

with :

$$P_B(z) = Ae^{\left(Bexp\left(-\frac{z}{l_d}\right) - \frac{z}{l_b}\right)} \tag{10}$$

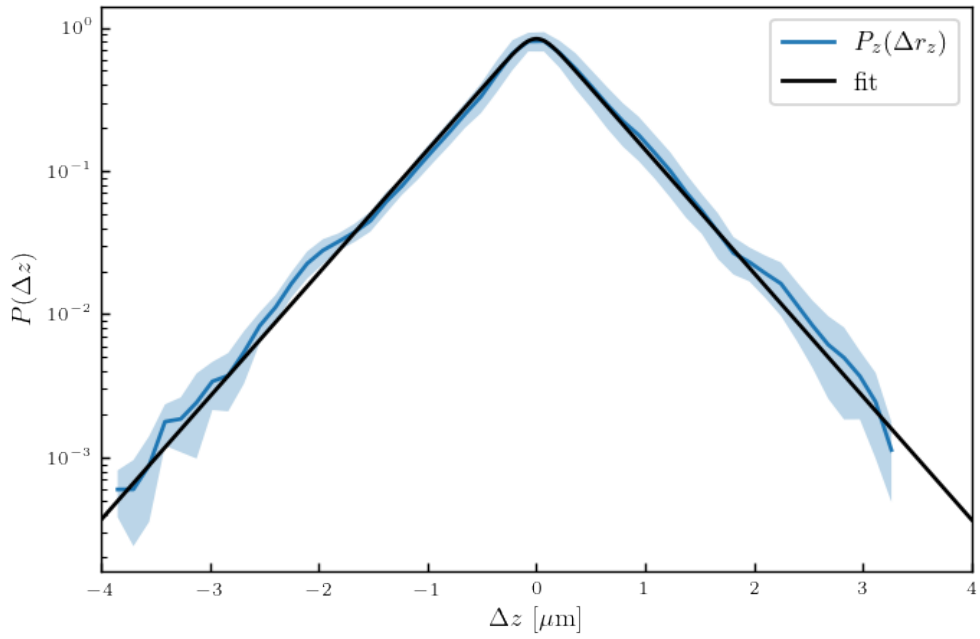
[25]:
```
def P_b(z, A, B, ld, lb):
    P_b = A * np.exp(-B * np.exp(-z / (ld)) - z / lb)
    P_b[z < 0] = 0
    return P_b


def dPdeltaz_long(z, DZ, A, B, ld, lb):
    return P_b(z, A, B, ld, lb) * P_b(z + DZ, A, B, ld, lb)


def P_computation(DZ, A, B, ld, lb):
    z = np.linspace(0, 20e-6, 1000)
    dP = dPdeltaz_long(z, DZ, A, B, ld, lb)
    P = trapz(dP, z)
    return P


def Pdeltaz_long(DZ, B, ld, lb):
    if type(DZ) == float:
        return P_computation(i, 1, B, ld, lb)

    pdf = np.array([P_computation(i, 1, B, ld * 1e-9, lb * 1e-9) for i in DZ])

    # normalisation of the PDF to not use A

    A = trapz(pdf, DZ * 1e6)

    return np.array([P_computation(i, 1, B, ld * 1e-9, lb * 1e-9) for i in DZ]) /
 ↪ A
```

[26]:
```
A = 0.14e8
B = 4
ld = 70
lb = 500
p1 = [B, ld, lb]

# Normalisation fo the pdf

pdf_long_t = pdf_long_t / trapz(pdf_long_t, bins_centers_long_t)


popt, pcov = curve_fit(Pdeltaz_long, bins_centers_long_t * 1e-6, pdf_long_t,␣
 ↪p0=p1)
dataset["pdf_longtime"] = pdf_long_t
dataset["x_pdf_longtime"] = bins_centers_long_t * 1e-6
```

```
<ipython-input-25-d08630fe76fc>:2: RuntimeWarning: overflow encountered in exp
  P_b = A * np.exp(-B * np.exp(-z / (ld)) - z / lb)
```

[27]:
```
A = 0.14e8
B = 400
ld = 70
lb = 500
p0 = [B, ld, lb]


plt.semilogy(bins_centers_long_t, pdf_long_t, label="$P_z(\Delta r_z)$")
plt.fill_between(
    bins_centers_long_t, pdf_long_t - err_long_t, pdf_long_t + err_long_t,␣
 ↪alpha=0.3
)

zz = np.linspace(-4, 4, 1000)
plt.plot(zz, Pdeltaz_long(zz * 1e-6, *popt), label="fit", color="k")
plt.xlim(-4, 4)
plt.legend()
plt.ylabel("$P(\Delta z)$")
plt.xlabel("$\Delta z$ [$\mathrm{\mu m}$]")
```

[27]:
```
Text(0.5, 0, '$\\Delta z$ [$\\mathrm{\\mu m}$]')
```



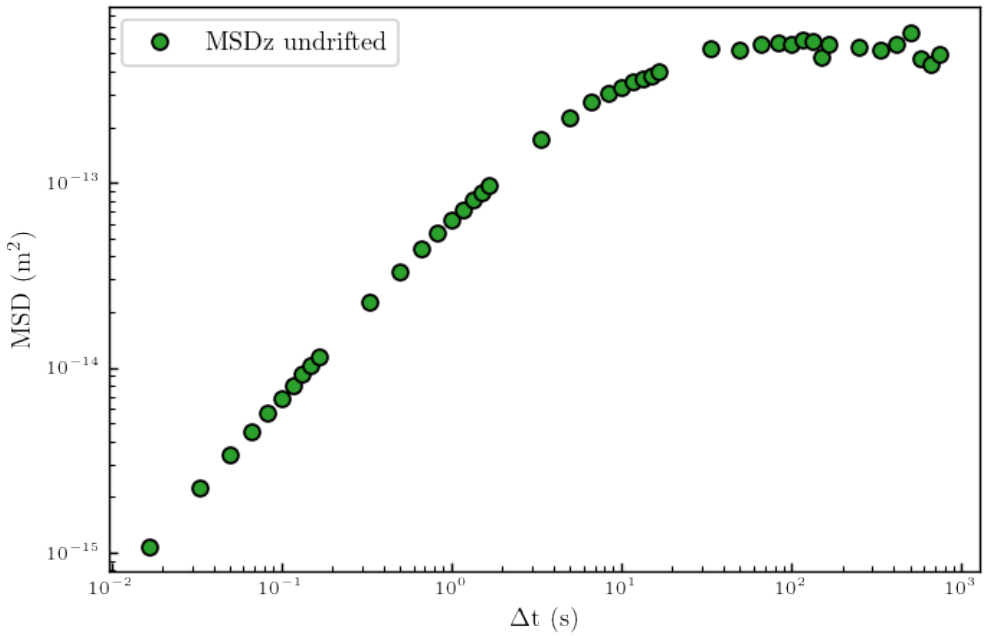
[28]:
```
print("We measure, B = {:.2f}, ld = {:.2f} nm, lb = {:.2f} nm".format(*popt))
B, ld, lb = popt
```

```
We measure, B = 20.71, ld = 71.84 nm, lb = 504.78 nm
```

3.6 Analyse of the MSD z plateau

[29]:
```
t = np.concatenate(
    (
        np.arange(1, 10, 1),
        np.arange(10, 100, 10),
        np.arange(100, 1000, 100),
        np.arange(1000, 10000, 1000),
        np.arange(10000, 50000, 5000),
    )
)

MSD_z_dedrift = MSD(z_dedrift * 1e-6, t)

plt.loglog(time[t], MSD_z_dedrift, "o", label="MSDz undrifted", color="tab:
 ↪green")
plt.legend()
plt.ylabel("MSD (m$^2$)")
plt.xlabel("$\Delta$t (s)")
```

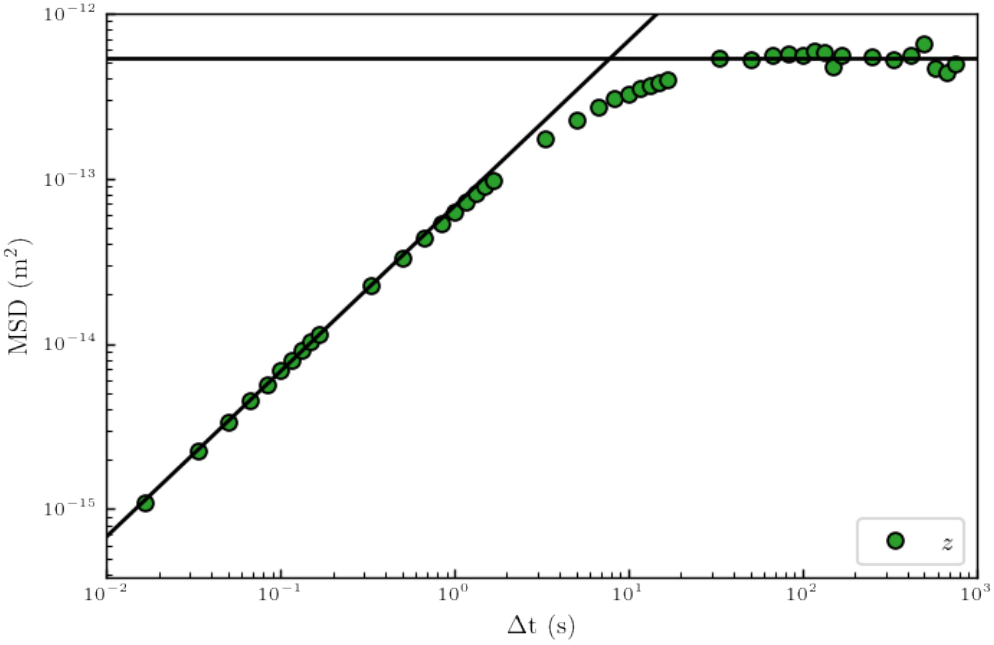
[29]:
```
Text(0.5, 0, '$\\Delta$t (s)')
```



[30]:
```
plateau = np.mean(MSD_z_dedrift[time[t] > 1e2])
```

[31]:
```
plt.loglog(time[t], MSD_z_dedrift, "o", label="$z$", color="tab:green")
plt.plot(tt, [plateau] * len(tt), "k")
plt.plot(tt, f(tt, *popt_3), color="k")
plt.xlim((1e-2, 1e3))
plt.ylim((None, 1e-12))
plt.legend()
plt.ylabel("MSD (m$^2$)")
plt.xlabel("$\Delta$t (s)")
```

[31]:
```
Text(0.5, 0, '$\\Delta$t (s)')
```



[32]:
```
np.mean(MSD_z_dedrift[time[t] > 1e2])
```

[32]:
```
5.348018604759325e-13
```

[33]:
```
# dataset["plateau_MSD"] = popt[0]
dataset["plateau_MSD"] = np.mean(MSD_z_dedrift[time[t] > 1e2])
print("Measured plateau : {:e}".format(popt[0]))
```

```
Measured plateau : 2.070536e+01
```

The MSD plateau is theoritically given by:

$$Plateau = \int_{-\infty}^{+\infty} \Delta z^2 P_{\Delta z, t \to +\infty}(\Delta z, B, l_d, l_b) d\Delta z \tag{11}$$

[34]:
```
x_Th_Plateau = bins_centers_long_t * 1e-6


def Theoritical_Plateau(B, ld, lb):
    x = dataset["x_pdf_longtime"]
    P = Pdeltaz_long(x, B, ld, lb) / trapz(Pdeltaz_long(x, B, ld, lb), x)

    res = trapz((x ** 2) * P, x)
    return res
```

[35]:
```
def minimize_plateau(x):
    B = x[0]
    ld = x[1]
    lb = x[2]
    return (
        np.log(Theoritical_Plateau(B, ld, lb)) - np.log(dataset["plateau_MSD"])
    ) ** 2 / np.log(Theoritical_Plateau(B, ld, lb)) ** 2
```

[36]:
```
res_plateau = minimize(minimize_plateau, x0=[B, ld, lb])
print("We measure, B = {:.2f}, ld = {:.2f} nm, lb = {:.2f} nm".
 ↪format(*res_plateau.x))
```

```
We measure, B = 20.71, ld = 71.84 nm, lb = 504.78 nm
```

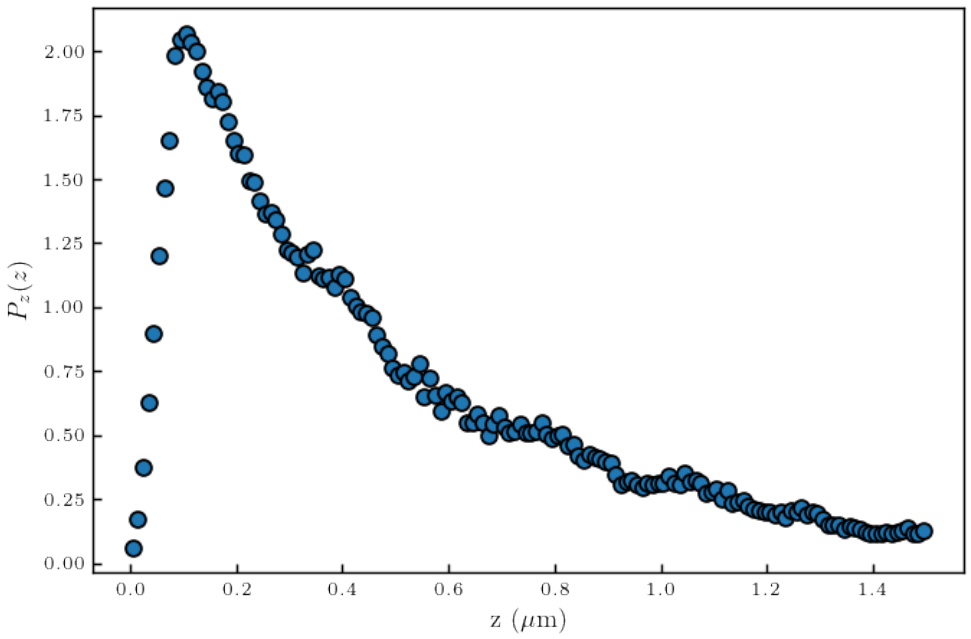
3.7 PDF of heights

[37]:
```
def logarithmic_hist(data, begin, stop, num=50, base=2):

    if begin == 0:
        beg = stop / num
        bins = np.logspace(
            np.log(beg) / np.log(base), np.log(stop) / np.log(base), num - 1,␣
 ↪base=base
        )
        widths = bins[1:] - bins[:-1]
        bins = np.cumsum(widths[::-1])
        bins = np.concatenate(([0], bins))
        widths = bins[1:] - bins[:-1]

    else:
        bins = np.logspace(
            np.log(begin) / np.log(base), np.log(stop) / np.log(base), num,␣
 ↪base=base
        )
        widths = bins[1:] - bins[:-1]

    hist, bins = np.histogram(data, bins=bins, density=True)
```

```
    # normalize by bin width
    bins_center = (bins[1:] + bins[:-1]) / 2

    return bins_center, widths, hist
```

[38]:
```
pdf_z, bins_center_pdf_z = pdf(z_dedrift[z_dedrift < 1.5], bins=150)
plt.plot(bins_center_pdf_z, pdf_z, "o")
plt.xlabel("z ($\mathrm{\mu m}$)")
plt.ylabel("$P_z(z)$")
```

[38]: Text(0, 0.5, '$P_z(z)$')



The idea now is to find where the substrate is, to do this we will use a first method which consist to adjust the PDF with an offset to make it fit with the measured mean Diffusion coefficient. With :

$$< D_i >= \int_{-\infty}^{\infty} dz D_i(z) P(z) \tag{12}$$

For z we are going to use the Padé approx :

$$D_z(z) \approx D_0 \left(\frac{6z^2 + 2rz}{6z^2 + 9rz + 2r^2} \right) \tag{13}$$

For x we are going to use the Faxen formula :

$$D_x(z) \approx D_0 \left[1 - \frac{9}{16} \left(\frac{r}{z} \right) + \frac{1}{8} \left(\frac{r}{z} \right)^3 - \frac{45}{236} \left(\frac{r}{z} \right)^4 - \frac{1}{16} \left(\frac{r}{z} \right)^5 \right] \tag{14}$$

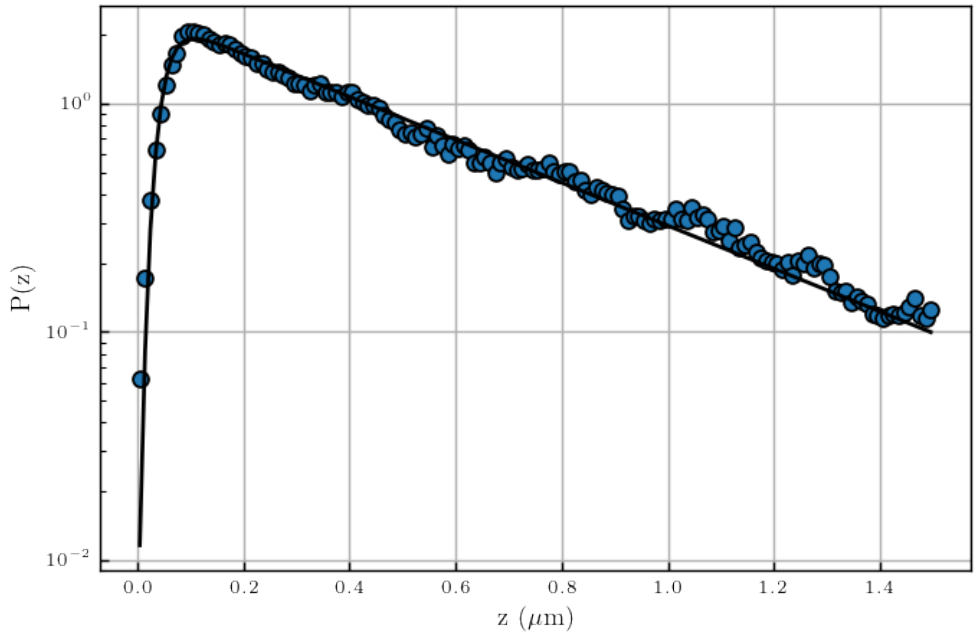
To do this we will fit the PDF with an offset, adjust it with the mean value of z. Let's first do it over z

[39]:
```
def P_b_off(z, z_off, B, ld, lb):
    z_off = z_off * 1e-6
    lb = lb * 1e-9
    ld = ld * 1e-9
    z = z - z_off
    P_b = np.exp(-B * np.exp(-z / (ld)) - z / lb)
    P_b[z < 0] = 0

    # Normalization of P_b

    A = trapz(P_b, z * 1e6)
    P_b = P_b / A

    return P_b
```

[40]:
```
# Normalization of the PDF

pdf_z = pdf_z / trapz(pdf_z, bins_center_pdf_z)


p2 = [0, B, ld, lb]

popt, pcov = curve_fit(P_b_off, bins_center_pdf_z * 1e-6, pdf_z, p0=p2)
```

```
<ipython-input-39-f5e0b340e678>:6: RuntimeWarning: overflow encountered in exp
  P_b = np.exp(-B * np.exp(-z / (ld)) - z / lb)
<ipython-input-39-f5e0b340e678>:12: RuntimeWarning: invalid value encountered in
true_divide
  P_b = P_b / A
```

[41]:
```
plt.semilogy(bins_center_pdf_z, pdf_z, "o")
plt.plot(bins_center_pdf_z, P_b_off(bins_center_pdf_z * 1e-6, *popt), "k")

plt.xlabel("z $\mathrm{(\mu m)}$")
plt.ylabel("P(z)")
plt.grid()
```



[42]:
```
mean_Dx = (popt_1[0] + popt_2[0]) / 2
mean_Dz = popt_3[0]
print(
    "We measure a mean diffusion coefficient of {:.3f}D0 for the perpendicular␣
 ↪motion and of {:.3f}D0 for the parallel motion".format(
        (popt_1[0] + popt_2[0]) / 2, popt_3[0]
    )
)


dataset["D_para"] = mean_Dx
dataset["D_perp"] = mean_Dz
```

```
We measure a mean diffusion coefficient of 0.522D0 for the perpendicular motion
and of 0.243D0 for the parallel motion
```

[43]:
```
Do = 4e-21 / (6 * np.pi * 0.001 * r)


def Dz_z(z):
    result = (6 * z * z + 2 * r * z) / (6 * z * z + 9 * r * z + 2 * r * r)
    return result


def Dx_z(z):
```

```
    result = (
        1
        - 9 / 16 * (r / (z + r))
        + 1 / 8 * (r / (z + r)) ** 3
        - 45 / 256 * (r / (z + r)) ** 4
        - 1 / 16 * (r / (z + r)) ** 5
    )
    return result
```

[44]:
```
def minimizer(z_off):
    Dx_pdf = trapz(
        Dx_z(bins_center_pdf_z * 1e-6)
        * P_b_off(bins_center_pdf_z * 1e-6, z_off, *popt[1:]),
        bins_center_pdf_z,
    )
    Dz_pdf = trapz(
        Dz_z(bins_center_pdf_z * 1e-6)
        * P_b_off(bins_center_pdf_z * 1e-6, z_off, *popt[1:]),
        bins_center_pdf_z,
    )

    return np.abs((1 - mean_Dx / Dx_pdf) + (1 - mean_Dx / Dx_pdf))


res = minimize(minimizer, 0, method="nelder-mead")
```

[45]:
```
offset = res
```

[46]:
```
offset = np.mean(res["final_simplex"][0])
print(
    "From the measurement of the mean diffusion coefficient, we measure an␣
 ↪offset of {:.3f} um".format(
        offset
    )
)
```

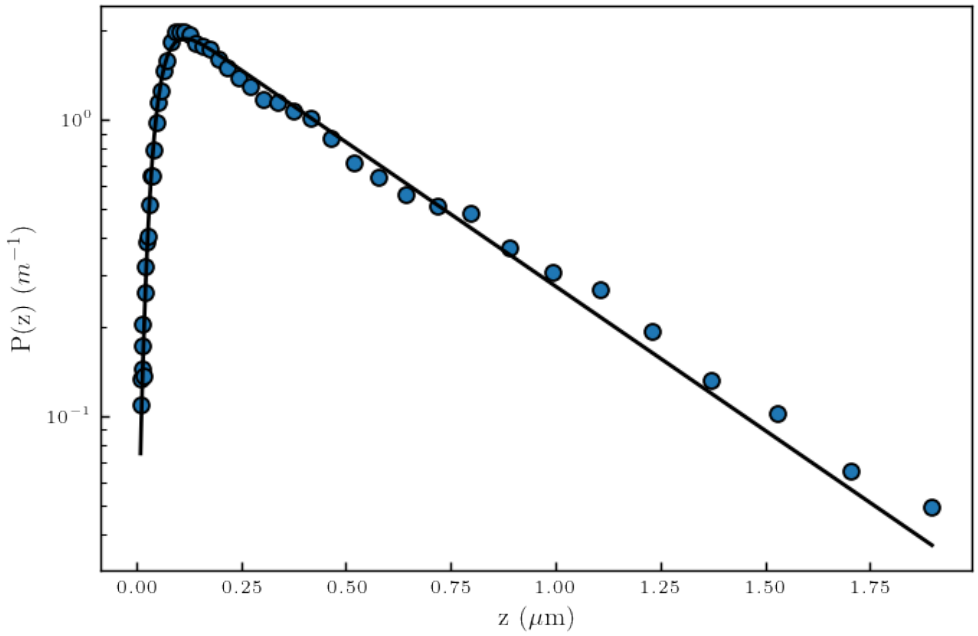
```
From the measurement of the mean diffusion coefficient, we measure an offset of
0.005 um
```

[47]:
```
def logarithmic_hist(data, begin, stop, num=50, base=2):
    """
    Function to make logarithmic histograms to have more points
    near the surface and where the particle spend the most of its time.
    """
    if begin == 0:
        beg = stop / num
        bins = np.logspace(
```

```
            np.log(beg) / np.log(base), np.log(stop) / np.log(base), num - 1,␣
 ↪base=base
        )
        widths = bins[1:] - bins[:-1]
        bins = np.cumsum(widths[::-1])
        bins = np.concatenate(([0], bins))
        widths = bins[1:] - bins[:-1]

    else:
        bins = np.logspace(
            np.log(begin) / np.log(base), np.log(stop) / np.log(base), num,␣
 ↪base=base
        )
        widths = bins[1:] - bins[:-1]

    hist, a = np.histogram(data, bins=bins, density=True)
    # normalize by bin width
    bins_center = (bins[1:] + bins[:-1]) / 2

    return bins_center, widths, hist


bins_center_pdf_z, widths, pdf_z = logarithmic_hist(z_dedrift, 0.01, 2, num=50,␣
 ↪base=12)

p2 = [0, B, ld, lb]
popt_pdf, pcov_pdf = curve_fit(P_b_off, bins_center_pdf_z * 1e-6, pdf_z, p0=p2)
dataset["pdf_z"] = pdf_z
dataset["x_pdf_z"] = bins_center_pdf_z * 1e-6

plt.semilogy(bins_center_pdf_z, pdf_z, "o")
plt.plot(bins_center_pdf_z, P_b_off(bins_center_pdf_z * 1e-6, *popt_pdf),␣
 ↪color="black")

plt.xlabel("z ($\mathrm{\mu m}$)")
plt.ylabel("P(z)  ($m^{-1}$)")
```

```
<ipython-input-39-f5e0b340e678>:6: RuntimeWarning: overflow encountered in exp
  P_b = np.exp(-B * np.exp(-z / (ld)) - z / lb)
<ipython-input-39-f5e0b340e678>:12: RuntimeWarning: invalid value encountered in
true_divide
  P_b = P_b / A
```

[47]: `Text(0, 0.5, 'P(z)  ($m^{-1}$)')`



[48]:
```
offset_pdf, B_pdf, ld_offset, lb_offset = popt_pdf
```

We write the diffusion function.

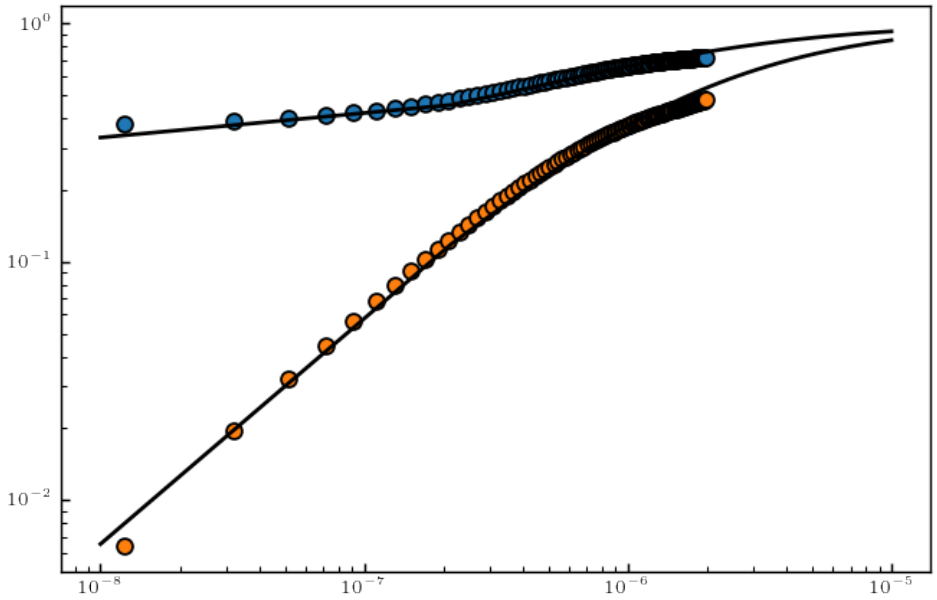
[49]:
```
def Dz_z(z, off):
    off = off * 1e-6
    z = z - off
    result = (6 * z * z + 2 * r * z) / (6 * z * z + 9 * r * z + 2 * r * r)
    return result


def Dx_z_off(z, offset):
    offset = offset * 1e-6
    z = z + offset
    result = (
        1
        - 9 / 16 * (r / (z + r))
        + 1 / 8 * (r / (z + r)) ** 3
        - 45 / 256 * (r / (z + r)) ** 4
        - 1 / 16 * (r / (z + r)) ** 5
    )
    return result
```

3.8 Measuring the diffusion coefficient using the Frishman and Ronceray's method

[50]:
```
from scipy.io import loadmat

dataset["D_0"] = 4e-21 / (6 * np.pi * 0.001 * dataset["r"])
D = loadmat("diffusionAnalysis.mat")["diffusion"]
dataset["z_D"] = D[:, 0]
dataset["z_D_x"] = (D[:, 1] + D[:, 2]) / 2
dataset["z_D_z"] = D[:, 3]
```

[51]:
```
plt.plot(dataset["z_D"], dataset["z_D_x"] / dataset["D_0"], "o")
plt.loglog(dataset["z_D"], dataset["z_D_z"] / dataset["D_0"], "o")

zz = np.linspace(1e-8, 1e-5)
plt.plot(zz, Dz_z(zz, 0), "k")

plt.plot(zz, Dx_z_off(zz, 0), "k")
```

[51]:
```
[<matplotlib.lines.Line2D at 0x27a85cc7850>]
```



[52]:
```
def c_P_D(B, ld, lb, offset=None):
    if offset == None:
        offset = 0

    z = np.linspace(1e-9, 15e-6, 1000)
```

```
    P_D = Dz_z(z, offset) * Do * P_b_off(z, offset, B, ld, lb)

    return Dz_z(z, offset) * Do, P_D / np.trapz(P_D, z)


def _P_Dz_short_time(Dz, Dt, B, ld, lb, offset=None):
    if offset == None:
        offset = 0

    D_z, P_D = c_P_D(B, ld, lb, offset)

    P = np.trapz(
        P_D / np.sqrt(4 * np.pi * D_z * Dt) * np.exp(-(Dz ** 2) / (4 * D_z *␣
 ↪Dt)), D_z
    )

    return P


def P_Dz_short_time(Dz, Dt, B, ld, lb, offset=None):
    if offset == None:
        offset = 0

    P = [_P_Dz_short_time(i, Dt, B, ld, lb, offset=offset) for i in Dz]
    P = np.array(P)
    P = P / np.trapz(P, Dz)

    return P
```

4 Fit everything in the same time !

Finaly we can fit everything in the same time to recap we have :

- MSD x and MSD y => $< D >$
- MSD z => $< D >$
- mean $< D >$ with the pdf
- Long time pdf Δz => l_d, l_b, B
- Pdf z => $offset, l_d, l_b, B$
- D parallel, perp => offset

The minimizer χ^2 we are going to optimize can be written as :

$$\chi^2 = \sum_{n=1}^{N} \chi_n^2 \tag{15}$$

$$\chi_n^2 = \sum^{A}(n)_i = 1\frac{1}{\sigma_{ni}}(y_{ni} - y_n(x_{ni}, \boldsymbol{a}))^2 \tag{16}$$

with σ_{ni} the uncertainty (can be set to 1), A the number of point in the dataset for each function, y_n, nth equation, $\boldsymbol{a}$ the fit parameters

We have nonlinear functions so we can use the Marquardt to optimize or Nelder-Mead methods to optimize the minimizer.

[53]:
```
def minimizer_diffusion_coeff(mean_D_para, mean_D_perp, z_off, B, ld, lb):
    # minimization of the mean diffusion coefficient measurement with the PDF␣
 ↪and MSD
    a = trapz(
        Dx_z_off(bins_center_pdf_z * 1e-6, z_off)
        * P_b_off(bins_center_pdf_z * 1e-6, z_off, B, ld, lb),
        bins_center_pdf_z,
    )
    b = trapz(
        Dz_z(bins_center_pdf_z * 1e-6, z_off)
        * P_b_off(bins_center_pdf_z * 1e-6, z_off, B, ld, lb),
        bins_center_pdf_z,
    )
    at = mean_Dx
    bt = mean_Dz
    return (a - at) ** 2 / at ** 2 + (b - bt) ** 2 / bt ** 2


dataset["z"] = z_dedrift
dataset["x"] = x
dataset["y"] = y


def minimizer_Dz_small_t(B, ld, lb):
    xi = 0

    for n, i in enumerate([1, 2, 3]):
        Dezs = (dataset["z"][0:-i] - dataset["z"][i:]) * 1e-6
        Dezs = Dezs  # - np.mean(Dezs)

        hist, bins_center = pdf(Dezs[~np.isnan(Dezs)], bins=30)
        hist = hist / np.trapz(hist, bins_center)

        Dz_th = bins_center
        PPP = P_Dz_short_time(Dz_th, time[i], B, ld, lb)

        # xi = xi + np.nanmean((((np.abs(hist) - (PPP) ) ) ** 2) / ((np.
 ↪abs(hist)**2)))
        xi = xi + np.nanmean(
            ((hist[hist > 0] - PPP[hist > 0]) ** 2) / hist[hist > 0] ** 2
        )
    return xi
```

[54]:
```
dataset["D_para"] = mean_Dx
dataset["D_perp"] = mean_Dz


def minimizer(x, *args):
    data = dataset
    ld = x[0]
    lb = x[1]
    B = x[2]
    offset_dif = x[3]

    chi_mean_D_pdf = minimizer_diffusion_coeff(
        dataset["D_para"], dataset["D_perp"], 0, B, ld, lb
    )
    chi_MSD_plateau = minimize_plateau([B, ld, lb])

    E_longtime_pdf = (Pdeltaz_long(data["x_pdf_longtime"], B, ld, lb)) - (
        data["pdf_longtime"]
    )
    chi_longtime_pdf = np.mean(
        (E_longtime_pdf[E_longtime_pdf > -np.inf] ** 2)
        / (((Pdeltaz_long(data["x_pdf_longtime"], B, ld, lb))) ** 2)
    )

    E_chi_pdf_z = P_b_off(data["x_pdf_z"], 0, B, ld, lb) - data["pdf_z"]
    chi_pdf_z = np.nanmean(
        (E_chi_pdf_z[E_chi_pdf_z > -np.inf] ** 2)
        / ((P_b_off(data["x_pdf_z"], 0, B, ld, lb)) ** 2)
    )

    E_D_z = (Dz_z(data["z_D"], offset_dif)) - (data["z_D_z"] / Do)
    chi_D_z = np.mean(
        (E_D_z[E_D_z > -np.inf] ** 2) / ((Dz_z(data["z_D"], offset_dif)) ** 2)
    )

    E_D_x = (Dx_z_off(data["z_D"], offset_dif)) - (data["z_D_x"] / Do)
    chi_D_x = np.mean(
        (E_D_x[E_D_x > -np.inf] ** 2) / ((Dx_z_off(data["z_D"], offset_dif)) **␣
↪2)
    )

    chi_Dz_small_t = minimizer_Dz_small_t(B, ld, lb)

    summ = (
        chi_mean_D_pdf
        + chi_MSD_plateau
        + chi_longtime_pdf
```

```
        + chi_pdf_z
        + chi_D_z
        + chi_D_x
        + chi_Dz_small_t
    )

    return summ
```

[55]:
```
B = 5
ld = ld_offset
x0 = [ld, 550, B, 0, offset_pdf]
```

[56]:
```
from scipy.optimize import leastsq

options = {
    "maxc1or": 30,
    "ftol": 2.2e-10,
    "gtol": 1e-5,
    "eps": 1e-08,
    "maxfun": 15000,
    "maxiter": 15000,
    "maxls": 20,
    "finite_diff_rel_step": None,
}

res = minimize(
    minimizer,
    x0,
    method="BFGS",
    tol=1e-1,
)
```

[57]:
```
res.x
results = {
    "ld": res.x[0],
    "lb": res.x[1],
    "B": res.x[2],
    "offset_diffusion": res.x[3],
}

results
```

[57]:
```
{'ld': 25.53322987706852,
 'lb': 549.9956782843908,
 'B': 4.856896668800334,
 'offset_diffusion': 0.0019089236000464675}
```

This final result has been used to plot theories along the manuscript.