Investigation of the Dynamics of Rod-Like Colloids during Sedimentation.

by

© Haruki Hirasawa

A thesis submitted to the

Department of Physics and Physical Oceanography

in partial fulfillment of the

requirements for the degree of

Bachelor of Science (Honours)

Memorial University of Newfoundland

April 2016

St John's Newfoundland & Labrador

Abstract

We study the dynamics of rod-like colloidal particles during sedimentation using experiments and simulations. The experiments are performed using a laser scanning confocal microscope and the simulations using Dissipative Particle Dynamics (DPD) and rigid body dynamics. We observe clustering of the particles in both experiment and simulation, an effect that is reminiscent of a clustering instability predicted by Crowley in 1971 [1], and observe that the colloids seem to cluster more when held in a vertical orientation by an electric field. Additionally, using DPD we simulate a number of simple systems to test the behavior of hydrodynamic effects such as the Crowley instability on colloidal particles at different Péclet numbers. We observe that the random motion becomes significant enough to disrupt hydrodynamics when Pe $<$100, which fits with the definition of the non-Brownian domain (Pe $>$100).

Acknowledgments

The completion of this thesis likely would not have been possible without the support and assistance of many good people. Firstly, I would like to take this chance to formally thank my family for guiding, supporting, and, when necessary, pushing me through the nearly 22 years of my existence, as well as providing support and advice during the writing of this thesis.

I would also like to thank my fellow denizens of the honors study room: Riley Brooks, Anna O'Grady, Cole Walsh, Tyler Downey, and many others, without whom I may well have been crushed by stress. I'd also like to acknowledge the contributions of the community of Stack Overflow, without whom this document would be a lot uglier and I may still be trying to stomp out bugs in my code, and the editors of Wikipedia, for creating the most important tool in the arsenal of any student.

Most of all, I extremely grateful to Dr. Anand Yethiraj and Dr. Ivan Saika-Voivod for their invaluable guidance over the last year, for giving me this opportunity to get involved in interesting and exciting research, and for teaching me so much about the field of soft matter physics and research in general. Working with you has been a pleasure and an honor.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The term *colloid* is, for a scientific term, rather ambiguous in definition. While the term *colloidal suspension* can be used without ambiguity to refer to a system in which insoluble microscopic (10^{-9} to 10^{-6} m in size) particles of one substance are dispersed throughout another substance, the term *colloid* can be used to refer to either the system in its entirety or the dispersed substance alone. For reasons of convenience and habit, we fall into the latter camp and will use *colloid* to refer to the dispersed particles, against the recommendations of some chemists [2].

The physics of colloids have been the subject of study for well over a century since they were first described by Thomas Graham in the 1860s [3]. Recent research on colloidal systems has been directed at problems such as improving our understanding of the physics of biological systems and creating novel "soft" materials. One property of colloids that has attracted interest from researchers is their ability to "self-assemble" into complex structures, such as crystals, chains, and clusters, through both Brownian

motion and as a result of being driven by outside forces [4, 5]. In the past, research on colloidal self-assembly has largely been focused on near-equilibrium systems [4] and spherical colloids for reasons of practicality. However, research has increasingly shifted towards the study of more complex non-equilibrium colloidal systems, such as those under electromagnetic fields [5].

One important set of non-equilibrium systems are those in which colloidal particles are driven by a gravitational field (also known as sedimentation). Historically, sedimentation has been an important tool used to study colloidal suspensions and continues to be a useful near-equilibrium system for studies of colloid physics and self-assembly [6]. In addition, with improvements to techniques for synthesizing non-spherical particles, studies have begun to study the self-assembly of such particles during sedimentation [7, 8] which have demonstrated interesting phase behavior in concentrated suspensions of rod-like colloids. However, successfully creating colloidal structures using sedimentation can be difficult, as doing so can be very time consuming and delicate and often requires a high degree of control over the colloidal suspension. This means that inducing controlled self-assembly in non-spherical colloids is an even more delicate process, since the additional rotational degrees of freedom further complicate the dynamics of the sedimenting colloids, increasing the care needed to create colloidal crystals and other structures. We seek to help address this problem

by using experiments and simulations to study how the decrease of particle sizes to the colloidal domain affect various hydrodynamic effects observed in the dynamics of sedimenting particles in the hydrodynamic domain. In doing so we hope to get a better understanding of how anisotropy changes the dynamics of colloids during sedimentation.

Chapter 2

Background and Theory

2.1 Colloidal Hydrodynamics

Due to the complexity of hydrodynamic interactions between particles, understanding the dynamics of sedimenting colloids is a difficult task. The fact that hydrodynamic interactions tend to be long-ranged in the case of Stokesian drag ($\propto 1/r$) [6, p. 310] means that the complexity of the hydrodynamics of a system increases rapidly as the number of particles is increased. This is especially true when dealing with systems like ours, where we are dealing with hundreds of particles. Indeed, when dealing with groups of sedimenting particles, the flow of the fluids results in the chaotic motion of particles even when thermal Brownian motion is negligible [6, p. 298]. As a result, analytic hydrodynamic predictions can only be obtained for systems with few particles or many particles distributed with a high degree of regularity. Among the hydrodynamic effects we are interested in studying is an instability observed

in sedimenting particles that results in the clustering of the particles, known as the Crowley instability [1,9,10], and the periodic rotational motion of pairs of anisotropic particles during sedimentation [11]. We seek to investigate the role of the introduction of the rotational degree of freedom and Brownian motion on these effects.

When studying the dynamics of particles in the colloidal domain, it is necessary to take into account the influence of Brownian motion. Brownian motion results in thermal diffusion which become increasingly important as the particles decrease in size and the suspension increases in temperature. This additional contribution to the dynamics of the particles further complicates analytic solutions to the motion of the particles, making simulations a much more viable strategy for studying our systems theoretically (see Chapter 4).

2.1.1 Péclet Number

When discussing the movement of colloidal particles in suspension, one useful quantity used to describe such a system is the Péclet number, Pe. The Péclet number is defined as the ratio between the advective and mass diffusive transport rates (or equivalently the product of the Reynolds and the Schmidt numbers) [12] and in the case of colloidal suspensions serves as a measure of the relative strengths of hydrodynamic and thermal Brownian forces. For a spherical particle of radius a moving at speed u through an

unbounded fluid the Péclet number is

$$\mathrm{Pe} = \frac{ua}{D} \tag{2.1}$$

where D is the thermal diffusion constant of the particle [6].

When $\mathrm{Pe} \gg 1$ the dynamics of the system are dominated by hydrodynamic interactions (macroscopic systems) while when $\mathrm{Pe} \ll 1$ the dynamics of the system are dominated by Brownian motion (microscopic systems). However, in the case of mesoscale colloidal systems such as those we examine ($0.01 < \mathrm{Pe} < 100$), it is necessary to take into account the effects of both interactions. We use the definition of the non-Brownian domain as any system in which Pe >100 [6].

If we consider the case of a spherical particle under Stokes drag, which diffuses according to the Stokes-Einstein equation, the terminal speed of the particle is $u_{term} = \frac{F}{\Gamma}$ and the thermal diffusion constant is $D = \frac{k_B T}{\Gamma}$ where Γ is the drag coefficient of the particle, T is the temperature, k_B is the Boltzmann constant, and F is the force driving the colloid. These relations give the following equation for the Péclet number:

$$\mathrm{Pe} = \frac{Fa}{k_B T} \tag{2.2}$$

meaning that the Péclet number does not depend explicitly on the form of the drag coefficient Γ for the system. This unusual result means that, other than temperature,

Pe does not depend on the characteristics of the solvent; a property that we will utilize to modify the Péclet numbers in our simulations of sedimenting colloidal particles.

In order to calculate the Péclet numbers of rod-like colloids, the equations used are the same as in the case of spherical particles. However, the radius a becomes the effective hydrodynamic radius R_H, which must be modified to reflect the non-spherical shape of the particles. In the case of a cylindrical particle of length L and diameter d, S. Hansen [13] the calculated hydrodynamic radius R_H using Monte Carlo simulations and found that

$$R_H = R_S(1.0304 + 0.0193x + 0.06229x^2 + 0.00476x^3 + 0.00166x^4 + 2.66 \times 10^{-6}x^7) \tag{2.3}$$

where $x = \ln(L/d)$, $R_S = (\frac{3}{4\pi}V)^{1/3}$, and V is the volume of the particle. Using this result we can estimate the drag coefficient $\Gamma = 6\pi\mu R_H$, where μ is the dynamic (shear) viscosity, and thereafter the Péclet number for a rod-like colloid.

2.1.2 Crowley Instability

One interesting observation made in studies of the hydrodynamics of macroscopic particles during sedimentation is the fact that particles tend to cluster together when sedimenting at low Reynolds numbers. This effect, known as the Crowley insta-

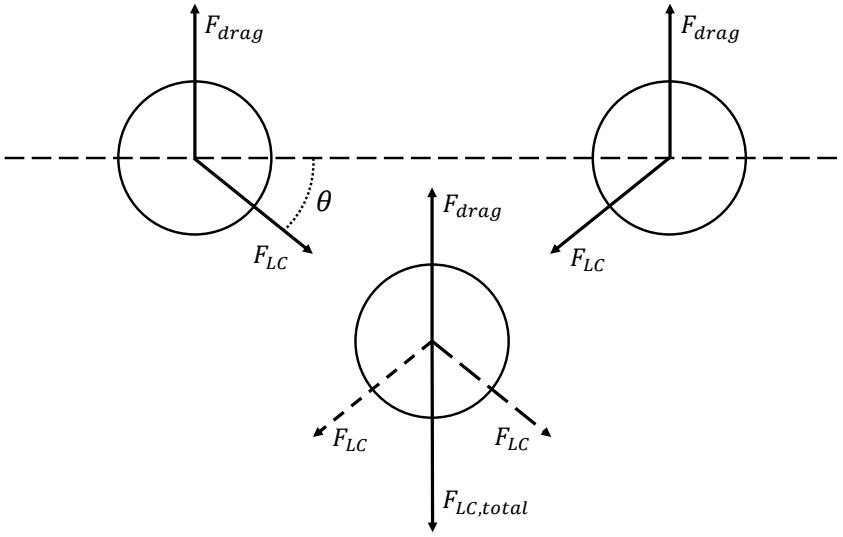

Figure 2.1: Illustration of the line of center and drag forces on a group of three sedimenting spherical particles.

bility, was first theorized and experimentally demonstrated by Joseph Crowley in 1971 [1] and has been studied using hydrodynamic theory and experimentation in the case of the sedimentation of 2D lattices of spherical particles [10] and 1D lattices of anisotropic particles [9].

The Crowley instability is the result of two hydrodynamic effects. The first is that if a sphere is placed near another sphere, the Stokes drag force on both particles is effectively reduced to approximately

$$F_D = 6\pi\mu a u \left(1 - \frac{3}{4}\frac{a}{d}\right) \tag{2.4}$$

for $a/d \ll 1$, where μ is the dynamic viscosity and d is the separation between the spheres. The second is that when two spheres sediment near each other, there is a force F_{LC} between them in the direction of the line joining them. If θ is defined as in figure 2.1, then the magnitude of this line of centers force is

$$F_{LC} = 6\pi\mu a u \left(\frac{3}{4}\frac{a}{d}\right) \sin(\theta). \tag{2.5}$$

If there are only two spheres, there is no relative motion between them as the line of centers force and the Stokes drag reduction apply to both spheres equally. However, if there are three or more spheres, there will be relative motion between the particles due to the line of center force. As can be seen in figure 2.1, in the case where one sphere is ahead of two others, the line of center forces bring the trailing particles together. When many particles sediment together, these forces cause instabilities in the particles and ultimately the formation of clusters of particles.

2.1.3 Dynamics of Anisotropic Particles

When dealing with non-spherical particles, the introduction of rotational degrees of freedom further increases the complexity of the hydrodynamics of sedimentation. For example, in [11] it was shown using experiments that when two anisotropic particles sediment next to one another at low Reynolds numbers and high Péclet numbers,

the particles rotate periodically in phase as illustrated in figure 2.2. When a third particle is added to such a system such that the particles are slightly perturbed from a symmetric placement, these periodic rotations disappear. This type of interaction demonstrates that the dynamics of systems of many anisotropic particles are considerably more complex than those of spherical particles.

The question of how the anisotropy of the particles affects the Crowley instability of particles was addressed in [9] wherein the Crowley instability was observed in experiments with 1D arrays of disks when these disks begin perpendicular to the direction of motion. Interestingly, this work found that when the disks began in a vertical position the disks tended to spread outwards, demonstrating that the effects of rotational degrees of freedom can oppose the effects of the Crowley instability.

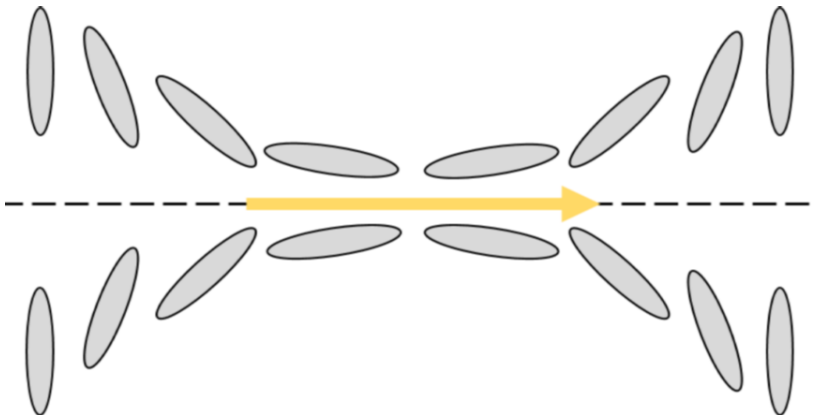

Figure 2.2: Diagram of the expected periodic rotational motion of a rod particle based on the findings of [11]. The yellow arrow indicates the direction of motion/sedimentation

Chapter 3

Experimental Setup

3.1 Experimental Setup

3.1.1 Colloidal Suspension

For our experiments, we used rod-like silica colloids synthesized by Thijs Besseling of the University of Utrecht. The colloids consist of a non-fluorescent core, followed by a 58 nm fluorescent shell, and then a 137 nm non-fluorescent shell. In total, the colloids are about 3.38 ± 0.2 μm in length and 0.631 ± 0.023 μm in diameter. Methods used for synthesis of the rod colloids and the fluorescent labeling are detailed in [14] and [15] respectively.

In order to get good quality confocal images of the colloids, it is necessary to suspend them in a solvent with a matching refractive index ($n = 1.45$). Failing to do so would result in the scattering of light in the sample, which would degrade the image quality. For our work we are interested in the dynamics of the colloids, so to be

Colloid Properties	Length	$3.38 \pm 0.2\ \mu m$
	Width	631 ± 23 nm
	Aspect Ratio	5.36
	Refractive Index	1.45
Solvent Properties	Glycerol Content	85% wt
	Density	1.22 g/cm^3
	Viscosity	89 mPa/s

Table 3.1: Information on the properties of the colloids and solvents at $25°C$ [16].

able to successfully track the rotational dynamics of the colloids, we would like to be able to create very high-speed movies of the colloids. However, due to the limitations of the imaging systems, this is not always possible. An alternative solution to this issue is to slow down the motion of the colloids by choosing a viscous solvent. We decided to take this route and opted to use a mixture of 85% by weight glycerol and water which has a refractive index of 1.451 and high viscosity of $\mu = 109$ mPa $\cdot$ s [16] at 20.0° C . Some information on the solvent and the colloids is listed in Table 3.1.

Because we are interested in the dynamics of the rod-like colloids in a dilute system, we dilute the original suspension provided by Thijs Besseling to about $1/100^{th}$ of the initial colloid concentration of 1 colloid per $\sim 40\ \mu$m^3, resulting in a suspension with ~ 1 colloid per 4000 μm^3. This was done by mixing 5 μL of the original colloidal mix into 245 μL of the glycerol-water mixture and sonicating the suspension.

3.1.2 Experimental Sample

For experiments without electric fields, the colloidal suspension was placed in 0.1mm thick glass capillary tubes which were glued to a standard glass microscope slide and sealed using UV curing glue. For experiments in which electric fields were used, the prepared colloidal suspensions were placed into a sample cell constructed using indium tin oxide (ITO) coated glass miccroscope covers as shown in 3.1.

To construct the electric field sample slides we first use a UV curing glue to glue an ITO coated cover slides onto a larger glass microscope slide with the ITO facing up. Then two strips of 0.1mm thick plastic spacer are placed across the cover slide about 1cm apart and a second ITO slide is glued on top, with the ITO coating facing downward. This second slide is offset from the first such that there is a section of ITO exposed on both slides. Once the glue is cured, colloidal silver paste is used to make contact with the ITO coatings on the two cover slides which are then connected to wires. Finally, the gap left between the spacers is filled with the colloidal suspension using capillary action and the sample is sealed by putting glue on the open ends and curing it under UV light while keeping the suspension itself covered to prevent the colloids from bleaching. An example of a completed sample slide is shown in figure 3.2.

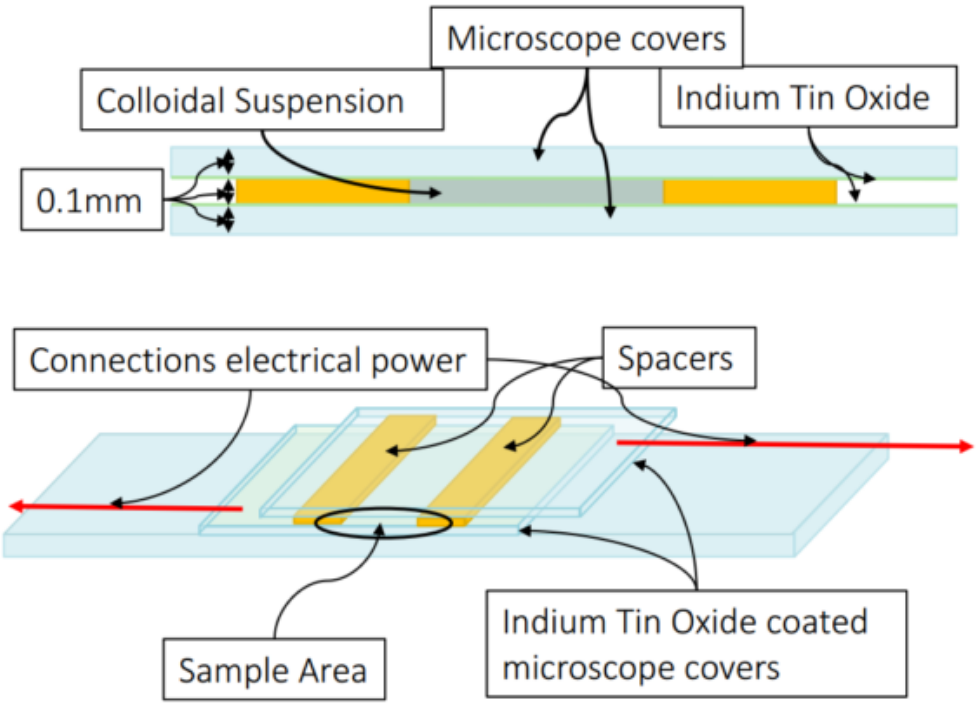

Figure 3.1: Top: Illustration of a side-on cross section of the experimental setup. Bottom: Illustration of the whole sample cell mounted on a microscope slide.

3.2 Confocal Image Analysis

In order to analyze the 3D confocal images of our rod-like colloids, it is necessary to be able to identify the colloids and determine their positions and orientations. There are a number of ways to approach this problem: a simple method would be to take an intensity threshold for the image and use the central moments of contiguous groups of

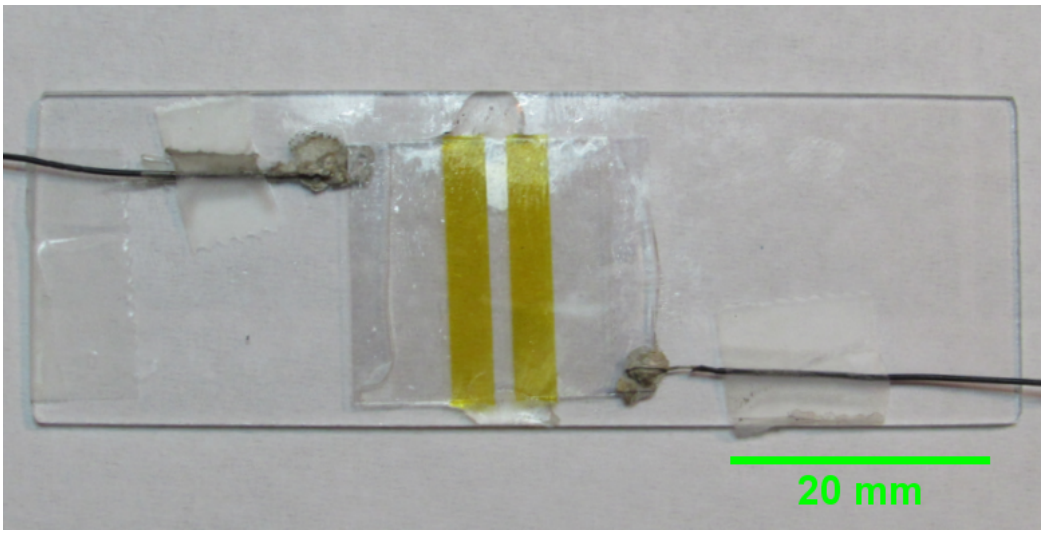

Figure 3.2: Image of a completed experimental sample. The sample suspension is placed between the yellow spacers.

pixels to determine their centroids and the direction of their primary axis. Another method developed by Besseling and Hermes et. al. [17] finds local maxima in the image and links them to find the backbones of the rod-like colloids. In our case, we have opted to use a method in which an intensity threshold is applied and the minimum volume enclosing ellipsoid is found for each contiguous group of pixels above the threshold. This was done by writing a Python implementation of the Minimum Volume Enclosing Ellipsoid MATLAB code developed by Nima Mostagh [18].

All image processing performed in this section was done using Python 2.7 along with the following Python packages:

- Numpy (http://www.numpy.org/) > 1.9.2

- Scipy (http://scipy.org/) > 0.16.0
- scikit-image (http://scikit-image.org/) > 0.11.3
- tifffile (https://pypi.python.org/pypi/tifffile) > 0.7.0
- pandas (http://pandas.pydata.org/) > 0.17.0
- matplotlib for plotting (http://matplotlib.org/) > 1.5.1

3.2.1 Description of Algorithm

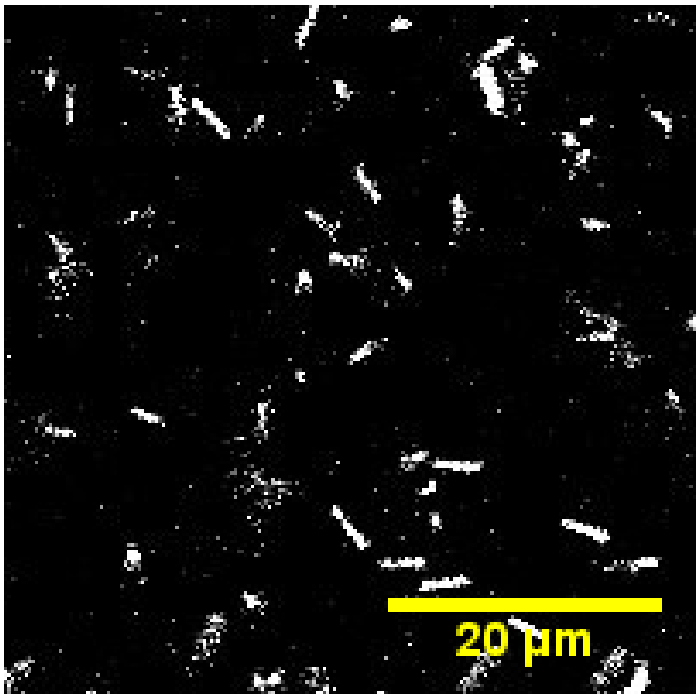

Figure 3.3: xy slice of a sample raw image.

Preprocessing

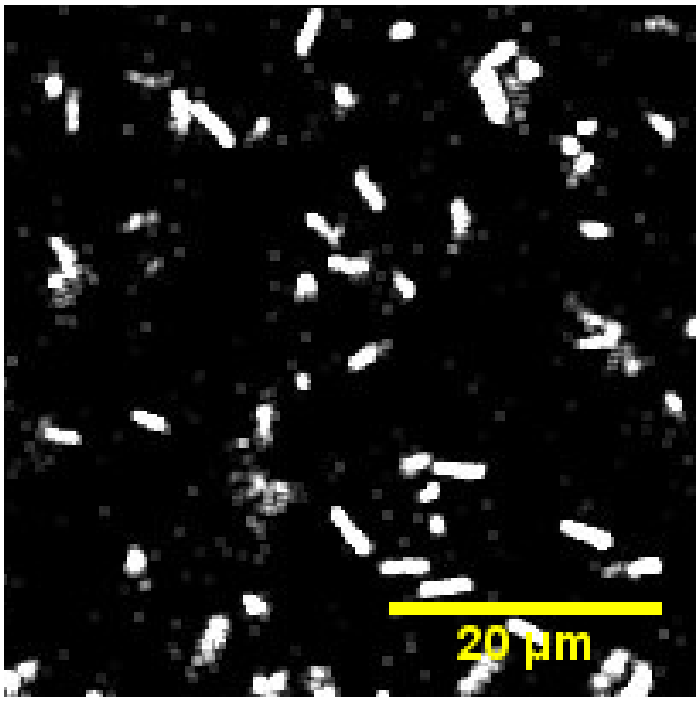

Figure 3.4: Sample image after the Gaussian filter is applied.

The images we get from the confocal microscope are 8bit grayscale Tagged Image File Format (.tif) stacks where each image in the stack is the xy plane at a different z coordinate. The package used to read the image stacks, tifffile, assumes that the colloids have a high intensity (white) on a low intensity (black) background. To start processing the image stacks, they are read using tifffile and converted into a Numpy array.

Before starting the analysis of the confocal images, it is necessary to process the

image in order to make them more suitable for analysis. The first step in the image analysis process is to smooth the image by convolving with a 3D Gaussian function with a standard deviation of 1.0px. This is done in order to smooth out the effects of white noise, making the background more uniform and ensuring that smaller or dimmer rods don't get split apart when the intensity threshold is taken. Afterwards, if the distance between z-slices is different from the size of the pixels in the xy-slices, the image is stretched and interpolated to make the pixels cubes.

Image Threshold

To determine what pixels in the image belong to the colloids, an intensity threshold is found and applied to the image. There are numerous methods for finding a suitable threshold for the images. If the rods are relatively dispersed and there is a significant difference in intensity between the rods and the background then one might opt to use Otsu's method [19], which is best suited for finding thresholds for images with bimodal intensity distributions. This is the method we use most often on our images, as our experiments use relatively dilute suspensions of colloids. Figure 3.5 is an example of an image where the threshold was found using the scikit-image Otsu's method function.

If there is a variation in the background intensity that is difficult to remove, as

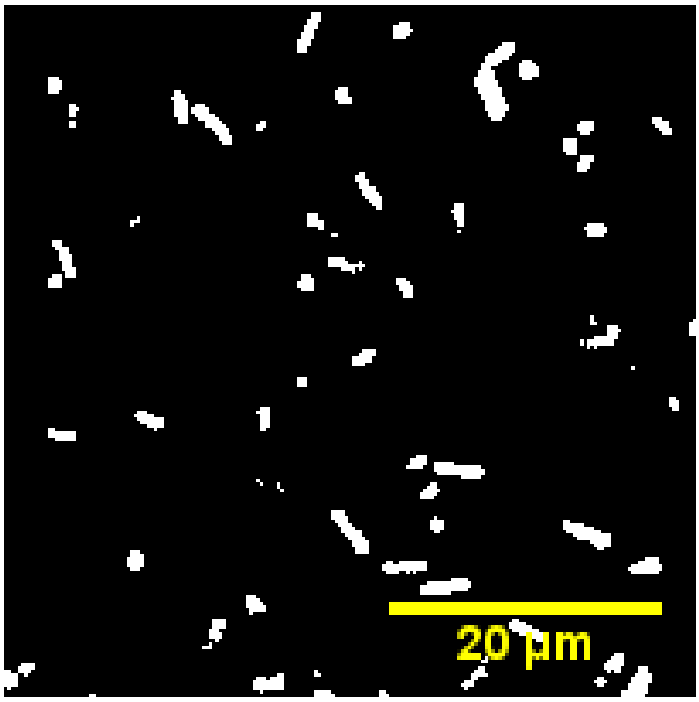

Figure 3.5: Sample image after an intensity threshold found using Otsu's method [19] is applied.

can be the case when there is a variation in the colloid concentration in the image, one can use local threshold methods which calculate the threshold for subsections of the image. Another method that can be used if the image has a high concentration of colloids and the colloids fill most of the image, is to find a threshold intensity by identifying the threshold value that maximizes the number of contiguous regions of pixels above the threshold that are larger than a given minimum volume. Finally, if the approximate dimensions of the colloids are known, the threshold can be refined by iteratively taking the threshold, identifying contiguous regions of pixels in the image

that are above the threshold and are too large to be a colloid, isolating such regions, and computing new thresholds in order to break these regions apart. These methods are most useful when finding the threshold of images where the concentration of the colloids is high, reducing the amount of background and making Otsu's method less effective.

Ellipsoid Fitting

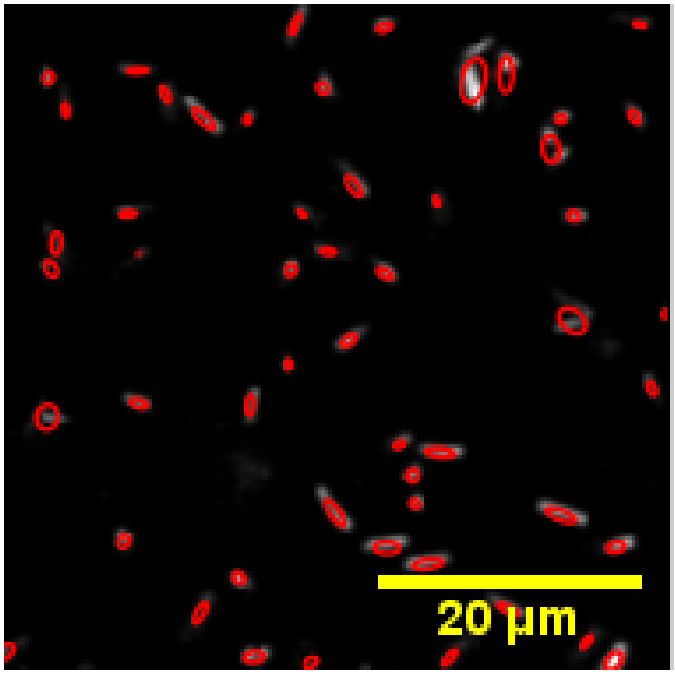

Figure 3.6: The intersection of the minimum volume enclosing ellipsoids with the plane of the image. The intersections of the ellipsoids and the plane of the image were calculated using an extension of the method described in [20] and the code used to do so is in Appendix A.2

Once the thresholding of the image is completed, groups of contiguous pixels with intensity above the threshold are identified and their sizes are found. For each group larger than a user defined minimum size (10 to 20 pixels in our images), we find the minimum volume ellipsoid that encloses the points in the group (the MVEE). The ellipsoids are calculated using an iterative algorithm developed by Nima Moshtagh [18] for MATLAB which was converted to Python using Numpy (see Appendix A.1). MVEE is an iterative numerical method, so there is a user defined tolerance value which is used to stop the algorithm. For our images we use a tolerance value of 0.01. Finding the MVEE for a group of points works well when the points are in a shape that is near ellipsoidal, like our rods, but for other cases, such as if two or more rods are stuck together, some points may end up outside the ellipsoid.

Once the ellipsoids are found, they are analyzed to get information like their lengths, orientations, and positions, which is then converted to a Pandas DataFrame and saved to a .csv file. Once a time series of 3D confocal images with adequate time resolution has been processed, it is possible to track the trajectories of the colloids using the centroids of the fitted ellipsoids and the Python particle tracking package trackpy [21], which is based on an set of IDL routines developed by Crocker and Grier [22].

3.3 Experiments

3.3.1 Experimental Procedure

In order to investigate the dynamics of the rod-like colloids, we obtained and analyzed 3D time series confocal images. During the course of the experimentation phase of our research, we performed three main types of experiments:

- Colloids begin lying horizontal and can freely rotate during sedimentation
- Colloids are initially held vertical by an electric field, which is turned off shortly after the colloids began to sediment, allowing the colloids to freely rotate.
- Colloids are held vertical by an electric field while the colloids sediment, preventing the colloids from rotating.

In particular, by comparing the results from the experiments in which the colloids were held vertical by the electric field and those in which they were not, it is possible to get an idea of how the introduction of a rotational degree of freedom might affect the dynamics of the colloids during sedimentation.

These experiments tended to follow the same basic procedure. First, the experimental sample is placed upside down to allow the colloids to sediment to the top wall of the sample. This typically takes 6 to 8 hours due to the high viscosity of the solvent. If the experiment being performed requires that the colloids start in a

vertical position, an AC electric field is applied to the suspension as the colloids sediment. To prevent electric field induced movement of the ions in the solvent, whenever an electric field is applied to a suspension, we use a high frequency (>100kHz) AC field. Furthermore, the field must be strong enough to keep the rods vertical while not so strong as to cause the rods to line up end to end due to induced dipoles in the colloids [5]. After some testing, we were able to determine that the fields used in the experiments should have a potential between 5V and 20V in a cell of thickness 0.1mm when the field has a frequency of 1MHz. Figure 3.7 shows confocal images of vertical and freely rotating rod-like colloids.

After the colloids have been allowed to sediment, the sample is flipped and placed under the microscope. The microscope is focused as quickly as possible in order to start the imaging soon after the colloids begin to sediment. In the case where the colloids are initially vertical and then allowed to freely rotate during sedimentation, the electric field in the suspension is turned off after the first time series image.

3.3.2 Experimental Images

For most of the experiments, the images were obtained using a 60x or 100x oil immersion objective, in which case we set the laser scanning confocal microscope (a Nikon

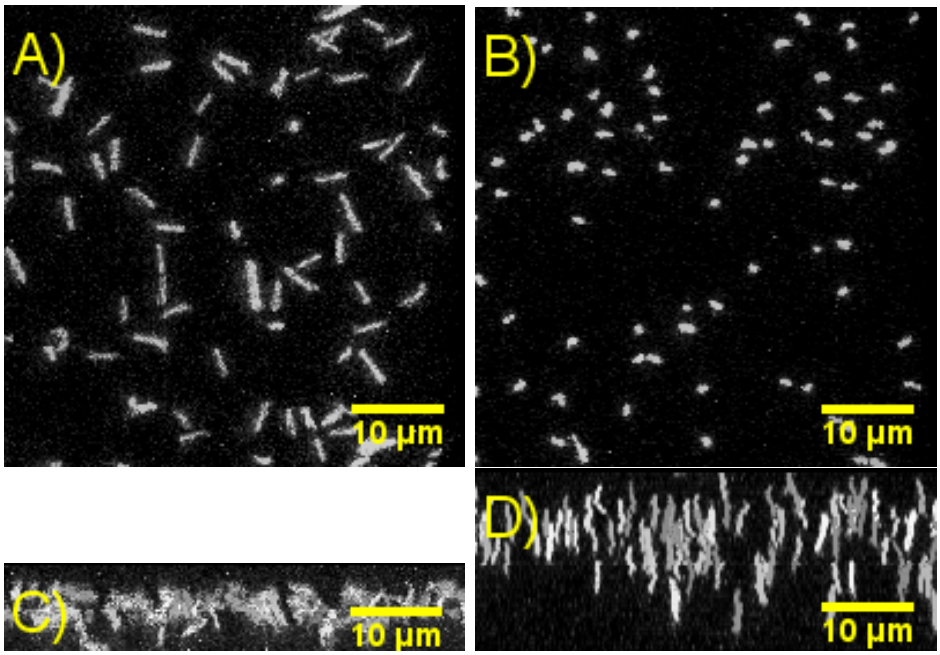

Figure 3.7: Left: xy (A) and yz (C) slices of a confocal image of freely rotating rods. Right: xy (B) and yz (D) slices of a confocal image of rods being held vertical by an electric field. Both sets of images were taken using a 100x objective.

C1 plus, mounted on a Nikon Eclipse 80i) to get pixel sizes of $0.414\mu m \times 0.414\mu m$ and $0.248\mu m \times 0.248\mu m$ respectively. Generally, the size of the z-increments is made to match these pixel sizes as closely as possible in order to get cubic pixels, though occasionally larger z increments are used when we are interested in larger scale effects rather than the dynamics of individual colloids. Sample images taken using the 60x and 100x objectives can be found in figure 3.8 .

One of the main problems that arises when using confocal imaging is that bright spots tend to be blurred more in the z direction compared to the blurring in the

x-y direction, a result of the Point Spread Function (PSF) of the system [23]. It is important to take this effect into account when analyzing confocal images, since a large PSF can make it difficult to determine the orientation of the rod colloids in our system unless the images are extensively pre-processed (e.g. deconvolving the image using an approximation to the PSF). The leftmost examples in figure 3.8 are from one such image, where the PSF more than doubles the diameter of the colloids in the z direction. Thankfully, after some testing of confocal images using various objectives we found that in both the 60x and 100x oil immersion images (center and rightmost columns in figure 3.8), the PSF does not distort the images of the colloids to the point where the orientation cannot be determined to a reasonable degree of accuracy (though the blurring in the z direction is larger for the 60x image), as the blurring in the z direction is less than the rod diameter. Thus, it is not necessary to apply any complicated pre-processing techniques to our images. Because the blurring in the z direction is larger when using a 60x oil immersion objective, most of the systems we analyze are imaged using a 100x oil immersion objective. However, if we are interested in large scale effects (such as the Crowley instability), the larger field of view provided by the 60x image can be useful.

When taking time series images, the upper limit on the frame rate of the time series was set by the time it takes to take a single 3D image. For the microscope

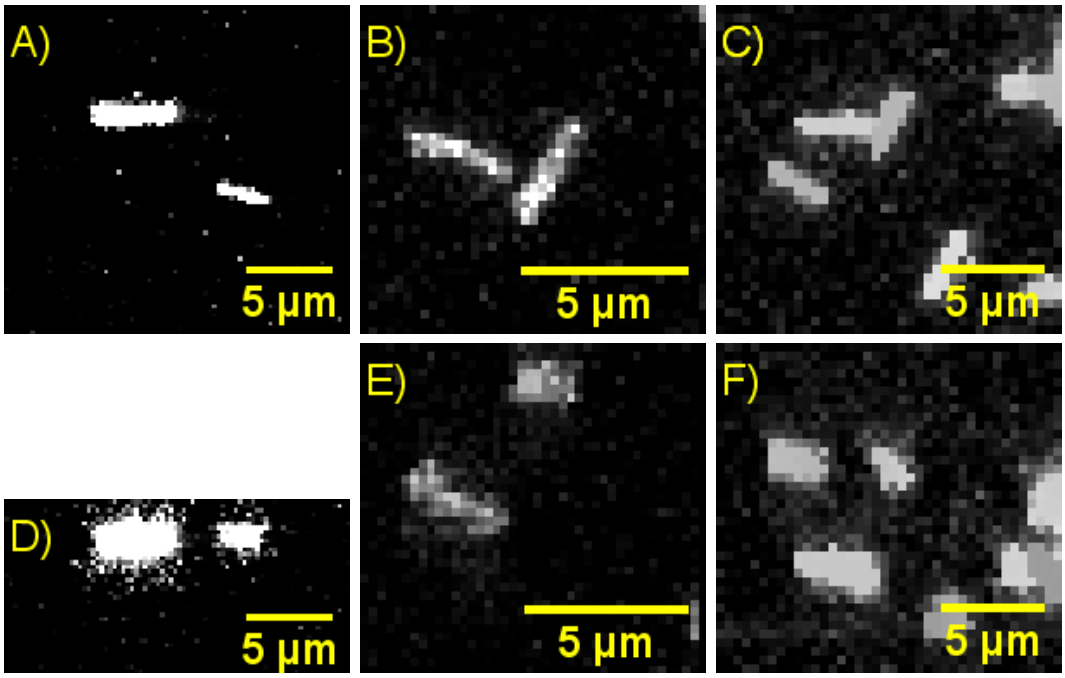

Figure 3.8: Comparison of the Point Spread Functions for three different image set, one in which the PSF is large (A - xy, D - xz) and two sample images taken using a 100x (B - xy, E - xz) and 60x (C - xy, F - xz) oil immersion objectives.

used for these experiments, a single 512 $\times$ 512 pixel image takes 1.20 seconds to take. Thus for a 3D image composed of 40 2D images, the image takes 48 seconds to complete. This means that the frame rate on our time series images is far larger than the timescales of Brownian diffusion ($\sim 10^{-3}$ s). On average the colloids sediment at a rate of around 0.4μm/min for the freely rotating rods and 1.5μm/min for the vertical rods. This means we can get fairly good tracking of the translation motion of the freely rotating rods as they generally only move around half a particle diameter between time frames, though we may not be able to see the details of the rotational

motion of the rods. On the other hand the vertical rods are a bit trickier, moving on average 1.5 rod diameters in the z-direction between time frames, and often more than that in the xy direction meaning it is important to be careful when tracking the movement of the particles, especially given that particles are often within two rod diameters of one another.

Chapter 4

Simulation Theory and Setup

All the simulations in this thesis were performed using the GPU molecular dynamics package HOOMD-Blue (Highly Optimized Object-Oriented Molecular Dynamics) [24], which is capable of simulating Dissipative Particle Dynamics much faster than comparable CPU packages [25]. Both the spherical and rod-like colloids used in our simulations were modeled using the rigid body NVE (microcanonical) integrator provided by HOOMD-Blue [26]. These simulations were in part carried out using GPU facilities provided by SHARCNET and Compute Canada. Sample code used for generating the initial .xml files can be found in Appendix B and sample code used to run the simulations can be found in Appendix C.

In discussions of the simulations all values are given relative to the model unit parameters σ (length), τ (time), m (mass), and ϵ (energy). For convenience, the temperature of the systems is expressed in energy units, so we let the Boltzmann constant $k_b = 1$ in our simulations. The simulations were set up by generating .xml

files containing the initial coordinates of the particles in the fluid and colloids (the Python scripts used to generate these .xml files are included in Appendix B). Note: using integrators other than NVE integrators results in unphysical behaviour as the DPD interaction serves as a thermostat [25].

4.1 Dissipative Particle Dynamics

When looking to model the dynamics of fluids, the scale of the problem in question factors heavily into deciding which methods to use. In microscopic simulations in which the actions of individual particles are important one might use methods like molecular dynamics, while in the macroscopic case one might treat the fluid as a continuum and use partial differential equation methods to model the dynamics of the fluids. In between these two domains are mesoscopic methods which are designed to model fluid interactions with characteristic length scales ranging from 10^{-7} to 10^{-4} meters and characteristic time scales ranging from 10^{-9} to 10^{-3} seconds. Among these methods is Dissipative Particle Dynamics (DPD), which is a coarse-grained method for simulating mesoscale fluid dynamics that is Galilean invariant and conserves momentum [27]. Due to the fact that it is computationally cheaper than microscopic scale simulations and more flexible than alternative mesoscale simulation methods (such

as Lattice-Boltzmann) [27], DPD has seen use in research on a number of mesoscale systems, including studies of polymers and colloids [28–30]. As the practice of using DPD to simulate colloidal systems has been well established, we decided to use it to study the dynamics of colloidal particles during sedimentation.

The simulation systems used in this chapter were based on the system used by Zhou and Schmid [30] in which the simulation system consists of two main parts: fluid particles and colloids. The colloidal particles are modeled as rigid bodies whose surface is covered with DPD interaction points. This method achieves the realization of the no-slip boundary condition on the colloid surface and takes full consideration of hydrodynamic interactions and Brownian motion, making it a good choice for our own systems.

Dissipative Particle Dynamics Theory

In our simulations, the position ($\vec{r}_i$) and velocity ($\vec{v}_i$) of a given fluid particle of mass m_i fluid particle satisfy the equations

$$\frac{d\vec{r}_i}{dt} = \vec{v}_i \quad \text{and} \quad m_i \frac{d\vec{v}_i}{dt} = \vec{F}_{net,i} \tag{4.1}$$

where $\vec{F}_{net,i}$ is the net force on the particle.

DPD forces are the main interaction force between particles, with the DPD force

between particles i and j being defined as

$$\vec{F}_{\mathrm{DPD},ij} = \vec{F}_{\mathrm{D},ij} + \vec{F}_{\mathrm{R},ij} \tag{4.2}$$

with a dissipative component

$$\vec{F}_{\mathrm{D},ij} = -\gamma\omega^2(r_{ij})(\hat{r}_{ij} \cdot \vec{v}_{ij}) \tag{4.3}$$

and a stochastic component

$$\vec{F}_{\mathrm{R},ij} = -\theta_{ij}\sqrt{3}\sqrt{\frac{2k_bT\gamma}{\Delta t}}\omega(r_{ij})\hat{r}_{ij}, \tag{4.4}$$

where r_{ij} is the distance between particle i and particle j, $\hat{r}_{ij}$ is the normalized vector from particle i to particle j, $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$, θ_{ij} is a uniformly distributed random number between -1 and 1, and

$$\omega(r_{ij}) = \begin{cases} (1 - \frac{r_{ij}}{r_{cut}}) & r_{ij} < r_{cut} \\ 0 & r_{ij} \geq r_{cut} \end{cases} \tag{4.5}$$

The drag coefficient γ, temperature T, and cutoff radius r_{cut} have values that depend on the system of interest and are user defined. The stochastic component of the force acts as a thermostat when integrated using an NVE integrator.

DPD fluid

In our simulations we use $\gamma_{\text{colloid-fluid}} = 10.0m/\tau$, $\gamma_{\text{fluid-fluid}} = 5.0m/\tau$, $r_{cut} = 1.0\sigma$, and $k_bT = 1.0\epsilon$, unless otherwise specified. The fluid particles have mass $1.0m$ each and a number density $\rho = 3.0\sigma^{-3}$. Using these settings, Zhou and Schmid found that the fluid has a dynamic viscosity of $\mu = 1.23 \pm 0.01m/\sigma\tau$ [30].

When generating the system initialization .xml files, the fluid particles are distributed randomly in the simulation box. Around each colloid and wall in the system, there is an empty buffer of 0.5σ to 1.0σ which prevents any fluid particles from being placed into high potential regions and flying out of the simulation. When the simulations are started, there is a warmup period ($> 10\tau$) during which the colloids are held fixed by turning off the integration of their movement while the fluid reaches the correct temperature and fills the buffers around the colloids. After the system has warmed up, we unfreeze the colloids and start to apply constant forces to the particles to cause the colloids to sediment.

4.2 Rigid Body Simulations

As mentioned in the preamble to the simulations section, the colloids used in our simulations were modeled using the rigid body procedures provided in the HOOMD-

Blue package [26]. Rigid body methods work by keeping the points in a body at a fixed position with respect to one another and calculating the net forces and torques on the body by summing over the component particles.

In HOOMD-Blue, a rigid body is defined by its center of mass $\vec{R}$, velocity $\vec{V}$, mass M, moment of inertia $\mathbf{I}$, angular momentum $\vec{L}$, and orientation quaternion q. The orientation quaternion is defined as $q = \cos(\theta/2) + \sin(\theta/2)(u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})$, which corresponds to a rotation through an angle θ about the unit vector (u_x, u_y, u_z) and $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are imaginary components defined by $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. Quaternions are used in part because, unlike the Euler angles, it avoids the issue of gimbal lock [31, p. 12]. The force on a body is given by

$$\vec{F_b} = \sum_{k=1}^{N} \vec{f_k} \tag{4.6}$$

and the torque is given by

$$\vec{\tau_b} = \sum_{k=1}^{N} (\vec{r_k} - \vec{R}) \times \vec{f_k} \tag{4.7}$$

where N is the number of component particles in a body, $\vec{f_k}$ is the force on the k^{th} particle, and $\vec{r_k}$ is position of the k^{th} particle. These values are then used to update the body's velocity, position, angular momentum, and orientation using microcanonical NVE integration.

One of the primary concerns when setting up rigid body simulations is how to create the bodies themselves in order to ensure they have the right properties. In the following subsections are descriptions of how the colloid rigid bodies were constructed for our simulations.

4.2.1 Spherical Colloid Setup

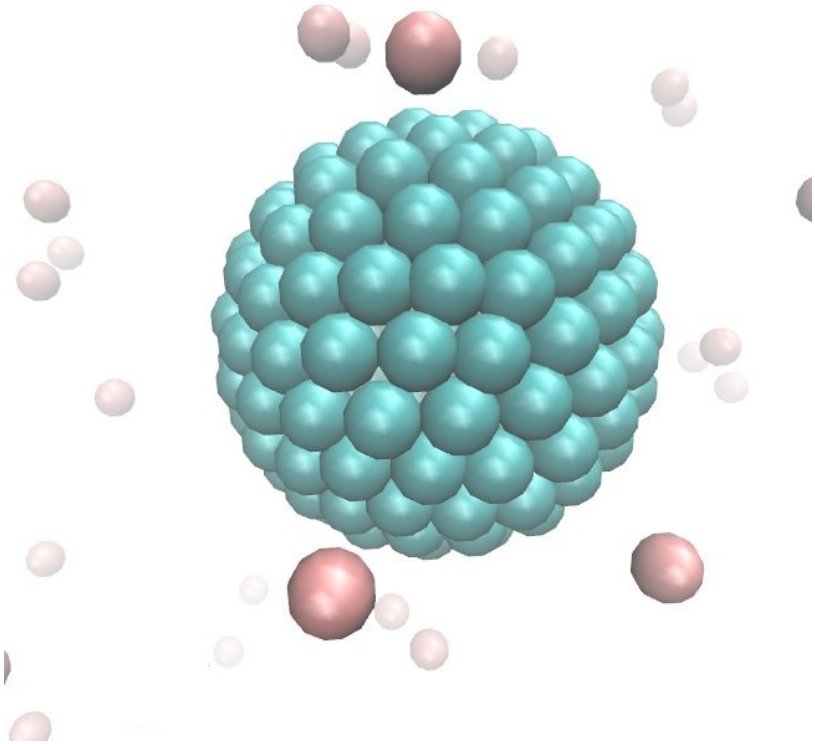

Figure 4.1: An example of a spherical colloid with 162 surface DPD interaction points in blue along with a handful of fluid particles in red. The beads in the image have a diameter of 1.0σ

For our spherical colloid simulations, the colloids were modeled by placing DPD

interaction points on a spherical geodesic grid, an example of which is in figure 4.1, using the method a Python implementation of Teanby's geodesic grid generation algorithm [32]. This gives a set of equally spaced interaction points on the surface of a sphere. In addition, there is a particle at the center of the colloid which interacts with the fluid particles through a shifted Lennard Jones potential

$$V(r) = \begin{cases} 4\epsilon_{LJ}\left[\left(\frac{\sigma_{LJ}}{r-\Delta}\right)^{12} - \left(\frac{\sigma_{LJ}}{r-\Delta}\right)^{6}\right] & r < r_{cut} + \Delta \\ 0 & r \geq r_{cut} + \Delta \end{cases} \tag{4.8}$$

where $\Delta = (d_i + d_j)/2 - 1$, d_i is the diameter of particle i, and $r_{cut} = \sqrt[6]{2}\sigma$. When simulating multiple colloids, the colloids also interact with each other through the shifted Lennard Jones potential between their central particles. The central LJ potential makes the colloids act like hard spheres. In all our spherical colloid simulations, the surface interaction points were gridded on a level 2 geodesic grid, resulting in 162 points. For the single sphere simulations in section 6.1 the particles had radius $R = 3.0\sigma$, mass $M = 100m$, and moment of inertia $I = 360m\sigma^2$ which corresponds to a uniformly distributed mass density.

To get the correct moment of inertia and mass values for a solid sphere using points on the surface of the sphere, we need an extra mass at the center of the colloid. Given that the moment of inertia of a spherical shell is $I_{\text{shell}} = \frac{2}{3}MR^2$ to get a moment of

inertia of $I_{solid} = \frac{2}{5}MR^2 = 360m\sigma^2$ (the moment of inertia of a solid sphere of mass $M = 100m$ and radius $R = 3.0\sigma$) , we require $M_{shell} = \frac{3 \cdot 360}{2 \cdot R^2} m\sigma^2 = 60.0m$. Thus, each surface interaction point was given a mass $m_{shell} = 60.0/162 = 0.370370\ldots m$ and the central particle was given a mass $m_{central} = 40.0m$ to bring the total mass to the required $100m$.

4.2.2 Rod-like Colloid Setup

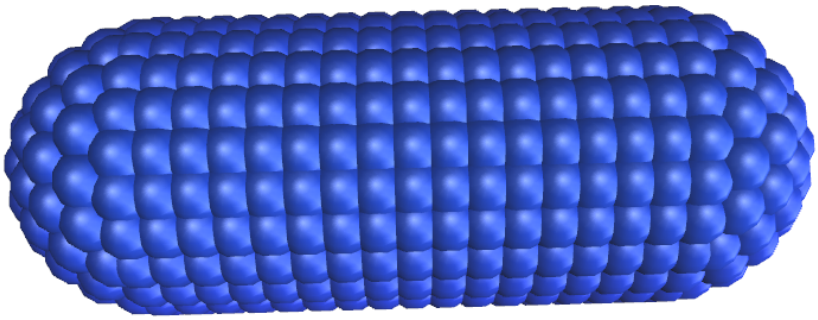

Figure 4.2: An example of a simulation rod-like colloid (length 12, radius 2) composed of spherical caps with a geodesic grid of points and a cylindrical center.

For our rod-like colloid simulations, the colloids used were cylinders with hemispherical caps, similar to our experimental rod-like colloids. The points on the caps are distributed using a geodesic grid, with each cap consisting of points $z \geq 0$ and $z \leq 0$ respectively (for a sphere centered at the origin). Note that both caps include points in the xy plane, which is important when distributing points on the surface

to get the correct moment of inertia. The points on the cylinders are distributed as a series of circles of points wherein each circle is separated by 0.5σ and the spacing between points on the circles is $\approx 0.5\sigma$, see figure 4.2 for an example. In addition, a series of particles are placed in a line on the primary axis of the cylinders which interact with the fluid particles and other colloids through a shifted Lennard Jones potential as described in equations 4.8 with a diameter equal to that of the rods. The space between these central LJ points is set to $0.\bar{3}\sigma$ in order to make the excluded volume of the shifted Lennard Jones potential similar to the shape of the rod. In the case of a rod with length $L = 12\sigma$, we use 36 such central LJ points.

Determining proper values for the moment of inertia for a rod is a bit more complicated than the case of a sphere. Setting the z-axis as the direction of the primary axis of the rod, we can use symmetries in the shape of the rod and the parallel axis theorem to find that the components of the moment of inertia tensor of a solid rod

$$I_{xx}^{\text{solid}} = I_{yy}^{\text{solid}} = 2(I_{xx,hemi}^{\text{solid}} + M_{hemi}^{\text{solid}}(\frac{1}{2}H + \frac{3}{8}R)^2) + I_{xx,cyl}^{\text{solid}} \tag{4.9}$$

$$I_{zz}^{\text{solid}} = 2I_{zz,hemi}^{\text{solid}} + I_{zz,cyl}^{\text{solid}} \tag{4.10}$$

$$I_{xy}^{\text{solid}} = I_{yx}^{\text{solid}} = I_{xz}^{\text{solid}} = I_{zx}^{\text{solid}} = I_{yz}^{\text{solid}} = I_{zy}^{\text{solid}} = 0 \tag{4.11}$$

Given $R = D/2$, $H = L - 2R$, mass density ρ, $M_{hemi}^{\text{solid}} = \frac{2}{3}\pi\rho R^3$, and $M_{cyl}^{\text{solid}} = \pi\rho HR^2$

the moment of inertia tensor of a solid hemisphere about its center of mass is

$$I^{\text{solid}}_{xx,hemi} = I^{\text{solid}}_{yy,hemi} = \frac{83}{480}\rho\pi R^5 \quad \text{and} \quad I^{\text{solid}}_{zz,hemi} = \frac{4}{15}\rho\pi R^5 \tag{4.12}$$

and the moment of inertia tensor of a solid cylinder about its center of mass is

$$I^{\text{solid}}_{xx,cyl} = I^{\text{solid}}_{yy,cyl} = \rho\pi R^2 H(\frac{1}{4}R^2 + \frac{1}{12}H^2) \quad \text{and} \quad I^{\text{solid}}_{zz,cyl} = \frac{1}{2}\rho\pi R^4 H \tag{4.13}$$

Thus the expression for the moment of inertia of a solid cylinder capped with two solid hemispheres is

$$I^{\text{solid}}_{xx} = I^{\text{solid}}_{yy} = \frac{1}{60}\pi\rho R^2(5H^3 + 20H^2R + 45HR^2 + 32R^3) \tag{4.14}$$

$$I^{\text{solid}}_{zz} = \frac{1}{30}\pi\rho R^4(15R + 16H) \tag{4.15}$$

$$I^{\text{solid}}_{xy} = I^{\text{solid}}_{yx} = I^{\text{solid}}_{xz} = I^{\text{solid}}_{zx} = I^{\text{solid}}_{yz} = I^{\text{solid}}_{zy} = 0 \tag{4.16}$$

Again, setting the primary axis of the rod to be the z axis and using symmetry arguments and the parallel axis theorem, we get that a rod shell has moment of inertia

$$I^{\text{shell}}_{xx} = I^{\text{shell}}_{yy} = 2(I^{\text{shell}}_{xx,hemi} + M_{hemi}(\frac{1}{2}H + \frac{1}{2}R)^2) + I^{\text{shell}}_{xx,cyl} \tag{4.17}$$

$$I^{\text{shell}}_{zz} = 2I^{\text{shell}}_{zz,hemi} + I^{\text{shell}}_{zz,cyl} \tag{4.18}$$

$$I^{\text{shell}}_{xy} = I_{yx} = I^{\text{shell}}_{xz} = I^{\text{shell}}_{zx} = I^{\text{shell}}_{yz} = I^{\text{shell}}_{zy} = 0 \tag{4.19}$$

As we must solve for two different equations (I_{zz} and $I_{xx} = I_{yy}$), we require two free

variables. Thus we allow the masses of the caps and the cylinder, M_{hemi} and M_{cyl} respectively, to have different values. As the force and torque on a rigid body are calculated without regard to the mass of individual particles and instead use the total mass and moment of inertia of the body, having the masses of the cap and cylinder particles differ does not affect the movement of the colloid.

Since the moment of inertia tensor of a hemispherical shell about its center of mass is

$$I^{\text{shell}}_{xx,hemi} = I^{\text{shell}}_{yy,hemi} = M_{hemi}\frac{5}{12}R^2 \quad \text{and} \quad I^{\text{shell}}_{zz,hemi} = M_{hemi}\frac{2}{3}R^2 \tag{4.20}$$

and the moment of inertia tensor of a cylindrical shell about its center of mass is

$$I^{\text{shell}}_{xx,cyl} = I^{\text{shell}}_{yy,cyl} = M_{cyl}(\frac{1}{2}R^2 + \frac{1}{12}H^2) \quad \text{and} \quad I^{\text{shell}}_{zz,cyl} = M_{cyl}R^2 \tag{4.21}$$

the moment of inertia tensor for a cylindrical shell capped by two hemispherical shells

$$I^{\text{shell}}_{xx} = I^{\text{shell}}_{yy} = \frac{1}{6}M_{hemi}(5R^2 + 3H^2 + 6HR + 3R^2) + \frac{1}{12}M_{cyl}(6R^2 + H^2) \tag{4.22}$$

$$I^{\text{shell}}_{zz} = \frac{4}{3}M_{hemi}R^2 + M_{cyl}R^2 \tag{4.23}$$

$$I^{\text{shell}}_{xy} = I_{yx} = I^{\text{shell}}_{xz} = I^{\text{shell}}_{zx} = I^{\text{shell}}_{yz} = I^{\text{shell}}_{zy} = 0 \tag{4.24}$$

By equating the inertia tensor components, we solve for M_{hemi} and M_{cyl} and then use that $\rho = \frac{M_{tot}}{4\pi R^3/3 + \pi H R^2}$ to find expressions for the required total masses of the

hemispherical caps and the cylindrical shell

$$M_{hemi} = \frac{3M_{tot}(15H^3 + 104H^2R + 180HR^2 + 96R^3)}{20(7H^2 + 18HR + 12R^2)(3H + 4R)} \tag{4.25}$$

$$M_{cyl} = \frac{3M_{tot}H(25H^2 + 58HR + 36R^2)}{10(7H^2 + 18HR + 12R^2)(3H + 4R)} \tag{4.26}$$

These masses are evenly distributed on the DPD interaction points which approximate a continuous shell. As in the case of the spherical colloid, adding these masses does not give the total mass M_{tot}, so an additional mass $M_{cent} = M_{tot} - M_{cyl} - 2M_{hemi}$ is added to the center of mass of the particle in order to give the colloid the correct mass without changing the moment of inertia.

Typically, we use rods with a diameter of $D = 4.0\sigma$, an end to end lengths of $L = 12.0\sigma$, a mass of $M_{tot} = 500m$, and a moment of inertia equal to the moment of inertia of a solid rod with uniform mass density and total mass $500m$ (equations 3.6 to 3.8). This gives a mass per interaction point of $0.300720m$ for the 425 points comprising the cylinder and $0.924387m$ for the 178 points comprising the hemispherical caps. This distribution of points produces a center of mass and total mass that are within 0.01% of the desired values and the moment of inertia tensor components differ from the desired values by 5%. These discrepancies are the result of the fact that we are attempting to model a continuous shell using discrete points and can be reduced by increasing the number of surface interaction points.

Chapter 5

Experimental Results and Discussion

5.1 Experiments Performed

In table 5.1, we list some information on the experiments we performed. Unfortunately, not all the experiments can be used for analysis. The 40x oil image sets had large point spread functions that make it difficult to get information on the orientations of the colloids, while image sets 5, 6, and 7 cannot be used because the colloids in the image were stuck to the glass at the top of the sample. Image sets 9, 11, 12, 13, 15, and 16 were taken using large z-steps so that we could capture larger scale effects such as clustering, however this means that we cannot get good orientation data from them, as the image pixels are 2 to 4 times longer in the z direction compared to the xy directions. In addition, the clustering effects we observe in our systems occur at the bottom of the sedimenting rods and only after some time has elapsed. Thus, the clustering of colloids can only be clearly observed in longer experiments (image sets

#	mm/dd/yy	Type	Objective	XY size (μm)	Z size (μm)	Time	Colloids
1	05/04/15	FR - 1	100x Oil	128 x 64	6 to 40	1:19hr	$\sim$ 350
2	05/04/15	FR - 2	40x Oil	317.4 x 158.7	15.5 to 88.04	3:03hr	$\sim$ 400
3	05/06/15	FR - 1	40x Oil	316.8 x 316.8	15.6 to 48.6	1:13hr	$\sim$ 700
4	05/06/15	FR - 2	40x Oil	316.8 x 316.8	30.6 to 100.8	3:03hr	$\sim$ 4000
5	08/02/15	FR - 1	100x Oil	128 x 128	6.25 to 10	0:34hr	$\sim$ 40
6	08/02/15	FR - 2	100x Oil	128 x 128	10	0:29hr	$\sim$ 40
7	08/02/15	IVR	100x Oil	128 x 128	10	0:08hr	$\sim$ 40
8	08/02/15	FR - 3	100x Oil	128 x 128	10	0:21hr	$\sim$ 100
9	08/02/15	IVR - 2	60x Oil	209.9 x 209.9	8.2	0:13hr	$\sim$ 130
10	08/05/15	FR - 1	100x Oil	128 x 128	10	0:14hr	$\sim$ 400
11	08/05/15	FR - 1	60x Oil	209.9 x 209.9	34	0:31hr	$\sim$ 800
12	08/10/15	IVR - 1	100x Oil	128 x 128	20	0:13hr	$\sim$ 200
13	08/11/15	IVR - 1	60x Oil	209.9 x 209.9	20 to 27	0:26hr	$\sim$ 450
14	08/11/15	VR - 1	100x Oil	128 x 128	20	0:13hr	$\sim$ 550
15	08/11/15	VR - 2	100x Oil	128 x 128	27	0:33hr	$\sim$ 500
16	08/12/15	VR - 1	100x Oil	128 x 128	20	0:14hr	$\sim$ 425
17	08/12/15	IVR - 2	100x Oil	128 x 128	10	0:26hr	$\sim$ 250

Table 5.1: Some information on the experiments performed. FR - an experiment with freely rotating rods, VR - An experiment where the rods are held vertical by an electric field, IVR - An experiment where the rods are initially vertical, but then allowed to freely rotate

1, 2, 3, 4, 11, and 15).

5.2 Péclet number

As noted in section 2.1.1, one important quantity to know when dealing with colloidal systems is the Péclet number of the colloids. To find the Péclet number for our suspensions, we used the formula

$$\mathrm{Pe} = \frac{uR_H}{D} \tag{5.1}$$

where u is the speed of the particle, R_H is the hydrodynamic radius, and D is the diffusion constant. Since our systems have very low Reynolds numbers

$$\mathrm{Re} = \frac{2\rho u R_H}{\mu} < 10^{-10} \tag{5.2}$$

where ρ is the density of the solvent and μ is the dynamic viscosity, we assume that the drag force on the particles is described by the Stokes drag. Note, since there are many colloids in the systems, this is not quite accurate due to the mutual drag reduction of particles when they are near one another. Thus using the Stokes-Einstein equation, we can find the diffusion constant D using the formula

$$D = \frac{k_B T}{6\pi\mu R_H} \tag{5.3}$$

where μ is the shear viscosity. Thus, we express the Péclet number as

$$\mathrm{Pe} = \frac{6\pi\mu u R_H^2}{k_B T}. \tag{5.4}$$

Using equation 2.4 in section 2.1.1 and the rod colloid dimensions in table 3.1, we calculate the approximate hydrodynamic to find $R_H = 0.788\mu\mathrm{m} = 1.27R_S$, assuming the rod colloids are the shape of a cylinder with hemispherical caps.

In order to calculate the Péclet number of our colloids, we need to know the terminal velocity of the particles u. To find this, we plot the mean z-coordinate of the colloids in a number of experiments (vertical rods - experiments 14 and 16, freely rotating - experiments 8, 10, 12, and 13) over time in figure 5.1 (Note: positive z is in the direction of gravity in figures 5.1 and 5.2). Since the mean z coordinate increases linearly with time for all the systems at early time periods, we perform a linear fit on each plot to determine the average sedimentation rate of the systems. The results of these linear fits are listed in table 5.2. We find that even among experiments of the same type there is a significant amount of variation in the sedimentation rates of the colloids. For freely rotating colloids, the sedimentation rates vary from 0.00562μm/s to 0.0142μm/s and for the vertically held colloids, we get sedimentation rates of 0.0154μm/s and 0.0230μm/s. Note, the plateauing of some of the plots at later times is likely the result of colloids exiting the frame of the image as they

sediment. This is demonstrated in figure 5.3, where we see that the distribution of z coordinates in the August 11 vertical rod experiment is cut off at the bottom edge of the image resulting in a shift in the average of the z coordinates to values lower than the actual average z coordinate. Note, this effect is entirely a result of the imaging and not because the colloids have reached the bottom of the sample, since the sample is $\sim 100\mu$m deep and experiment 11 images only captures a range of 20μm.

We do not know the cause of the discrepancies in the measured sedimentation rates of the various experiments. However, since the two freely rotating colloid experiments that yield lower sedimentation rates (August 2 and August 5) cover a smaller range in the z direction compared to the experiments which yield higher sedimentation rates (August 10 and August 11), the discrepancy between the two sets of experiments may be due to rods missing from the statistics of the experiments with smaller fields of view. Performing further experiments would allow us to get a better estimate for the sedimentation speed of our colloids. Nonetheless, based on the sedimentation rates listed in table 5.2, it seems as though the rod-like colloids sediment faster when held in a vertical orientation. If this is the case, it is likely because the surface area facing in the direction of motion is smaller in the case of the vertical rods.

Using the values for the sedimentation rate we found and equation 4.3, we cal-

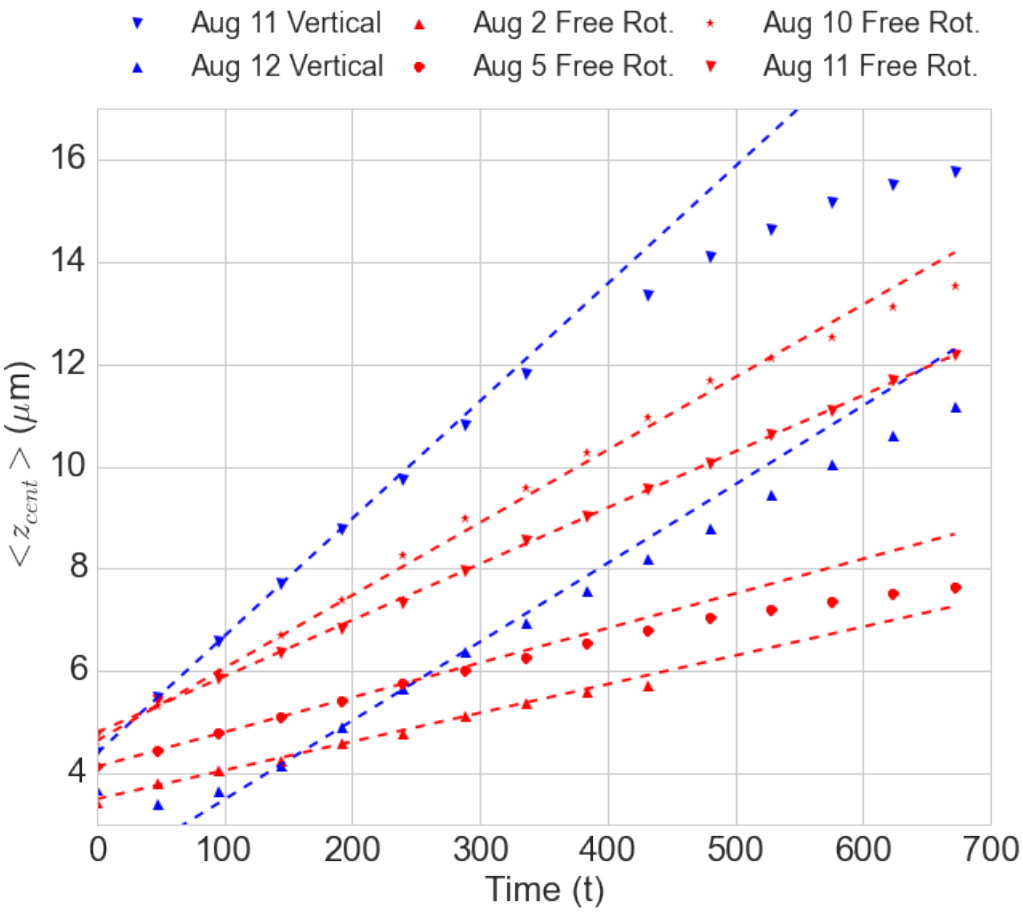

Figure 5.1: Mean z coordinate of the rod-like colloids over time for various experiments. Red points - experiments wherein the colloids rotate freely (including those in which they begin vertical and are subsequently allowed rotate). Blue points - experiments wherein the colloids were held vertical by an electric field. Information on the linear fits to the data can be found in table 5.2.

culate the approximate Péclet numbers of the experiments, which are listed in the rightmost column of table 5.2. In all our experiments Pe >1, which indicates that hydrodynamics are the most important contribution to the dynamics of the colloids, however the fact that all the calculated Pe are less than 10 means that Brownian motion is still quite significant. Notably, we find that the Péclet numbers of the vertical colloid experiments were higher than those of the freely rotating colloid experiments, meaning hydrodynamics tend to be more dominant when the rods are held in a vertical orientation.

In figure 5.2, the standard deviations of the z coordinates of the colloids are plotted over time. In this context, the standard deviation is used as a measure of how spread out the colloids are in the z direction. We observe that at early times the standard deviation also increases linearly with time, and plateau at later times as the colloids exit the frame of the image. The values of the slope and intercept obtained by performing linear fits to the data at early times are listed in table 5.2. From these values, it can be seen that colloids that sediment faster also tend to spread out faster as well.

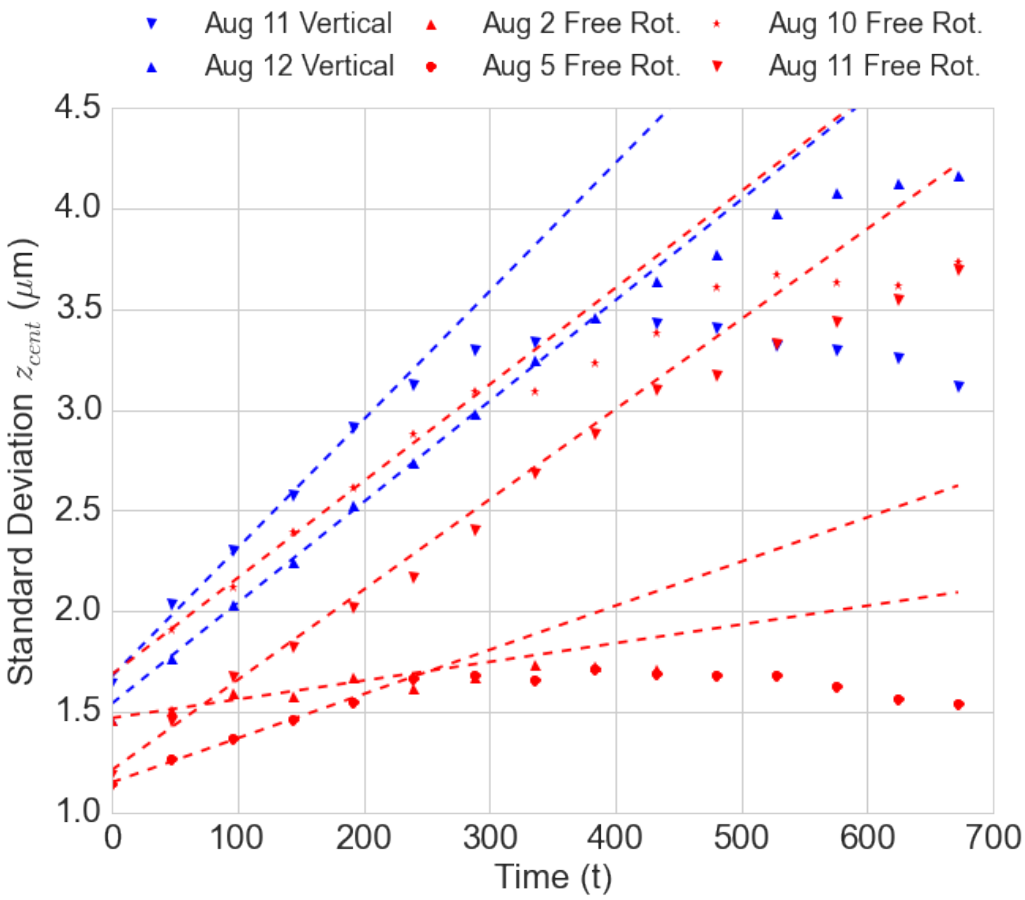

Figure 5.2: Standard deviation of the z coordinate of the rod-like colloids over time for various experiments. Red points - experiments wherein the colloids rotate freely (including those in which they begin vertical and are subsequently allowed rotate). Blue points - experiments wherein the colloids were held vertical by an electric field. Information on the linear fits to the data can be found in table 5.2.

Experiment	Mean z_{cent} fit			Standard Deviation z_{cent} fit			T	Pe
	$m(\mu$m/s)	$b(\mu$m)	Residual	$m(\mu$m/s)	$b(\mu$m)	Residual	$(^\circ C)$	
Aug 11 VR	0.0230	4.38	0.00053	0.00637	1.676	0.0042	23.9	5.8
Aug 12 VR	0.0154	1.94	0.00045	0.00501	1.540	0.00082	22.2	4.3
Aug 2 FR	0.00562	3.49	0.00941	0.000930	1.469	0.00177	23.3	1.5
Aug 5 FR	0.00678	4.12	0.000172	0.00219	1.15	0.000184	24.5	1.7
Aug 10 FR	0.0142	4.63	0.00107	0.00480	1.684	0.00113	24.9	3.5
Aug 11 FR	0.0110	4.79	0.00906	0.00448	1.210	0.00355	22.3	3.1

Table 5.2: Information on the linear fitting ($< z >= mt + b$) performed on the mean and standard deviations of the z coordinates of our sedimenting rod colloids in various experiments and the Péclet numbers we calculate for the systems using the sedimentation rates. Refer to figures 5.1 and 5.2 to see the data and fitted lines. (FR - Freely Rotating, VR - Vertical Rods)

5.3 Rotational Diffusion

As we are dealing with anisotropic, axially symmetric particles, the systems we are testing have two additional rotational degrees of freedom. This introduces complexity to the dynamics of our system that we were interested in testing. One test we performed was to do an experiment in which rod-like colloids initially sedimented in a vertical orientation under an electric field, which was then turned off after the colloids had sedimented for 00:19:18 hours (sedimented $\approx 20\mu$m). By doing this we could observe the onset of disorder in the orientations of the colloids.

In figure 5.4 we have plotted the mean $\cos^2(\theta)$ of the colloids over time after the electric field was turned off (corresponding to $t = 48$s on the graph) where θ is the

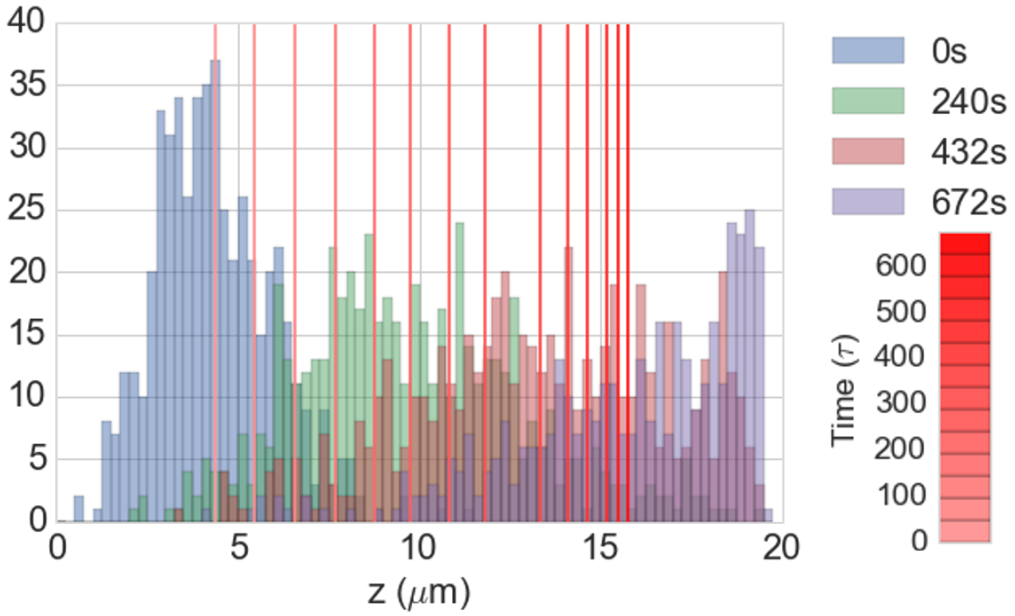

Figure 5.3: Distributions of the z coordinate of experimental colloids in the August 11 vertical rod experiment at selected time steps. Red vertical lines indicate the mean value of the z coordinate at times at each time frame. For later times, the distribution becomes cut off at the bottom of the image ($z = 20\mu$m)

polar angle measured from the z-axis. At this time, the colloids are in the middle of sedimenting so the colloids are fairly dispersed and are clustered. The mean $\cos^2(\theta)$ decays exponentially to its average value in the case of a set of completely disordered colloids ($< \cos^2(\theta) >= 1/3$) after around 350 seconds. For a particle with hydrodynamic radius R_H in a solvent with temperature T and dynamic viscosity μ, the Einstein-Smoluchowski-Stokes relation says that the rotational diffusion coefficient

such a particle is

$$D_r = \frac{k_B T}{8\pi\mu R_H^3} \tag{5.5}$$

where k_B is the Boltzmann constant. Using the hydrodynamic radius $R_H = 0.618\mu$m, we estimate the diffusion constant to be $D_r = 0.0038\text{rad}^2/\text{s}$. Since the rotational Péclet number is defined to be

$$Pe_r = \frac{\dot{\gamma}}{D_r} \tag{5.6}$$

where $\dot{\gamma} \approx v/R_H$ we get $Pe_r \approx 2.3$, which indicates that the disordering of the orientations of the colloids is largely the result of chaotic hydrodynamic motion resulting from the many body nature of the system, while Brownian motion has a significant, but ultimately weaker contribution to the disordering.

5.4 Clustering of Colloidal Particles

Based on qualitative examination of the confocal images of our experiments, we observe that as the colloids sediment instabilities form at the leading edge of the sedimenting colloids, which eventually form clusters and later columns of sedimenting colloids, leaving voids in the xy profiles of the images. This is observed in both the

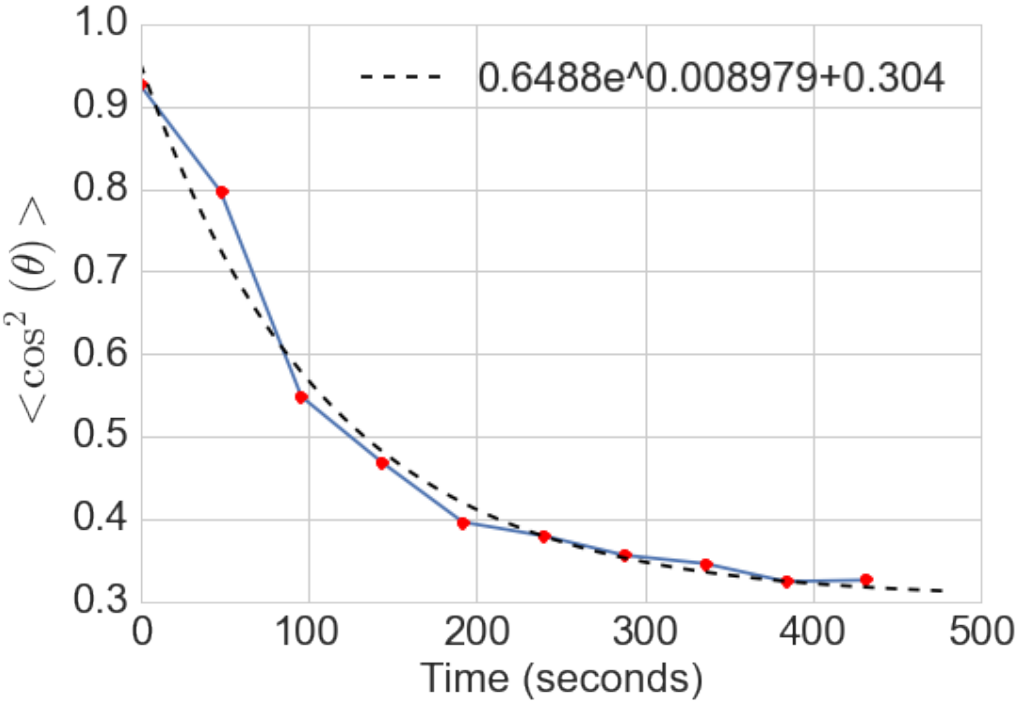

Figure 5.4: Mean $\cos^2(\theta)$ over time after the electric field is switched off in a sedimenting colloid experiment, where θ is the polar angle measured from the z axis. An exponential function is fitted to the data (dashed black line).

freely rotating and vertically held rod-like colloid systems. This is illustrated in figures 5.5, 5.6, and 5.7. The xy profiles of the images show rods in the bottom portion of the full 3D images, since the clustering tends to occur at the leading edge of the sedimenting colloids. The development of instabilities and clusters in the leading edge of the colloids can be seen in the later time images in figure 5.6. At longer times, these clusters eventually become columns of rods, as can be seen in figure 5.7. Generally we find that these instabilities and clusters are on the order of 5 to 10 rod lengths

in width. Based on the images in figure 5.5, we see that the colloids seem to cluster more in the system in which the rods are held vertical by a field, since the voids in the colloids are larger and clearer.

In order to quantify the degree of clustering of the colloids in our experimental images, we generated Voronoi diagrams for the centroids of the colloids. For (x, y) coordinates of the centroids of the colloids in an image, the Voronoi diagram partitions the image into regions such that for each centroid there is a corresponding region such that any point in that region is closer to its associated centroid than any other. The Voronoi diagrams associated with the xy profiles in figure 5.5 are shown in figure 5.8. Our Voronoi diagrams were generated using Qhull [33].

By examining the distribution of the areas of the polygons generated by the Voronoi diagram (excluding any that have vertices outside of the image), we can get an idea of how clustered the rods are. When the rods are clustered, we expect the histogram of the areas to be shifted left and to have a longer tail compared to the histogram of the areas for the Voronoi diagram of randomly distributed points. To analyze our images, we first divided the colloids based on whether they were above or below the median z coordinate of the colloids in the system. We then generated the Voronoi diagrams for the two halves of the 3D image, found the areas of the polygons, and plotted the histograms of the areas divided by the mean area (denoted

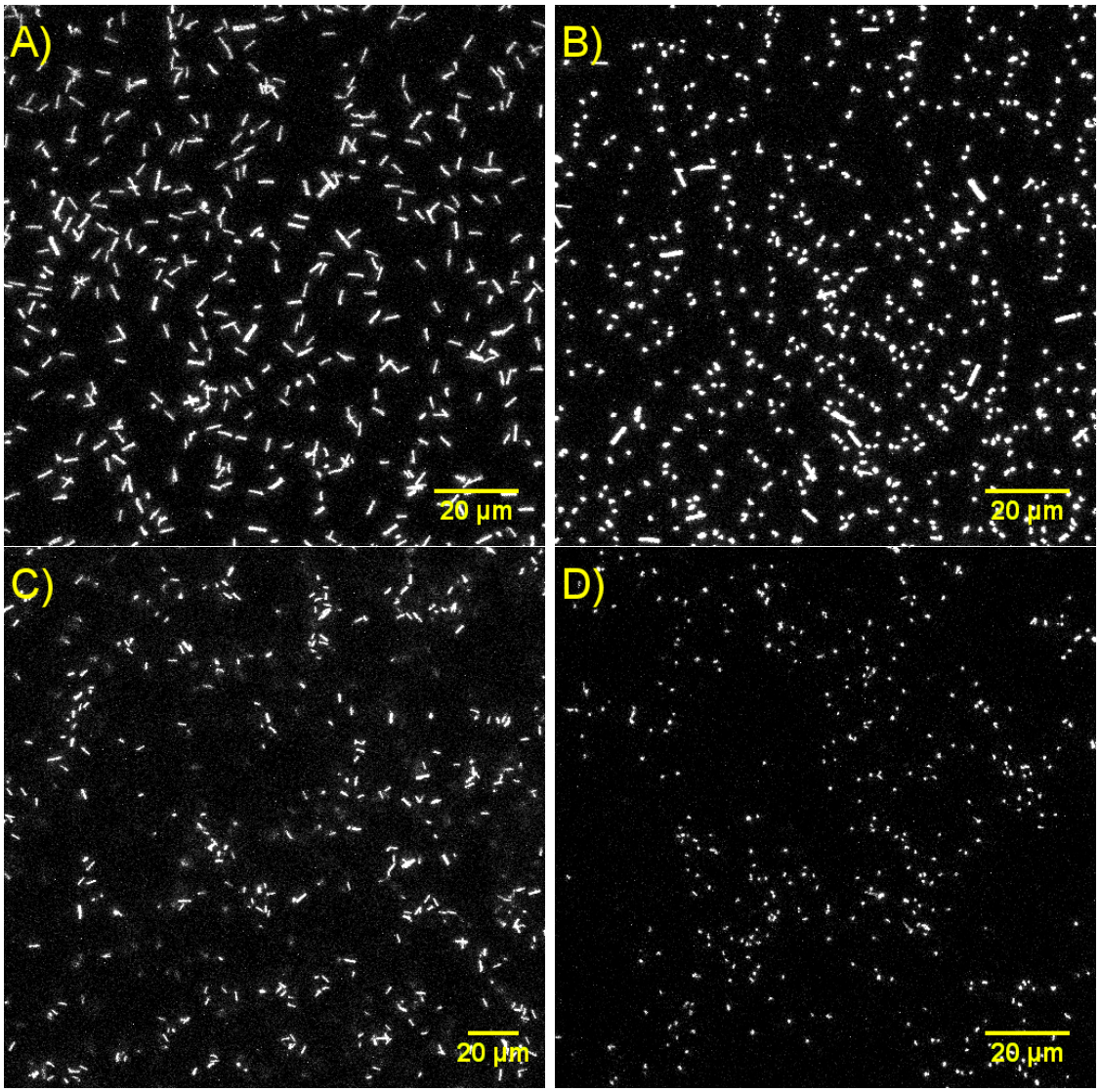

Figure 5.5: Images of colloids clustering during sedimentation for freely rotating colloids (A - experiment 10, t =00:02:45hr and $< z > \approx 4\mu$m, C - experiment 11, t = 00:21:58hr and $< z > \approx 12\mu$m) and colloids held vertical by electric field (B - experiment 14, t = 00:01:45hr and $< z > \approx 4\mu$m, D - experiment 15, t = 00:21:05hr and $< z > \approx 30\mu$m).

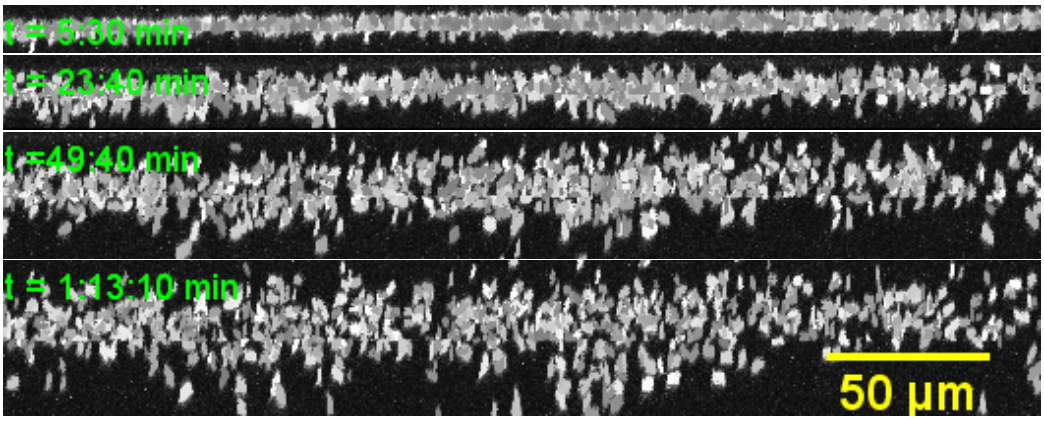

Figure 5.6: xz profiles of images taken during experiment 3 at different times showing the development of instabilities in the sedimenting colloids.

$A_{vor}/ < A_{vor} >$) in order to normalize against the concentration of the colloidal particles so that we can compare the distributions to those generated from a set of randomly placed points.

In figure 5.10 and 5.11, we have plotted the $A_{vor}/ < A_{vor} >$ for experiments 11 (freely rotating colloids) and 15 (vertical colloids) soon after the colloids begin sedimenting in the experiment and after the colloids have sedimented for about 20 minutes (through $\sim 1/3$ of the cell's thickness for the vertical rods and $\sim 1/8$ of the cell's thickness for the freely rotating rods). Along with the histograms, we have plotted the Kernel Density Estimation (KDE) function for the $A_{vor}/ < A_{vor} >$ histogram of the Voronoi diagram for a randomly distributed set of points which was calculated using Scipy [34]. At early times, the histograms of top and bottom halves of the points are similar to one another as well as matching fairly well with the KDE for a

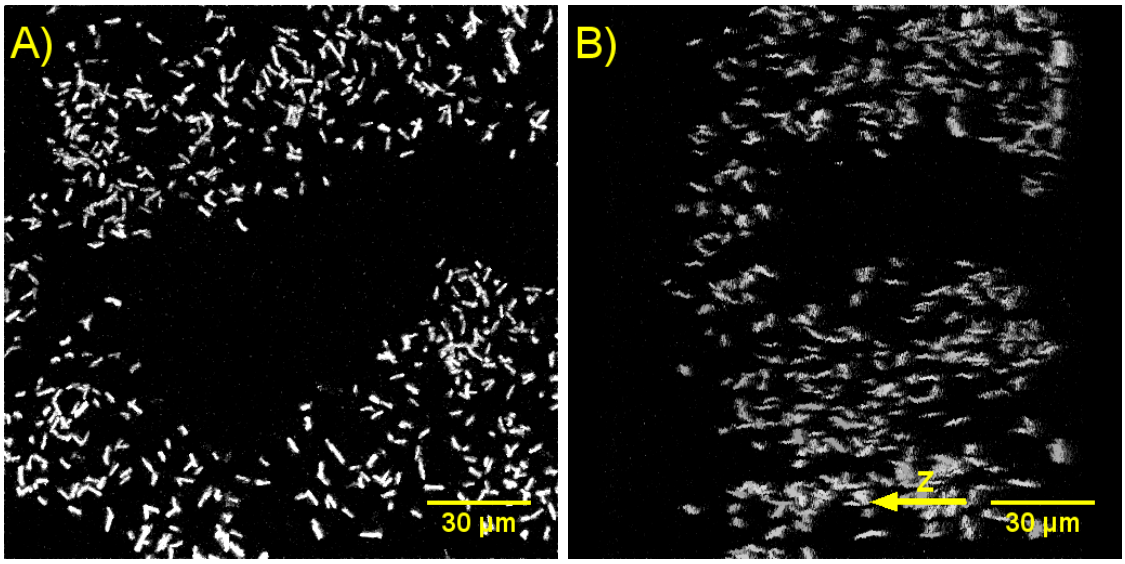

Figure 5.7: xy (A) and yz (B) profiles of clustering colloids at long times take from experiment 1 in table 5.1 at after 1:19:00hr.

random distribution. This is because the colloids have just started sedimenting, so we do not see any clustering and thus the colloids are fairly randomly distributed. In later time histograms, we see that the histogram for colloids in the upper half of the system is still similar to the KDE for a random distribution. However, the histogram for colloids in the lower half of the system has shifted to the left considerably and has a longer tail than the upper half histogram. That this shift only occurs for colloids below the median z coordinate supports the observation that the formation of colloid clusters occurs near or at the leading edge of the sedimenting colloids.

Based on qualitative assessments of the experimental images (such as those in figure 5.5), it would seem that the colloidal particles tend to cluster more closely

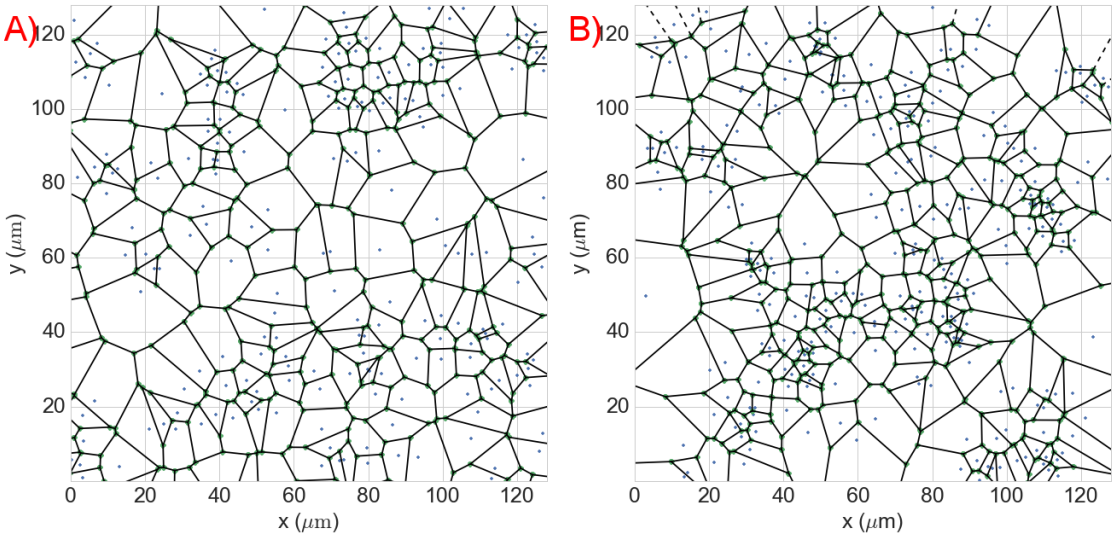

Figure 5.8: Voronoi diagram of colloids clustering during sedimentation for freely rotating colloids (A - experiment 11, t = 00:21:58hr and $< z > \approx 12\mu$m) and colloids held vertical by electric field (B - experiment 15, t = 00:21:05hr and $< z > \approx 30\mu$m).

when they are held vertical by an electric field than when they are allowed to rotate freely. To quantify this effect, we plotted the histograms of A_{vor} for the points below the median z coordinate in figure 5.9. In this figure, data from multiple time steps (experiment 11 t=00:21:58 to 00:30:58 hr for freely rotating colloids and experiment 15 t = 00:19:05 to 00:33:05hr for vertical colloids) were used to improve the statistics of the histograms. From these histograms we can see that both histograms peak at similar values, but that the freely rotating rod histogram is broader and has a lower peak. This may support the observation that the rod colloids tend to cluster together more tightly when held vertical, however this difference in clustering may also be

due to the higher colloid concentration in vertical colloid images ($\rho_{vertical} \approx 0.012$ colloid/μm^2, $\rho_{free} \approx 0.009$ colloid/μm^2). Performing further experiments using suspensions with the same colloid concentration will be needed to determine the cause of the differences in the A_{vor} histograms. In addition, we find that the histograms of $A_{vor}/ < A_{vor} >$ (figure 5.9 inset) for the two data sets are very similar, which may be indicative of similarities in the "structure" of the clusters.

To quantify how clustered the colloids are, we can use the skewness of the Voronoi area histograms. The skewness is a measure of the asymmetry in a distribution and increases as the tail grows longer in the right direction and the distribution becomes more concentrated. Since more clustered images have histograms that are more concentrated near zero, we expect the skewness to increase as the colloids become more clustered. One issue that arises when using the skewness is that it is highly sensitive to outliers in the data, as it is calculated using the third order central moment. In order to suppress large variations in the skewness due to outliers, we remove any A_{vor} larger than $< A_{vor} > +3\sigma$ where σ is the standard deviation of A_{vor}. In figure 5.12, the skewness of the A_{vor} is plotted over time for three of our experiments in which we are able to observe the formation of instabilities and clusters in the sedimenting colloids (experiments 1, 2, and 3 in table 5.1). At early times May 4 experiment 2 and May 6 experiment 1 have high skewness values which may be caused by the fact

that the sample was slightly tilted. As a result, when the colloid centroids are split based on z coordinate, there is a gradient in the density of the (x, y) points resulting in a wide A_{vor} distribution. At later times, the colloids become sufficiently spread out such that this is no longer an issue. At these times we see that the skewness of the A_{vor} increases steadily for all three experiments, likely corresponding to the increasing clustering of the colloids. However, the rates at which the skewness of the three experiments differ considerably. Again, further experimentation would be required to determine the cause of the differences, but possible causes could be statistical issues arising from the differing fields of view in the image and differences in the concentrations of the colloids in the suspensions. Some statistical information on the Voronoi area analysis of the experiments in which we observe clustering is listed in table 5.3.

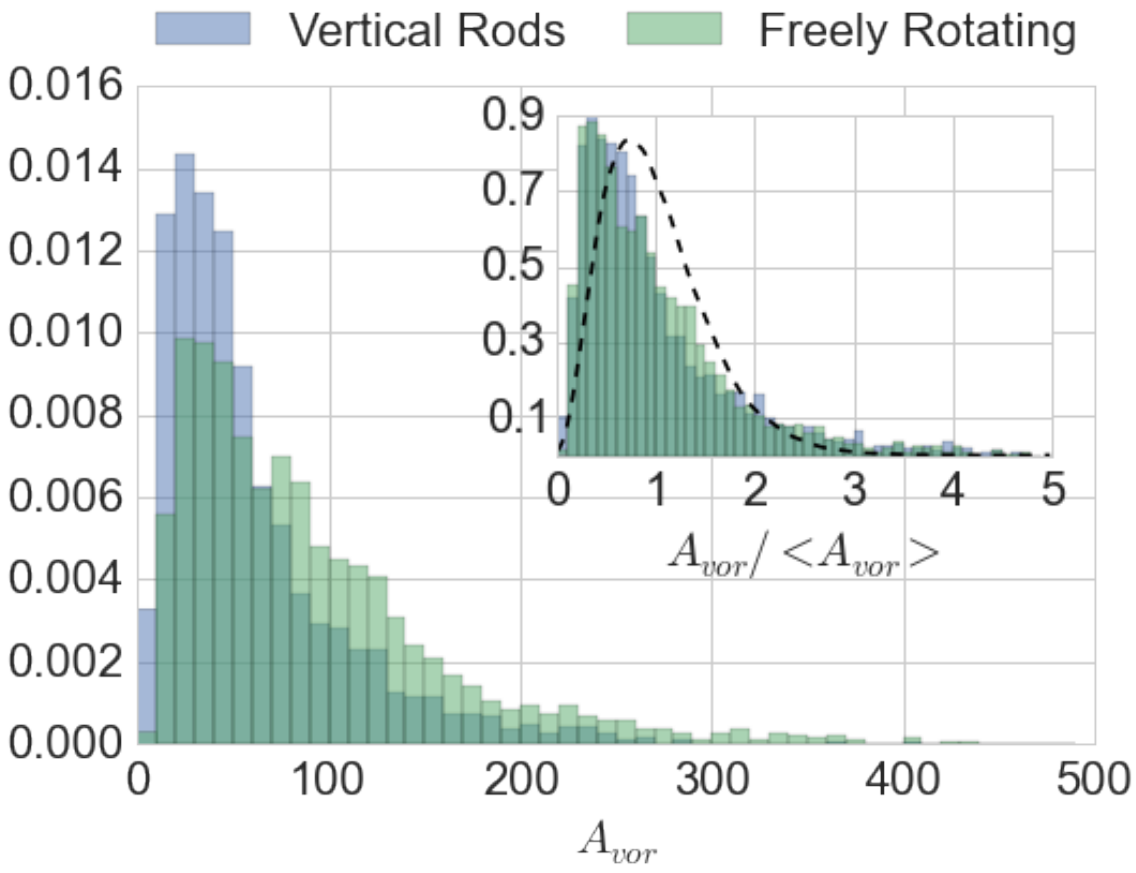

Figure 5.9: Normalized histograms of Voronoi polygon areas A_{vor} for points below the median z coordinate. Multiple time steps were used to improve the statistics for both histograms. Vertical rod histograms are in light green, freely rotating rods are in blue, and the overlap is in dark green. Inset: Histograms of A_{vor} divided by the mean area $< A_{vor} >$. The black dashed line is the Kernel Density Estimation for the $A_{vor}/ < A_{vor} >$ for a random distribution of points.

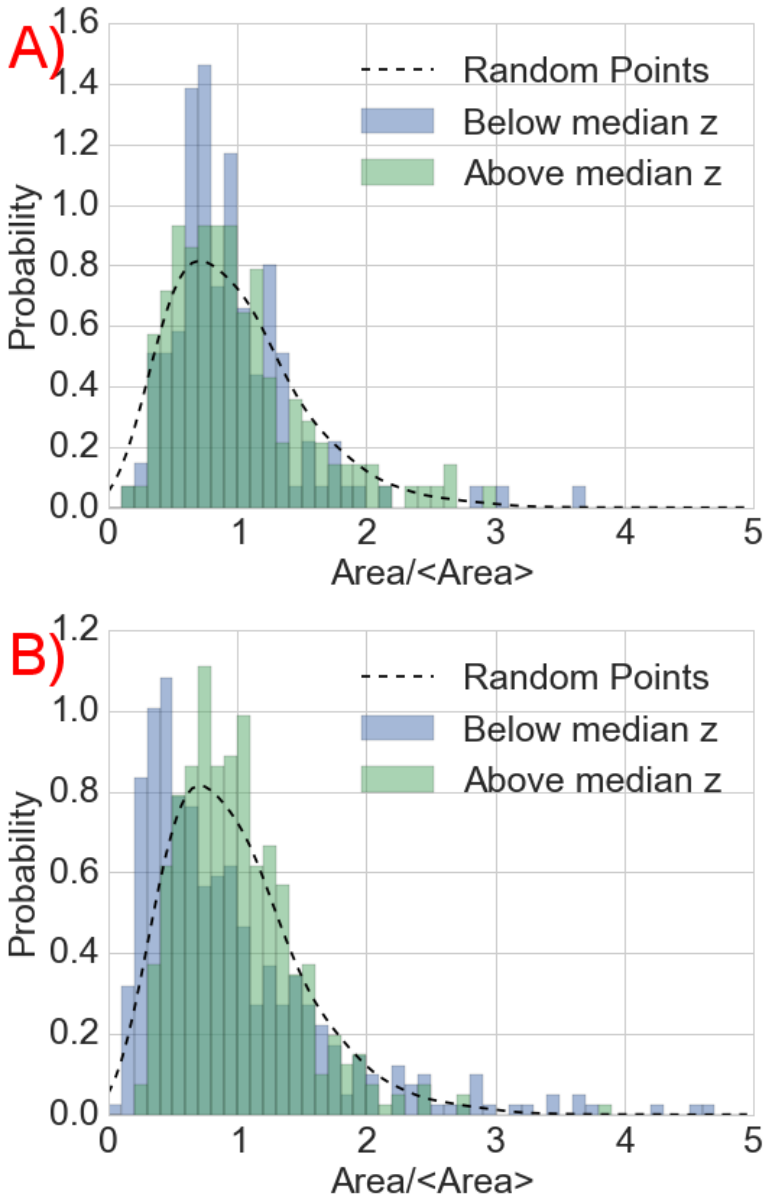

Figure 5.10: Normalized histograms of $A_{vor}/ < A_{vor} >$ for a freely rotating colloid experiment initially (A - experiment 10, t =00:02:45hr and $< z > \approx 4\mu$m) and after some time (B - experiment 11, t = 00:26:58hr and $< z > \approx 14\mu$m). The data are divided based on whether the colloids are above (green) or below (blue) the median z coordinate of the colloids (dark green - overlap of the histograms). The black dashed line is the Kernel Density Estimation for the areas of the Voronoi diagram for a random distribution of points.

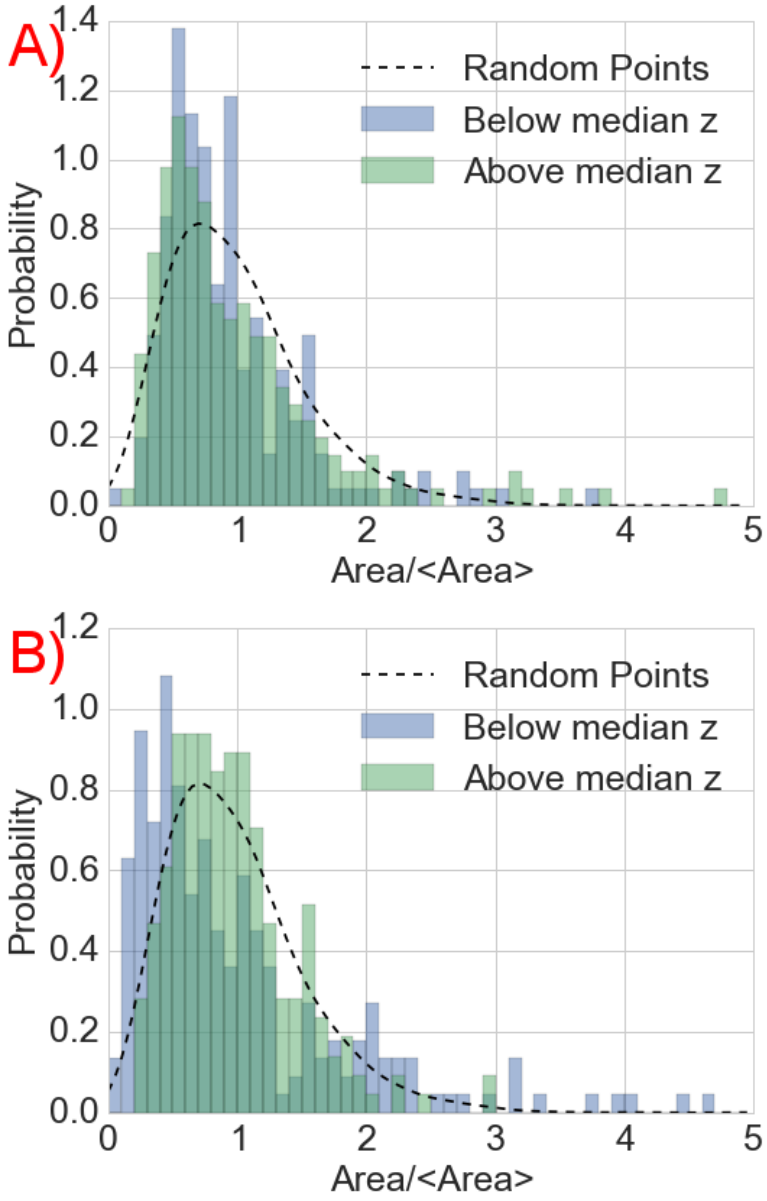

Figure 5.11: Normalized histograms of $A_{vor}/ < A_{vor} >$ for a vertical colloid experiment initially (A - experiment 14, t = 00:01:45hr and $< z > \approx 4\mu$m) and after some time (B - experiment 15, t = 00:21:05hr and $< z > \approx 30\mu$m). The data are divided based on whether the colloids are above (green) or below (blue) the median z coordinate of the colloids (dark green - overlap of the histograms). The black dashed line is the Kernel Density Estimation for the areas of the Voronoi diagram for a random distribution of points.

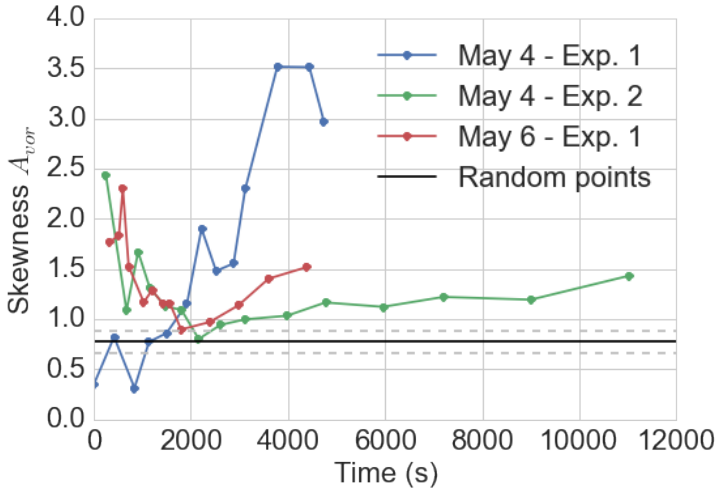

Figure 5.12: Skewness of A_{vor} over time for experiments 1 (blue), 2 (green), and 3 (red) in table 5.1. The black solid line shows the A_{vor} skewness for a set of randomly distributed points averaged over 80 different random sets. The gray dashed lines are this average value plus/minus the standard deviation of the skewnesses of the different random sets.

System	Standard Deviation A_{vor} (μm^2)	Mean A_{vor} (μm^2)	Skewness	Area Density (μm^{-2})
Random Distribution	15.8	30.9	0.78	0.026
FR - Exp. 1 (t = 3120s)	38.4	34.6	2.30	0.015
FR - Exp. 2 (t = 9000s)	179	216	1.19	0.003
FR - Exp. 3 (t = 4390s)	318	413	1.52	0.0016
VR - Exp. 15 (t = 1318 to 1858s)	44.4	57.2	1.57	0.012
FR - Exp. 11 (t = 1145 to 1985s)	64.8	87.7	1.45	0.009

Table 5.3: Skewness of the distributions of Voronoi areas in different systems (FR - Freely Rotating, VR = Vertical Rod).

Chapter 6

Simulation Results and Discussion

The single particle simulations with system sizes of less than around 100000 particles discussed in this section were run on a Nvidia GTX 780 GPU while larger simulations (multi-colloid) were run on SHARCNET's monk GPU cluster, which uses Nvidia Tesla M2070 GPUs.

6.1 Simulations of Spherical Colloids

6.1.1 Diffusion of Spherical Colloids

As noted in the Chapter 4, we are basing our method for modeling our colloids on the method used by Zhou and Schmid in [30]. Thus, it is naturally important to ensure that we can replicate the behaviors observed by Zhou and Schmid (who performed simulations using ESPResSo) in our simulations performed using HOOMD-Blue. To do this, we used a single colloidal particle with properties as specified in table 6.1 in a

fluid with temperature $k_BT = 1.0\epsilon$. The dynamics of this colloid are then simulated in a DPD fluid with parameters as specified in section 4.1 for a simulation box size L with periodic boundary conditions.

Sphere Properties	Radius	3.0σ
	Mass	$100.0m$
	Moment of Inertia	$360m\sigma^2$
	Surface Points	162

Table 6.1: Properties of the simulation system for our single sphere simulations.

One important effect seen in the simulations performed in [30] was the dependence of the diffusion constant D of the colloid on the size of the simulation box as a result of the periodic boundary conditions. This effect can be seen in the mean square displacement of the colloids with time, which in the diffusive regime should follow

$$\lim_{t\to\infty} \langle (\mathbf{r}(t) - \mathbf{r}(0))^2 \rangle = 6Dt \tag{6.1}$$

where D is the diffusion coefficient and $< \cdots >$ signify an ensemble average. In figure 6.1, the mean square displacements of a spherical colloid in cubic simulation boxes of varying L are plotted for both our simulations ($L = 15\sigma$, 25σ, and 30σ) and those performed by Zhou and Schmid ($L = 10\sigma$ and 30σ). The mean square displacement curve we obtain for the $L = 30\sigma$ simulations matches closely with the curve obtained

by Zhou and Schmid.

From curves in figure 6.1 it is clear that the diffusion constant increases with

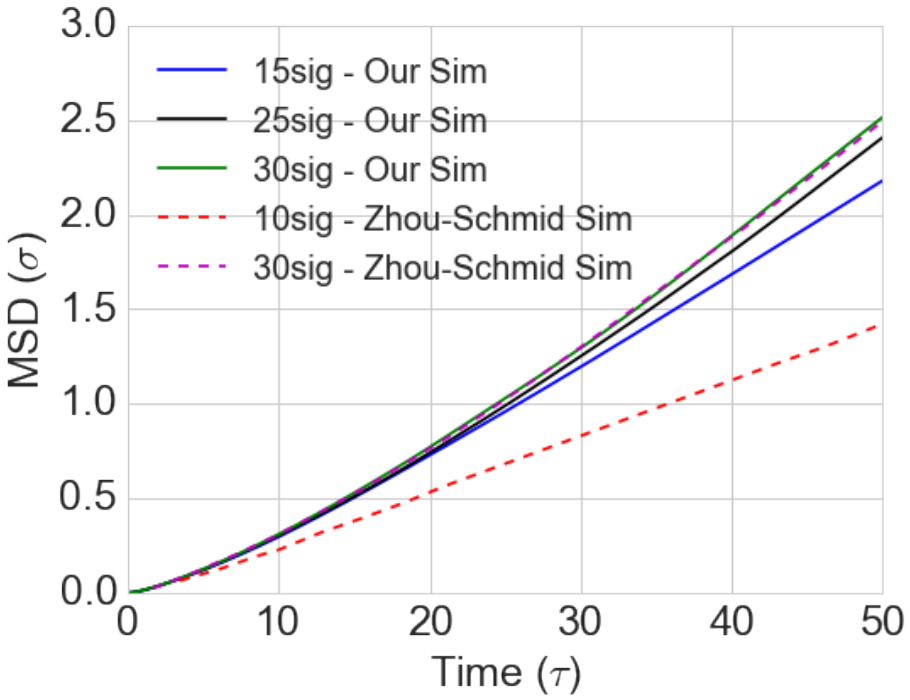

Figure 6.1: The mean squared displacements of spherical colloidal particles over time for various system sizes. Solid curve show our simulation results, while dashed curves are from ref. [30]

increasing box size. This is a hydrodynamic effect caused by the interactions between the colloid and its periodic image. The relation between the diffusion constant and the system size L can be derived analytically in terms of an expansion of powers of

$1/L$ [35]

$$D = \frac{k_B T}{6\pi\mu}\left(\frac{1}{R} - \frac{2.837}{L} + \frac{4.19R^2}{L^3} + \dots\right). \tag{6.2}$$

Using equation 6.1, we calculated the diffusion constant of the colloids at large times by finding the slope of the mean squared displacement at large times. These diffusion constants are plotted in terms of $1/L$ and compared to the curve from equation 6.2 in figure 6.2. Like Zhou and Schmid, we found that the simulations and equation 6.2 agree fairly well.

Other benchmarks used in [30] are the velocity and angular velocity auto corre-

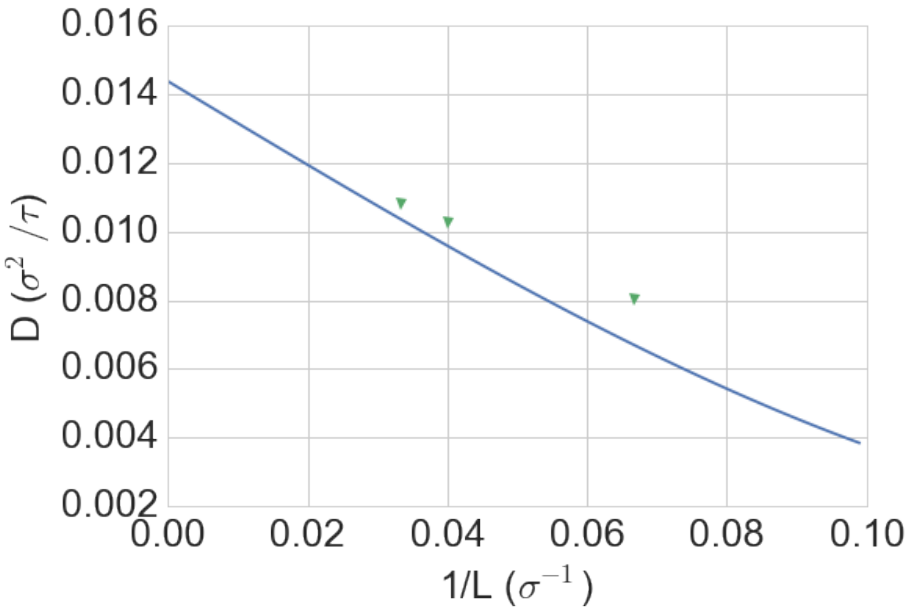

Figure 6.2: Simulation diffusion constant plotted against the reciprocal of the length (points) along with the theoretical prediction of 6.2 (line)

lation functions over time

$$C_v(t) = \frac{\langle \mathbf{v}(0) \cdot \mathbf{v}(t) \rangle}{\langle \mathbf{v} \rangle} \tag{6.3}$$

$$C_\omega(t) = \frac{\langle \omega(0) \cdot \omega(t) \rangle}{\langle \omega \rangle} \tag{6.4}$$

where $\mathbf{v}(t)$ and $\omega(t)$ are the translational and rotational velocities of the colloid at time t. For short time scales, the autocorrelation functions decay exponentially. The Enskog dense-gas kinetic theory predicts that

$$\lim_{t \to 0} C_v(t) = \exp(-\xi^v_{\mathrm{ENS}} t) \tag{6.5}$$

$$\lim_{t \to 0} C_\omega(t) = \exp(-\xi^\omega_{\mathrm{ENS}} t) \tag{6.6}$$

where ξ^v_{ENS} and $\xi^\omega_{\mathrm{ENS}}$, the Enskog friction coefficients, are

$$\xi^v_{\mathrm{ENS}} = \frac{8}{3} \left(\frac{2\pi k_B T m M}{m + M} \right)^{1/2} \rho R^2 \frac{2}{M} \tag{6.7}$$

$$\xi^\omega_{\mathrm{ENS}} = \frac{8}{3} \left(\frac{2\pi k_B T m M}{m + M} \right)^{1/2} \rho R^2 \frac{5}{2M} \tag{6.8}$$

where m is the fluid bead mass, M is the colloid mass, T is the temperature, k_B is the Boltzmann constant and ρ is the solvent density. In our simulation system $\xi^v_{\mathrm{ENS}} = 3.61\tau^{-1}$ and $\xi^\omega_{\mathrm{ENS}} = 4.51\tau^{-1}$. For long time scales, mode-coupling theory

predicts algebraic behavior [30]

$$\lim_{t\to\infty}\langle \mathbf{v}(0)\cdot\mathbf{v}(t)\rangle = \frac{k_bT}{12m\rho(\pi(\mu/\rho+D))^{3/2}}t^{-3/2} \tag{6.9}$$

$$\lim_{t\to\infty}\langle \omega(0)\cdot\omega(t)\rangle = \frac{\pi k_bT}{m\rho(4\pi(\mu/\rho+D))^{5/2}}t^{-5/2} \tag{6.10}$$

where μ is the shear viscosity and D is the diffusion constant. We get

$$\lim_{t\to\infty}\langle \mathbf{v}(0)\cdot\mathbf{v}(t)\rangle = 0.95t^{-3/2} \tag{6.11}$$

$$\lim_{t\to\infty}\langle \omega(0)\cdot\omega(t)\rangle = 3.13t^{-5/2}. \tag{6.12}$$

for our simulations.

The mode coupling and Enskog dense gas kinetic theory predictions are plotted against the translational and angular velocity auto-correlation functions for a spherical colloid in a box of size $L = 25\sigma$ in figure 6.3. We see that the predictions of the Enskog dense gas kinetic theory agree well with the velocity autocorrelation function at small timescales ($t < 10^{-1}\tau$), but does not agree as well in the case of the angular velocity autocorrelation function. This is likely due to the fact that we are approximating a continuous surface using a set of discrete DPD interaction points, which would affect the angular velocity autocorrelation function more than the translational velocity autocorrelation function. We would expect the simulations to more closely match theory as the number of interaction points is increased. In the case of the

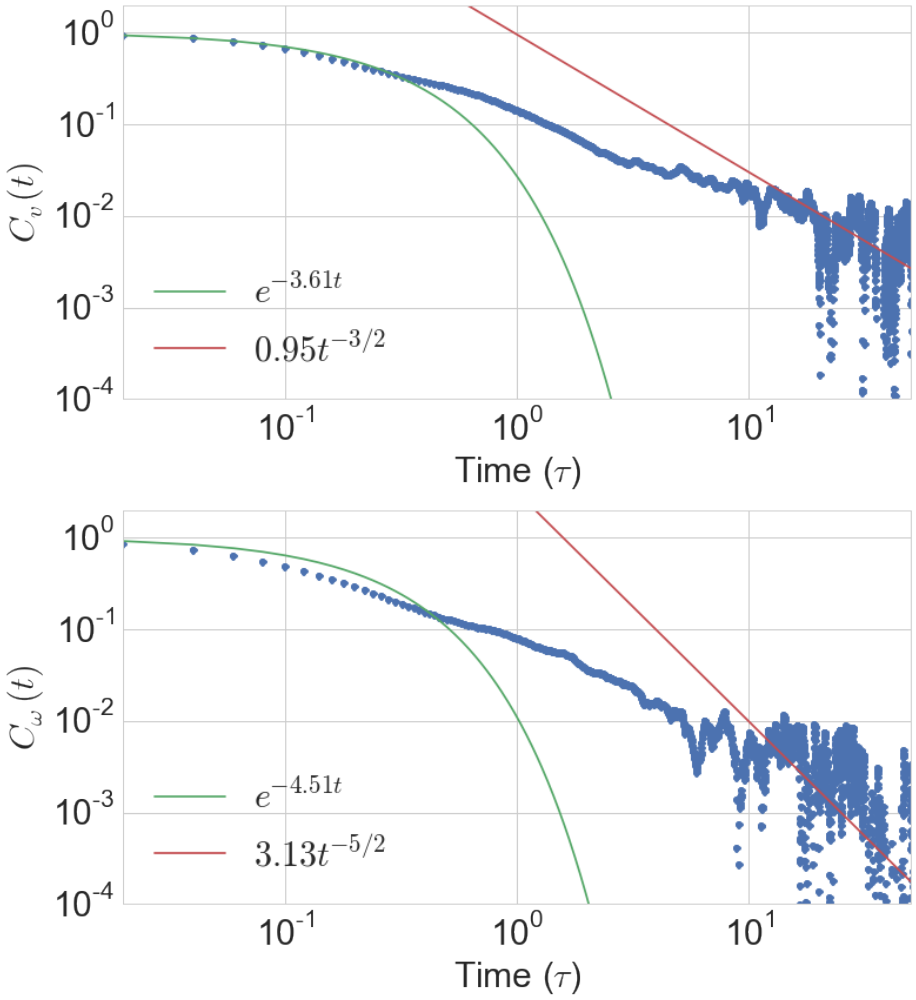

Figure 6.3: The translational (top) and angular (bottom) velocity autocorrelation (blue markers) functions plotted alongside the predictions of the Enskog dense-gas kinetic theory (green line) and the mode-coupling theory (red line).

mode coupling theory predictions, the velocity and angular velocity autocorrelation functions seems to be consistent with the theory at large times ($t > 10\tau$). However, at these timescales both the autocorrelation functions have large fluctuations due to a lack of statistics. Improving the statistics for $t > 10\tau$ would require very long simulations.

6.1.2 Sedimenting Spherical Colloids

In order to simulate the sedimentation of particles, we apply a constant force on every point in the rigid bodies in the z direction. This results in a force on the center of mass with a net torque of zero. When simulating sedimentation, it is important to keep in mind that we are dealing with a finite system, and that the boundary conditions can play a significant effect on the dynamics of the colloids. If we were to naïvely simulate the sedimentation of our colloids with periodic boundary conditions, the friction between the colloids and the fluids would cause the entire system to continuously accelerate downwards. This scenario is unphysical, as we expect a sedimenting particle to eventually reach a terminal velocity and stop accelerating. Thus, it is necessary to counter this effect in some way. The two methods we tested in our simulations are to either introduce walls at the top and bottom of the simulation box or to apply a

force on the fluid such that the net force on the fluid and colloids is zero.

The walls were modeled using a Lennard Jones potential defined as

$$V(r) = \begin{cases} 4\epsilon_{LJ}\left[\left(\frac{\sigma_{LJ}}{r}\right)^{12} - \left(\frac{\sigma_{LJ}}{r}\right)^{6}\right] & r < r_{cut} \\ 0 & r \geq r_{cut} \end{cases} \tag{6.13}$$

where r is the distance to the wall. Typically we use $\sigma_{LJ} = 1.0$, $\epsilon_{LJ} = 1.0$, and $r_{cut} = \sqrt[6]{2}\sigma$. This means that the cutoff occurs at the minimum in the potential, ensuring that the interaction is purely repulsive. These walls are placed at the highest and lowest z coordinates in the simulation box and lie in the xy plane.

To test the walled system, a spherical colloid of radius $R = 3.0\sigma$ was simulated in a cubic box with side lengths $L = 25\sigma$ for various applied forces. In figure 6.4, the mean velocity of the colloids (the terminal velocity) is plotted against the applied force along with the predicted curve for a spherical particle sedimenting under Stokes' drag $u/F = 1/(6\pi\mu a) = 0.0148\tau/m$, using the value for viscosity $\mu = 1.23m/\tau\sigma$ found by Zhou and Schmid [30]. Based on this graph, we can see that the walled system does not reproduce the expected behaviour for a sedimenting spherical particle. These results are consistent with the idea that the velocity of particles in a viscous medium tends to decrease near walls.

For the simulations in which the net force is zero, a constant force $F_{fluid} = \frac{F_{colloid}}{n_{fluid}}$

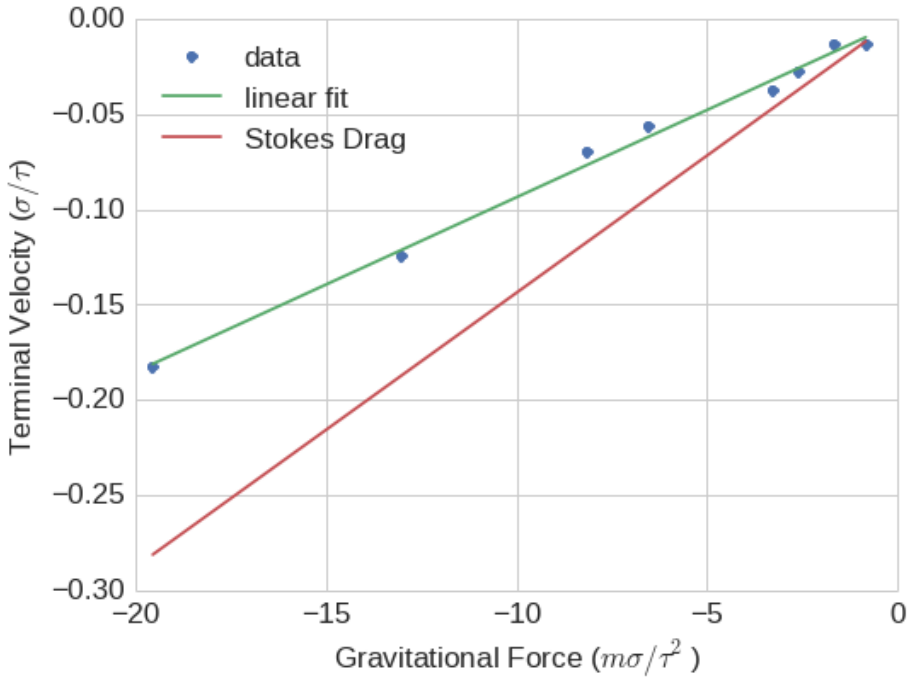

Figure 6.4: Terminal velocity of a spherical particle plotted against the applied force in a system where there are walls at the top and bottom boundaries of the system. A line representing the relation between the quantities for Stokes drag is also plotted.

is applied to the every fluid particles in the positive z direction. This results in a net force of zero on the system which prevents the entire system from accelerating. This is confirmed in figure 6.5 where it can be seen that the velocity of the sedimenting particle remains level over long times. In figure 6.6, the relative terminal velocity of the spherical particles is plotted against the applied force, again alongside the relation predicted by Stokes' Law. In this case we can see that the terminal velocity of the simulated colloids matches closely with the expected values, with the slope of

the linear fit giving a value of $\frac{du}{dF} = 0.0144\tau/m$. Because of this result, most of the simulations we performed used the zero net force method, especially as it has the additional benefit of allowing much longer simulations.

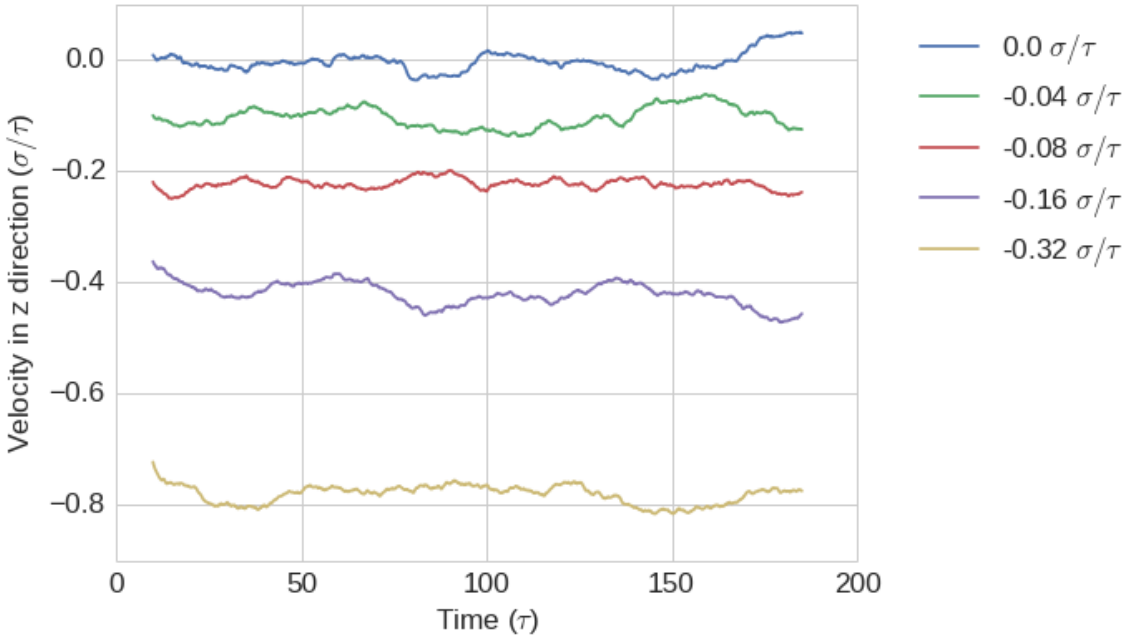

Figure 6.5: Running mean velocity in the z direction over time of the sedimenting colloid for various applied forces.

The Péclet number of the simulated systems can be found using the equation Pe $= \frac{Fa}{k_B T}$. Because this expression for Pe depends only on the applied force, the radius, and the temperature we can easily vary the Péclet number by changing these variables. For $F = 6.52m\sigma/\tau^2$ on a spherical particle of radius $a = 3.0\sigma$ and in a fluid

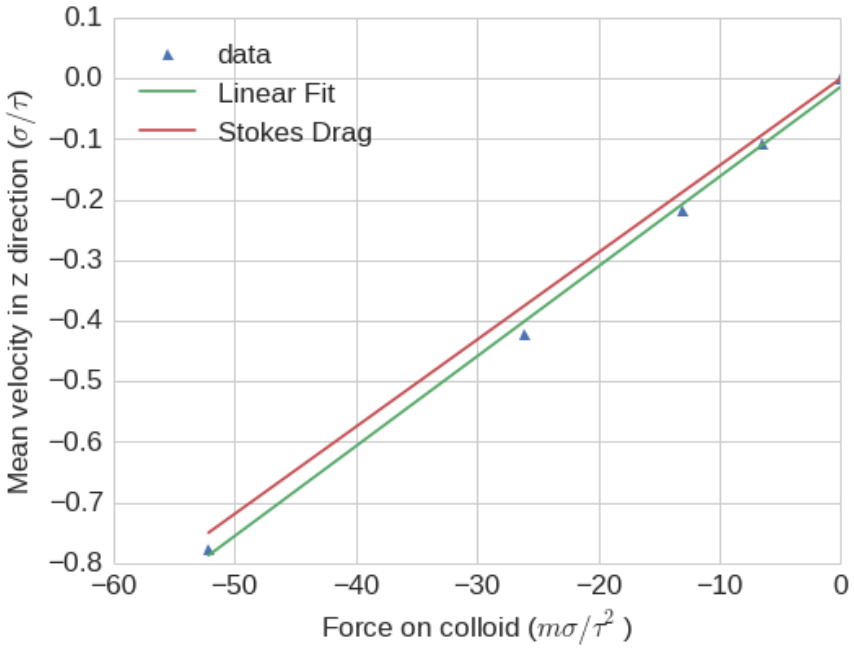

Figure 6.6: Terminal velocity of a spherical particle plotted against the applied force in a system in which a force is applied to the particles to make the net force on the system zero. A line representing the relation between the quantities for Stokes drag is also plotted.

with temperature $k_BT = 1.0\epsilon$, the Péclet number is Pe = 19.56.

6.1.3 Sedimentation of 1D arrays of Spherical colloids

One of the simplest systems in which we can observe the Crowley instability is a line of 5 spherical particles. In such a system, we expect that due to the line of center forces between the particles, the closer to the center a particle is, the faster it sediments.

Figure 6.7: Initial condition of the 5 sphere sedimentation simulations.

Simulation Box	L_x	L_y	L_z
	60σ	30σ	30σ
Sphere Properties	Radius		2.0σ
	Mass		$100.0m$
	Moment of Inertia		$160m\sigma^2$
	Surface Points		162

Table 6.2: Properties of the simulation system for a line of 5 spherical particles.

A line of 5 spheres along the x-axis with initial conditions as shown in figure 6.7 was simulated with sedimentation force $F = 6.52m\sigma/\tau^2$ and an inter-particle distance of 8σ. The Péclet number was varied by changing the temperature of the system. As we are interested in relatively long-time effects, we used a zero net force system. The properties of the spherical colloid particles are specified in table 6.2.

In figures 6.8 and 6.9 the trajectories of the spherical particles are plotted over time for $T = 0.05\epsilon$ (Pe = 262), 0.1ϵ (Pe = 131), 0.25ϵ (Pe = 52), and 0.5ϵ (Pe = 26), where the temperature T is expressed in model energy units and $k_b = 1$. We can clearly observe the clustering of the central particles in the $T = 0.05\epsilon$ and $T = 0.1\epsilon$ systems, while in the $T = 0.25\epsilon$ and $T = 0.5\epsilon$ this effect is not observed, though the proximity

of the particles to one another does seem to have an effect on the sedimentation speeds of the particles, as the spheres seem to move under the influence of line of center forces. This confirms that, as the Péclet numbers of the systems suggest, the $T = 0.05\epsilon$ and $T = 0.1\epsilon$ systems are largely non-Brownian, and Brownian motion has a significant effect on the motion of the particles when $T = 0.25\epsilon$ and $T = 0.5\epsilon$.

One problem that arises in the systems due to the periodic boundary conditions used in the simulations is that the particles "see" themselves above and below during sedimentation. Thus, the five particles end up moving in their own wakes and start to accelerate rapidly due to the lower drag in the wake of their periodic images. At long times in the $T = 0.05\epsilon$ and $T = 0.1\epsilon$ simulations, this leads to the particles grouping up and forming columns which fall at very high speeds. This is illustrated in figure 6.10, where there is a clear shift in the acceleration on the particles causing a large fluctuations in the speed of the particles after around 600τ. Thus, we can only use results from earlier portions of the simulations. Nonetheless, the clustering effects we observe occur well before the periodic boundary conditions begin to cause such behavior in the particles.

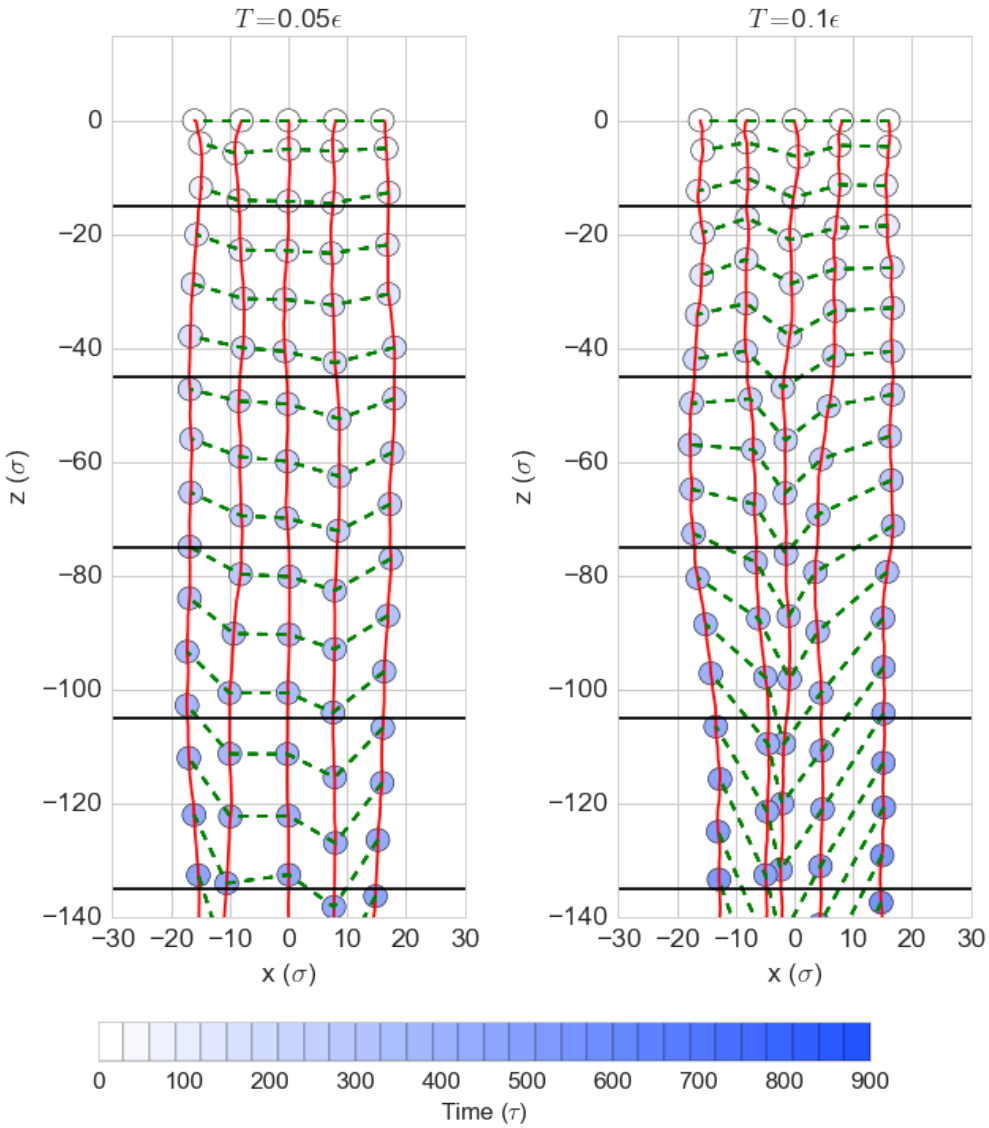

Figure 6.8: x and z positions of the five spheres over time for temperatures $T = 0.05\epsilon$ (Pe = 262) on the left and $T = 0.1\epsilon$ (Pe = 131) on the right. Red lines indicate the centroid of the spheres and circles are used to represent the spheres at intervals of 30τ. The solid black lines indicate the periodic boundaries of the system and the green dashed lines connect spheres of the same time step.

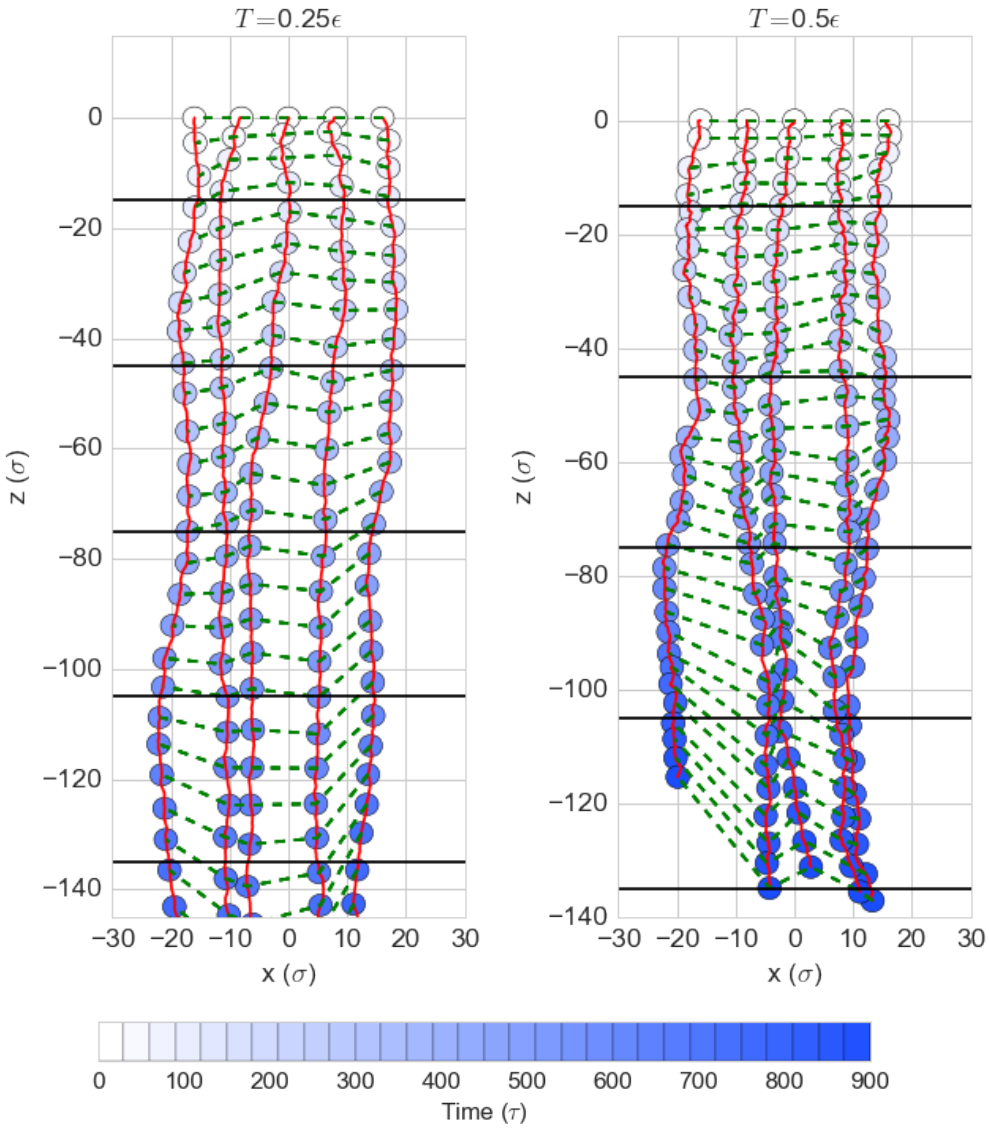

Figure 6.9: x and z positions of the five spheres over time for temperatures $T = 0.25\epsilon$ (Pe = 52) on the left and $T = 0.5\epsilon$ (Pe = 26) on the right. Red lines trace the centroid of the spheres and circles are used to represent the spheres at intervals of 30τ. The solid black lines indicate the periodic boundaries in the system and the green dashed lines connect spheres of the same time step.

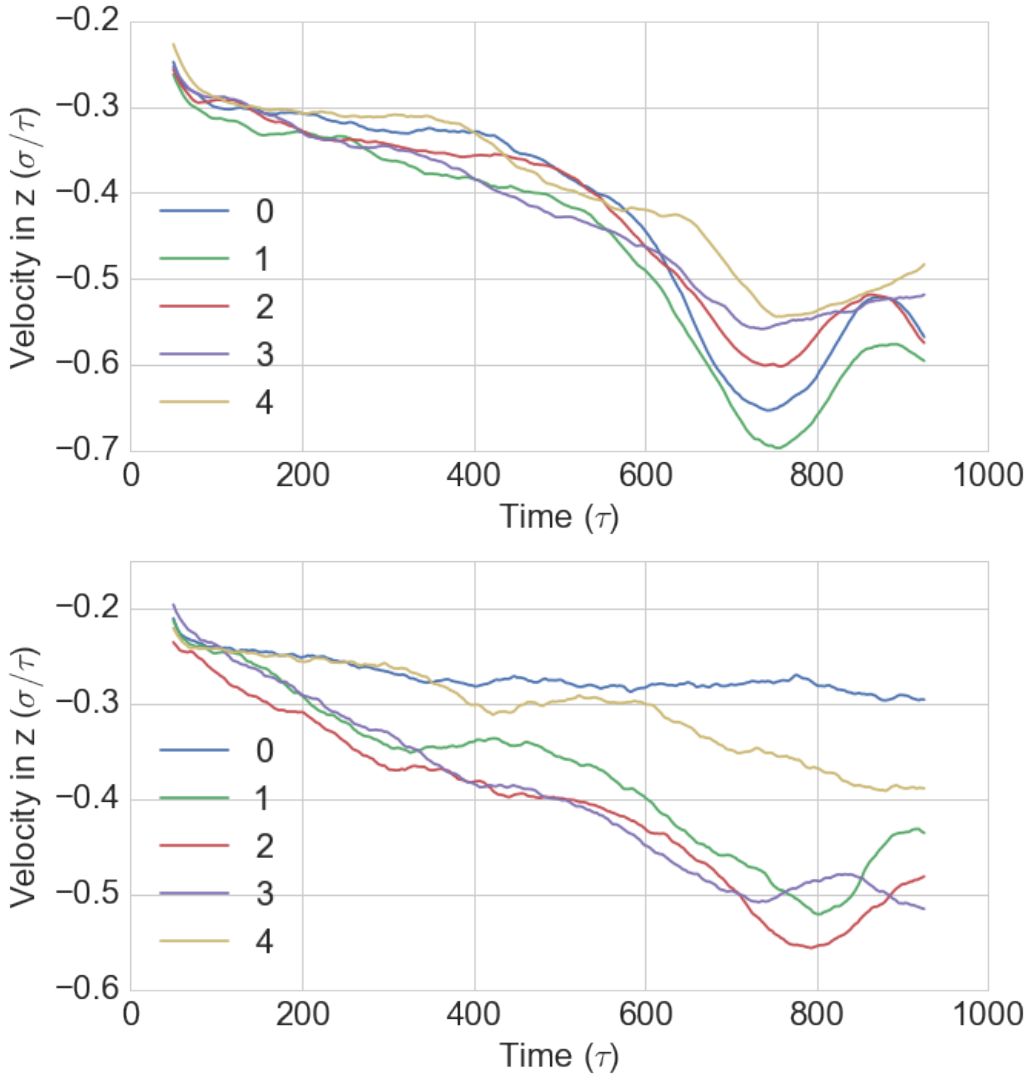

Figure 6.10: Velocity in the z direction over time for 5 sphere simulations with $T = 0.05\epsilon$ (Pe = 262) on the top and $T = 0.1\epsilon$ (Pe = 131) on the bottom. The spheres are numbered from left to right as they appear in figure 6.8.

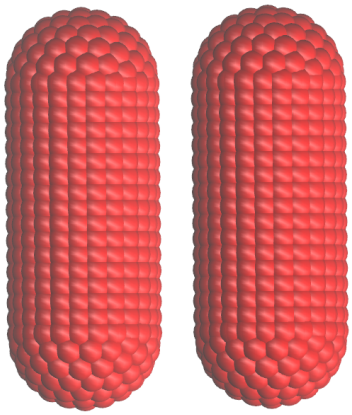

Figure 6.11: Initial condition for simulations of two initally vertical rods sedimenting.

6.2 Simulations of Rod Colloids

6.2.1 Sedimentation of Two Adjacent Rods

One hydrodynamic effect we are interested in reproducing in our simulations is the periodic rotations of anisotropic particles during sedimentation reported by Jung et. al. [11]. In order to do so, we simulated two sedimenting rod particles which are initially oriented vertically near one other as shown in figure 6.11 with an initial inter-particle distance of 6.0σ. The properties of the rod particles can be found in table 6.3. These simulations were carried out using an applied force of $F = 13.04$ and the Péclet number of the system was varied by changing the temperature of the

fluid. Based on the experiments and hydrodynamic theory studied in [11], we expect the rod particles to rotate in periodically as illustrated in figure 2.2.

Simulation Box	L_x	L_y	L_z
	60σ	60σ	60σ
Rod Properties	Radius (R)		2.0σ
	Length (L)		12.0σ
	Hydrodynamic Radius (R_H)		3.43σ
	Mass (M)		$500.0m$
	I_{xx}	I_{yy}	I_{zz}
	$\approx 5257m\sigma^2$	$\approx 5290m\sigma^2$	$\approx 970m\sigma^2$
	Surface Points		628

Table 6.3: Properties of the rod particles.

In figure 6.12, the orientations and positions two rod particles initially oriented vertically are plotted over time for $T = 0.1\epsilon$ (Pe $\sim$ 445), 0.25ϵ (Pe $\sim$ 180), and 0.5ϵ (Pe $\sim$ 90). We estimate the Péclet number Pe using the equation Pe= $\frac{FR_H}{k_BT}$, where $k_B = 1.0$ and use equation 2.4 in section 2.1.1 to calculate the hydrodynamic radius $R_H = 3.43$. In the plot for the $T = 0.1\epsilon$ simulation, we observe that the rods move as expected for part of the rotation, spreading outwards and rotating into a horizontal position while largely remaining in the yz-plane. However, while the rods do begin to tip inwards, they do not undergo the next stage of the rotation and continue to sediment in a horizontal position. This may be due to the periodic boundary conditions, which produce periodic images of the rods on either side of the

simulation box. As a result, the tendency of the rods to rotate and move closer to one another is opposed by similar influences due to the periodic images of the rods, resulting in the rods remaining horizontal as they sediment. However, this could also be a result of the initial separation distance between the rods, as it was found in [9] that above some cutoff separation a pair of sedimenting disks (1mm wide, 1.2cm in diameter) will rotate into a horizontal orientation and will not rotate back into a vertical orientation. Experimentally this cutoff separation was found to be around 5 times the disks' widths, which is larger than the separation used in our simulation which was 1.5 times the rods' diameters. However, this cutoff distance may vary between differently shaped particles, so we cannot rule out this explanation for the observed behavior of the rods. Performing simulations using larger simulation boxes and using different separation distances will help address this question. Taking the time taken for the rods to rotate from a vertical position to a horizontal position to be half the period of the rotational motion, we find the period is $\approx 650\tau$.

In the $T = 0.25\epsilon$ and $T = 0.5\epsilon$ simulations, the rods also rotate into a roughly horizontal orientation and continue to sediment horizontally thereafter. As can be seen from figure 6.13, the rods in the $T = 0.25\epsilon$ simulation tend to reach a horizontal orientation at around the same time ($\approx 300\tau$) as those in the $T = 0.1\epsilon$ simulation. The $T = 0.5\epsilon$ simulations rods also tend to rotate to the horizontal, but experience

much more variation in orientation and rotation speed, which results in the rods rotating out of the xz plane in which they are initially positioned. The fact that the rotational motion of the rods in our simulations is similar to the motion of anisotropic particles in the hydrodynamic domain fits with what we'd expect given their Péclet numbers, as all the simulations have Pe >90. The greater influence of random motion on the rotation of the rods in the $T = 0.5\epsilon$ simulations also fits with the behavior we'd expect given the system's Pe $\sim$ 90, which is outside of the non-Brownian domain (Pe $>$ 100).

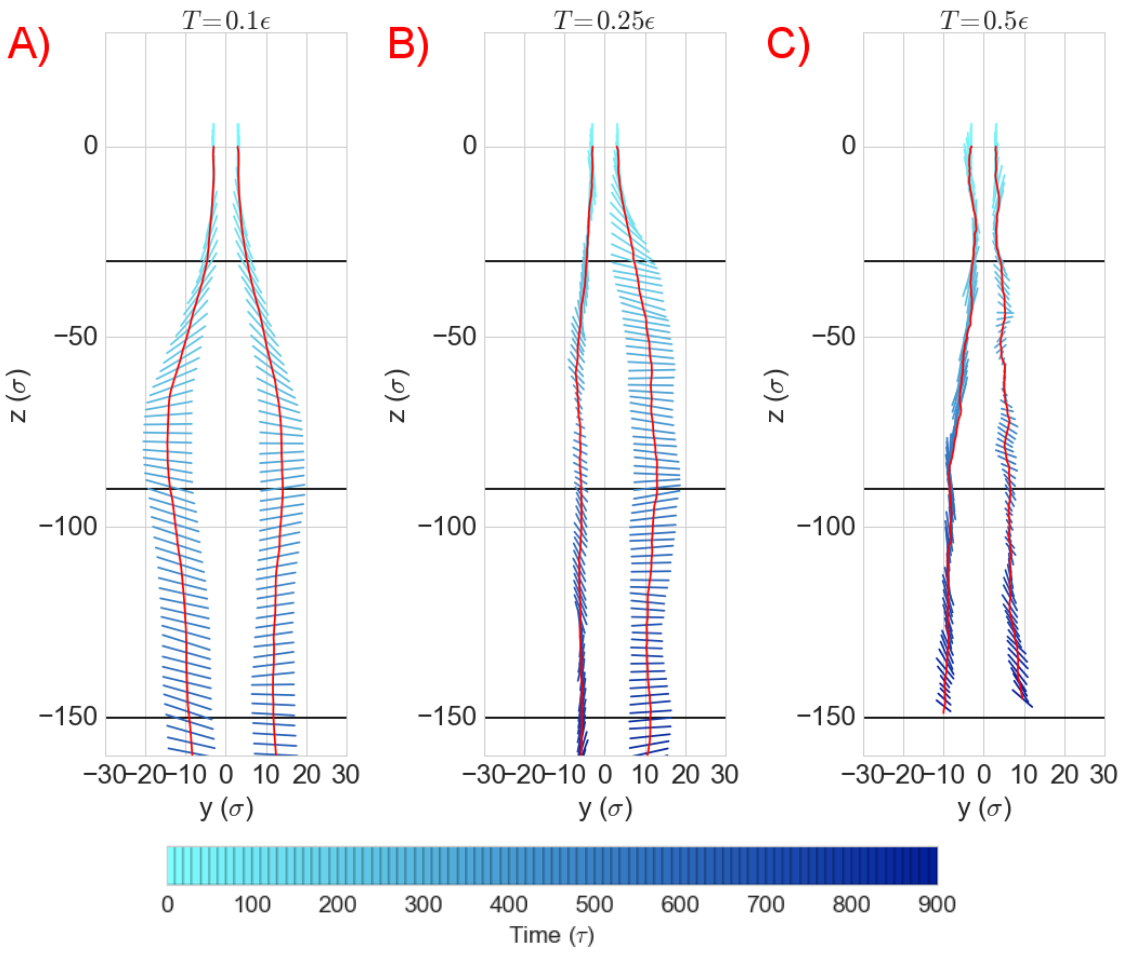

Figure 6.12: y and z positions of particles in a two rod simulation over time for temperatures $T = 0.1\epsilon$ (A - Pe $\sim$ 445), $T = 0.25\epsilon$ (B - Pe $\sim$ 180) and $T = 0.5\epsilon$ (C - Pe $\sim$ 90). Red lines trace the centroid of the rods and blue lines are used to represent the primary axis of the rods at intervals of 10τ. The solid black lines indicate the periodic boundaries in the system.

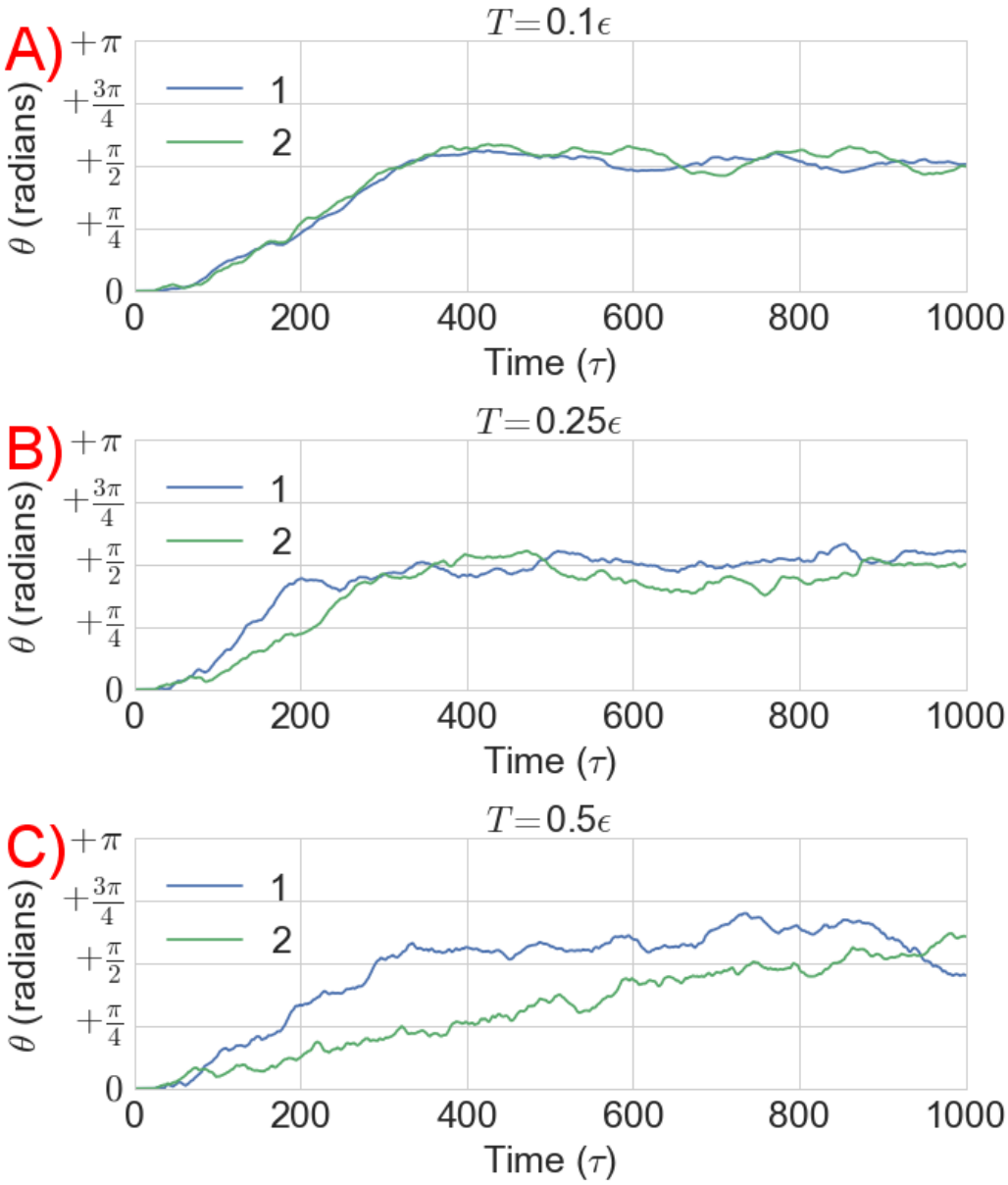

Figure 6.13: Running mean of the polar angle θ of the rods over time of a two rod simulation wherein the rods begin vertical. The simulations were run at temperatures $T = 0.1\epsilon$ (A - Pe $\sim$ 445), $T = 0.25\epsilon$ (B - Pe $\sim$ 180) and $T = 0.5\epsilon$ (C - Pe $\sim$ 90).

6.2.2 Sedimentation of a 1D array of Rod Particles

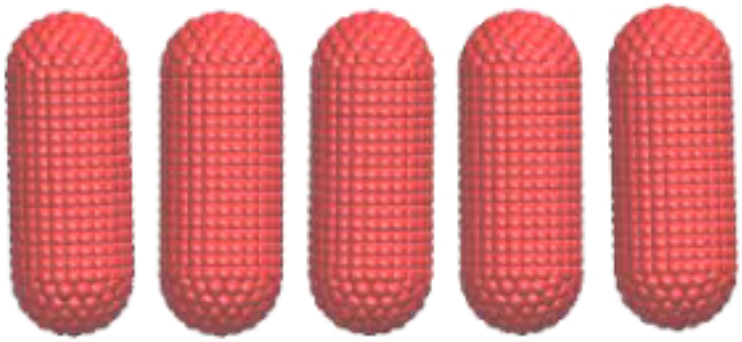

Figure 6.14: Initial condition for simulations of five initally vertical rods sedimenting

Another system we are interested in is one that is similar to one that was tested experimentally by Rahul Chajwa [9]: the sedimentation of a line of 5 disks (Pe $\sim 10^{15}$). In [9], it was found that when the disks were initially oriented vertically, they tended to spread outwards. We represented this system with a line of 5 rods oriented vertically and aligned in the yz plane as shown in figure 6.14. The properties of the rod particles are the same as in table 6.3, with the rods having an inter-particle distance of 6.0σ, an applied force of $F = 6.52m\sigma/\tau^2$, and a system temperature which is varied to change the Péclet number.

Figure 6.15 plots the positions and orientations of the rods as they sediment for temperatures of $T = 0.1\epsilon$ (Pe $\sim$ 230), 0.15ϵ (Pe $\sim$ 150), and 0.2ϵ (Pe $\sim$ 110). In all three simulations, we can see that the rods initially swing outwards, just as

shown experimentally in [9], with the non-central particles rotating into horizontal orientations. In the $T = 0.15\epsilon$ and 0.2ϵ simulations, the random motion causes the rods to rotate out of the yz plane and the rods continue to sediment in a horizontal orientation.

In the $T = 0.1\epsilon$ simulation, the central rod remains vertical and quickly falls ahead the other rods. Soon after, the center-left and center-right rods tip inwards from their horizontal positions and rotate into a vertical orientation, picking up speed as they enter the wake of the central particle. These phenomena can be seen in the velocity of the rods in the z direction is when it is plotted over time (fig. 6.16). One can see that the speed of the outer particles usually undergo an initial increase, then decrease as the rods rotate into a horizontal position. This does not occur for the central particle (trajectory 3) in the $T = 0.1\epsilon$ simulation, and at $t \approx 500$ we observe that the speeds of the center-left (2) and center-right trajectories (4) increase rapidly corresponding to the rotation of the particles into a vertical position and the reduced drag force in the wake of the central rod. The motion of the rods in the $T = 0.1\epsilon$ system seems to be the result of a combination of the Crowley instability, as the rods cluster together as they sediment, and the periodic rotational motion described in [11], as the center-left and center-right rods undergo a full rotation in a manner similar to the two rod systems they described.

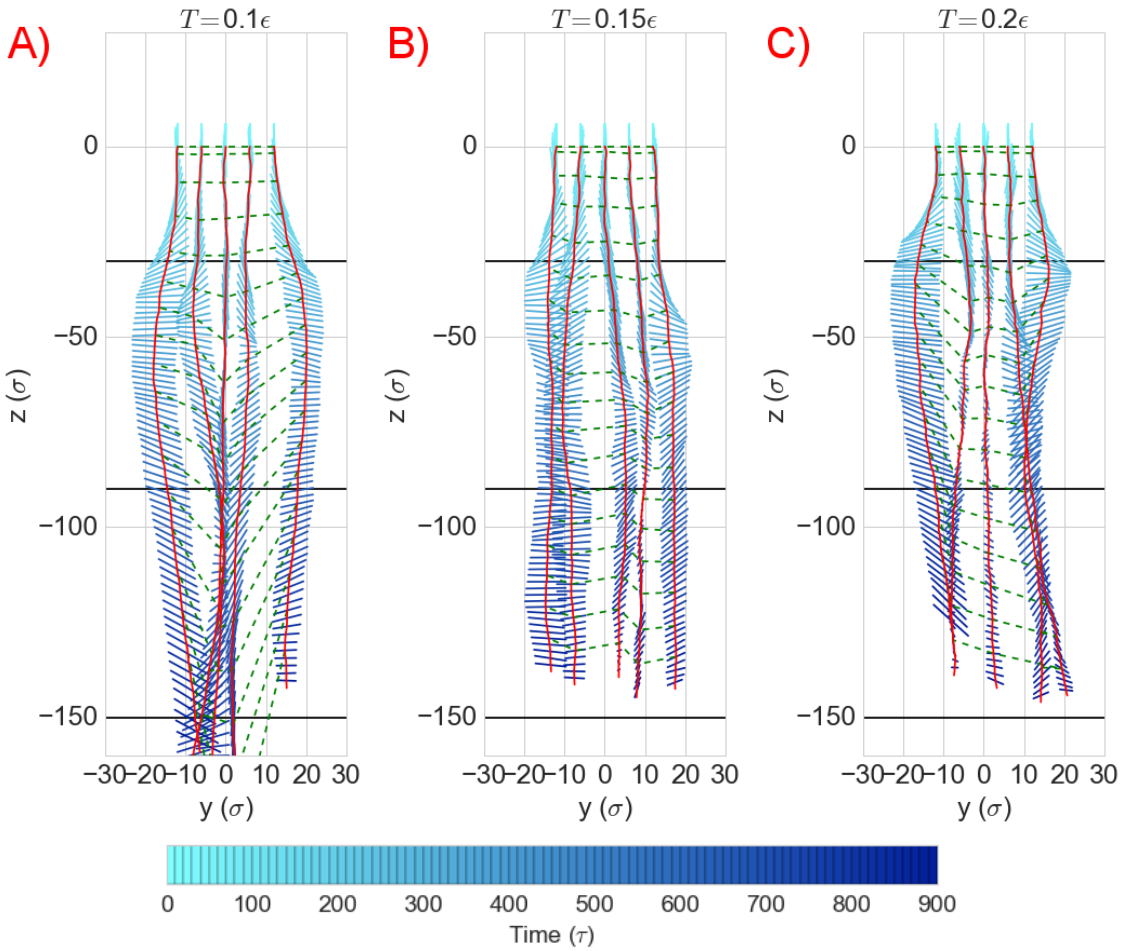

Figure 6.15: y and z positions of particles in a five rod simulation over time for temperatures $T = 0.1\epsilon$ (A - Pe $\sim$ 230), $T = 0.15\epsilon$ (B - Pe $\sim$ 150) and $T = 0.2\epsilon$ (C - Pe $\sim$ 110). Red lines trace the centroid of the rods, blue lines are used to represent the primary axes of the rods at intervals of 10τ, and green dashed lines connect points in the same timestep at intervals of 50τ. The solid black lines indicate the periodic boundaries in the system.

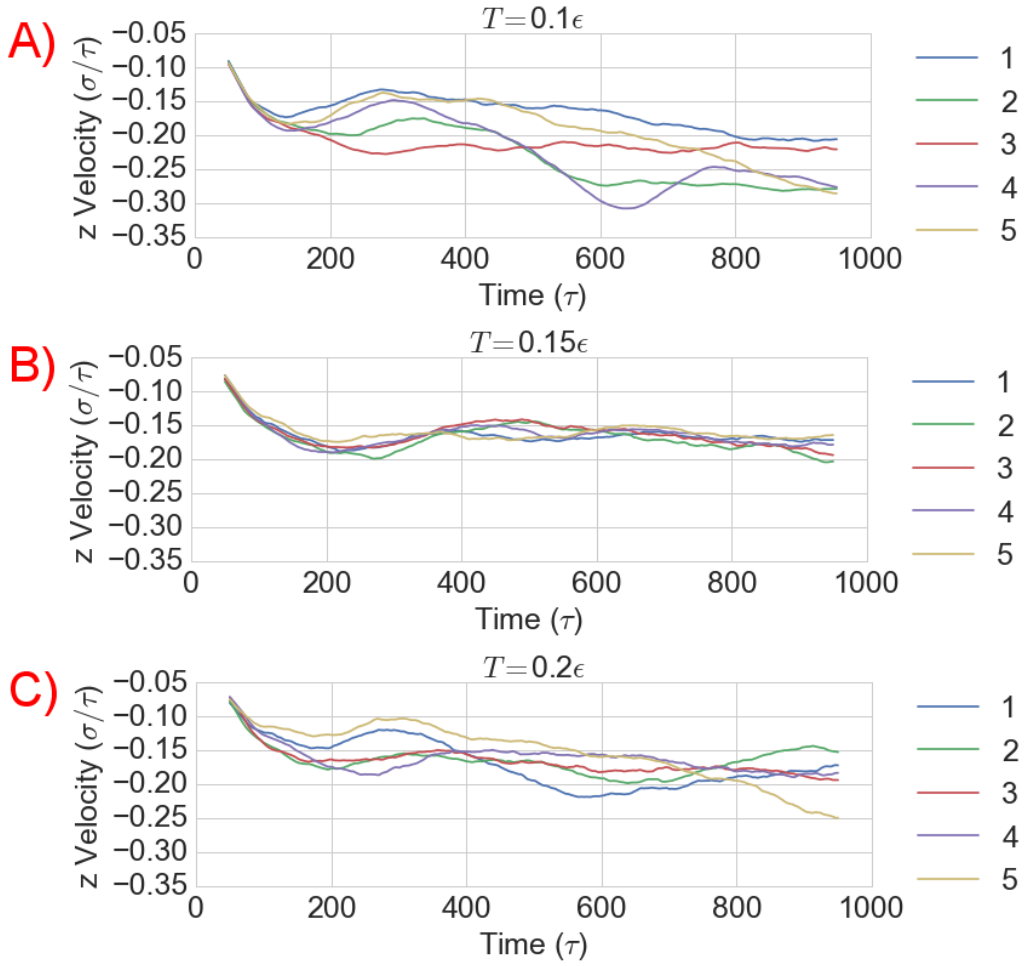

Figure 6.16: Running mean of the velocity in the z direction of the rods over time of a five rod simulation wherein the rods begin vertical. The simulations were run at temperatures $T = 0.1\epsilon$ (A - Pe $\sim$ 230), $T = 0.15\epsilon$ (B - Pe $\sim$ 150) and $T = 0.2\epsilon$ (C - Pe $\sim$ 110). The trajectories are numbered from left to right in the yz plane.

6.2.3 Sedimentation of a 2D array of Rod Particles

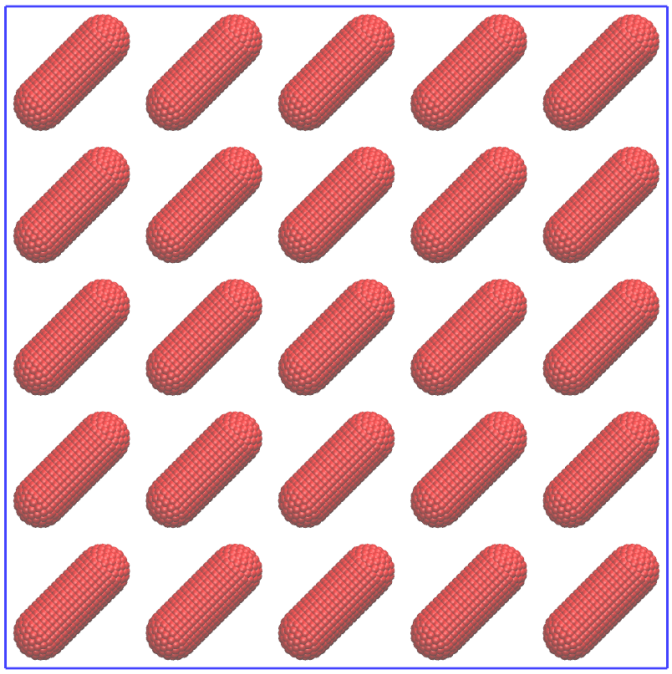

Figure 6.17: Initial condition for simulations of a 5 by 5 array of horizontal rods sedimenting. The blue box indicates the periodic boundaries of the simulation.

The final set of simulations we ran were simulations of the sedimentation of 5 by 5 grids of horizontal rods. The rods have properties as shown in table 6.3. The rods were oriented 45 degrees from the xz plane and placed with an initial inter-particle spacing of 12σ and were simulated with a system temperature of $k_BT = 1.0\epsilon$. Due to the proximity and number of rods, the effect of a given rod on itself due to the periodic boundary conditions is minimal, though does manifest at long enough timescales.

The rods are then simulated as they sediment with applied forces of $F = -1.63m\sigma/\tau^2$ (Pe ~ 5.6), $-3.26m\sigma/\tau^2$ (Pe ~ 11), $-6.52m\sigma/\tau^2$ (Pe ~ 22), and $-26.08m\sigma/\tau^2$ (Pe ~ 90).

Figures 6.18 and 6.19 show the sedimenting rods at various time intervals. Note that because of the differing sedimentation speeds, the time intervals are not the same across the images. For the simulations with higher applied forces ($F = -6.52m\sigma/\tau^2$ and $-26.08m\sigma/\tau^2$), we observe that groups of rods begin to break away from the rest after some time. This effect can be seen in figure 6.19 at times $t = 250\tau$ and $t = 125\tau$ for the $F = -6.52m\sigma/\tau^2$ and $-26.08m\sigma/\tau^2$ systems respectively. At these times we observe that a handful of rods have sedimented faster than the rest. This results in the rods being far more spread out in the z direction at these times than at similar times in the lower Péclet number systems, where we would expect equal amounts of dispersion due to Brownian motion (all systems have the same temperature), but less due to hydrodynamic effects. For simulations with lower applied forces ($F = -1.63m\sigma/\tau^2$ and $-3.26m\sigma/\tau^2$), the rods become disordered and slowly disperse, and we do not observe the rods forming clusters that fall away from the rest of the rods. This does not necessarily mean that clustering does not occur, as the distances sedimented in the lower force simulations is less than those in the higher force simulations. Nonetheless, when we compare the simulations after the rods have

sedimented similar distances, there is a visible difference in degree to which the rods have clustered. The observed changes in the clustering of the rods as the Péclet number decreases match the expected contributions of hydrodynamics and Brownian motion to the dynamics of the particles. This is because the clustering, if it is indeed caused by the Crowley instability, is a hydrodynamic effect and would disappear as Brownian motion became more important. At very long times ($t > 200\tau$ for Pe $\sim$ 90 and $t > 400\tau$ for Pe $\sim$ 22 simulations), the rods begin to fall into the wakes of their periodic images and accelerate rapidly, just like in the previous multiple colloid simulations.

By plotting the standard deviations of the areas of the Voronoi polygons (figure 6.20) we can get an idea of how clustered the rods become over time. Due to the random motion of the rods, we expect the standard deviation to increase from the initial value even for a completely Brownian system. However, if there is any clustering occurring due to hydrodynamics, the value will increase much faster than in a simple Brownian system. This behavior can be seem in standard deviations of the Voronoi areas of the Pe $\sim$ 22 and Pe $\sim$ 90 systems, where we would expect stronger hydrodynamic effects and therefore more clustering due to the Crowley instability.

While we don't observe any noticeable clustering in the lower Pe systems, this does not mean that the Crowley instability does not manifest at these Péclet num-

bers. Since the rods sediment slower at lower Pe, even though each simulation was run for the same amount of time steps, the rods in low Pe systems don't sediment as far as those in systems with higher Pe. Thus, it may be the case that the Crowley instability still occurs for relatively low Pe ($\sim 1 - 10$) and we simply have not simulated the sedimentation of the rods for long enough for the clustering to manifest. Furthermore, as noted in the experimental results section, the clustering effects we observe in our experiments (Pe $\sim$ 2 - 5) have sizes greater than 5 rod lengths, which is larger than our simulation box size. Performing larger and longer simulations would be needed to get a better understanding of the behavior of the Crowley instability at Pe values similar to those in our experiments.

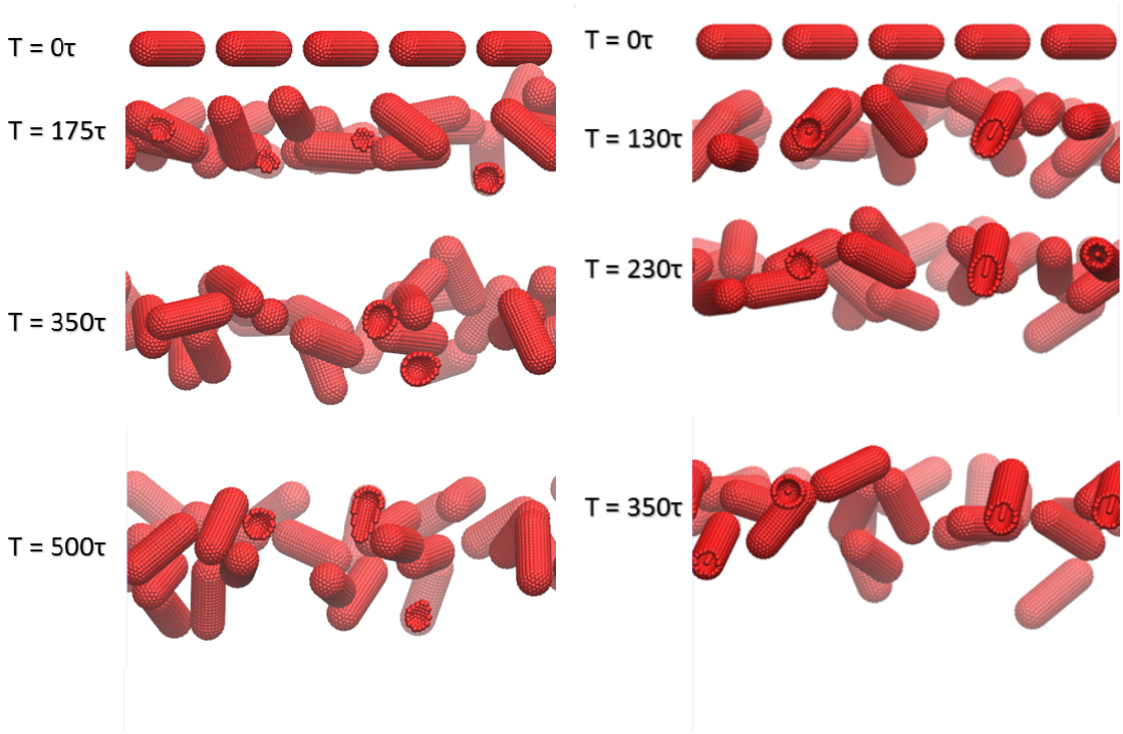

Figure 6.18: Rendering of the 25 rod sedimentation simulations in the xz plane at different time intervals for Pe $\sim$ 5.6 (left) and Pe $\sim$ 11 (right).

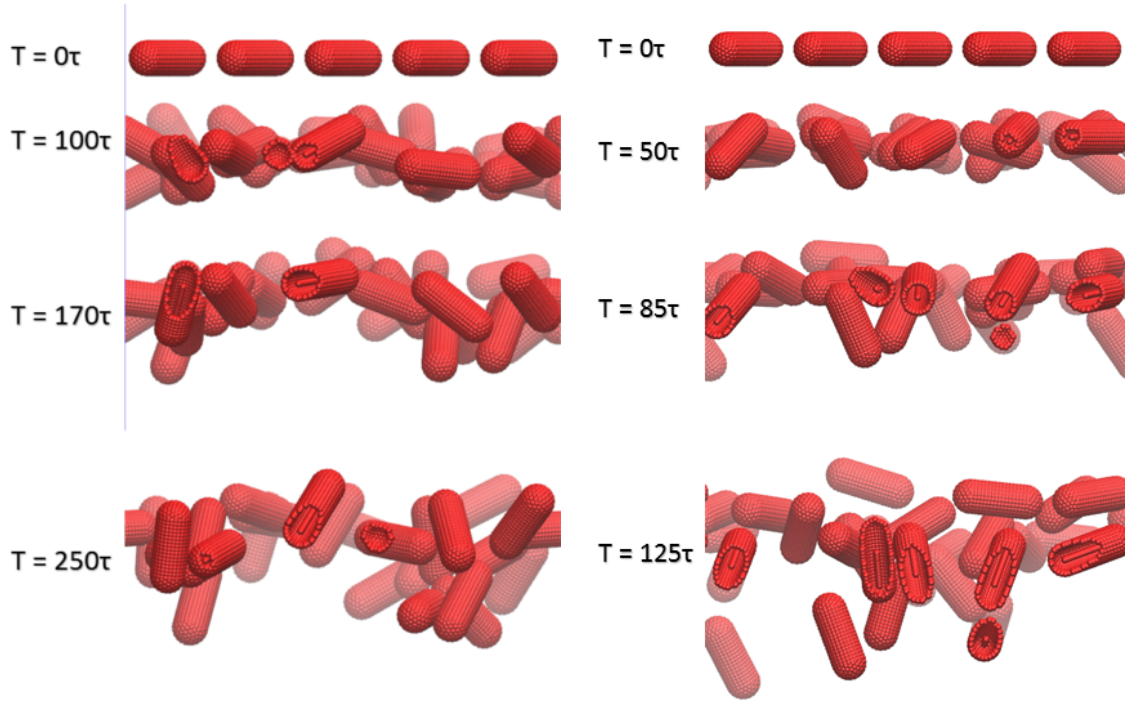

Figure 6.19: Rendering of the 25 rod sedimentation simulations in the xz plane at different time intervals for Pe $\sim$ 22 (left) and Pe $\sim$ 90 (right).

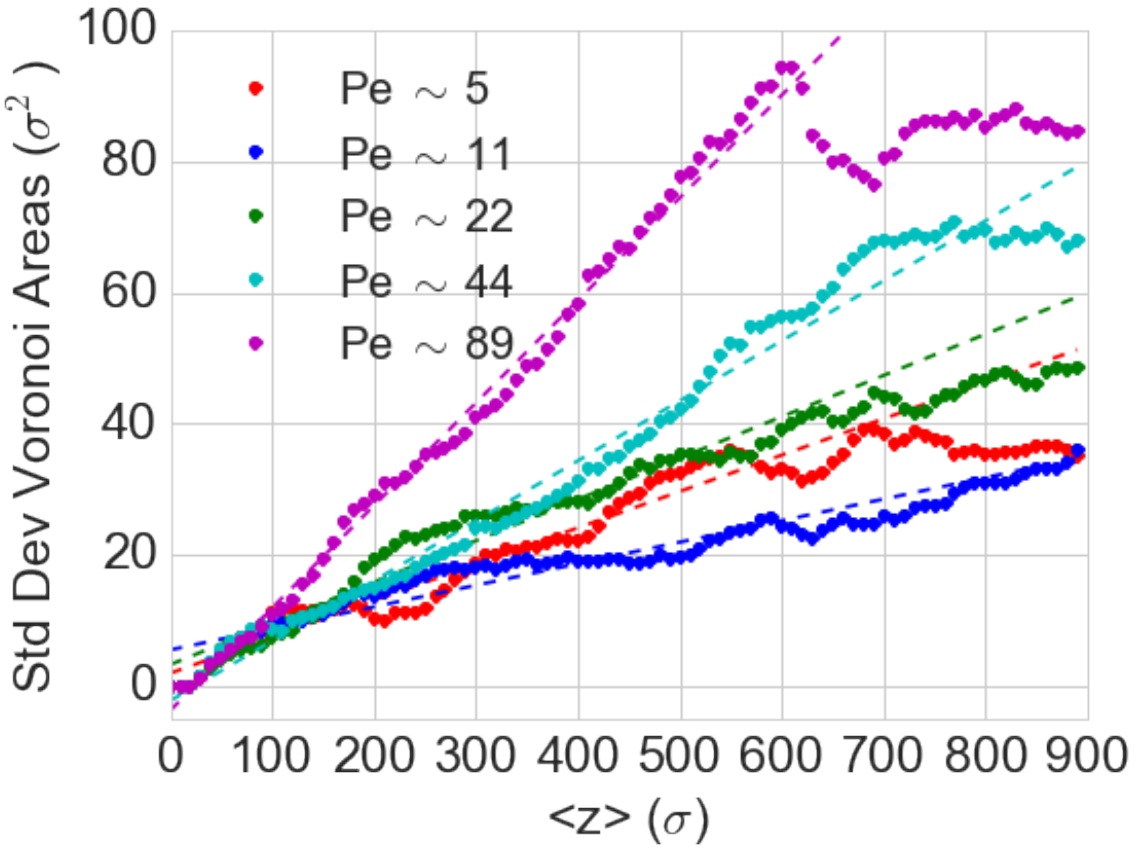

Figure 6.20: Standard Deviation of the Voronoi Areas of colloids in a 25 rod simulation over time for various Péclet numbers.

Chapter 7

Conclusion

Summary of Results

Through the use of experiments and simulations we were able to make some progress towards gaining a better understanding of the dynamics of rod-like colloids during sedimentation, though many questions still remain. In the experimental images, it was observed that the instabilities form in the rod-like colloids as they sediment, which subsequently develop into clusters after some time and into columns of sedimenting colloids at even later times. This behavior is very reminiscent of the Crowley instability that is seen in the sedimentation of particles in the hydrodynamic domain. The fact that these instabilities were observed in spite of the fact that the Brownian motion has a significant contribution to the dynamics of the colloids (Pe $\sim$ 1 to 5) suggests that the Crowley instability in many body systems is a fairly robust hydrodynamic effect.

By analyzing the images using the distributions of the areas of the polygons gen-

erated by a Voronoi diagram A_{vor}, it is possible to get quantitative measures of the clustering of the colloids. Using the distributions of A_{vor} for colloids above and below the median z coordinate in images in which we observe the clustering of colloids, we confirmed that the clustering of colloids tends to first occur at the leading edge of the sedimenting particles. By using the skewness of A_{vor} as a measure of the clustering and calculating it for the experimental image sets in which the formation of instabilities and clusters can be clearly observed, it is possible to track development of clusters in the sedimenting colloids over time. It is found that the skewness of A_{vor} increases steadily as the colloids sediment and cluster together, though the rate at which the skewness increases varies among the image sets we analyzed.

Both quantitative examination and the histograms of A_{vor} for experiments of vertical and freely rotating colloids at similar times seem to suggest that the rod-like colloids tend to cluster closer together when they are held in a vertical position. However, since the two systems have different colloid concentrations and have similar $A_{vor}/< A_{vor} >$ histograms, we cannot conclusively say that this is actually the case.

Using Dissipative Particle Dynamics (DPD), the dynamics of sedimenting colloids in a number of systems were simulated. In the case of a single spherical colloid diffusing in a DPD fluid, the mean squared displacement, velocity autocorrelation functions, and angular velocity autocorrelation functions of our simulation colloids

matched closely with the behavior observed by Zhou and Schmid [30]. Furthermore, it was found that the diffusion constants of the spherical colloids agreed well with the theory of Hasimoto for spheres in a box with periodic boundaries.

In order to simulate the sedimentation of colloids, two different methods for correcting the long-time behavior of the systems were tested. When a wall is introduced at the top and bottom of the simulation box, the terminal velocity of the colloid during sedimentation becomes considerably smaller than the value predicted by Stokes' drag. On the other hand, when a constant force is applied to the fluid so that the net force on the system is zero, the relation of the terminal velocity of a single sedimenting sphere to the applied force agrees well with Stokes' drag. However, when simulating more than one colloid, hydrodynamic interactions such as the mutual drag reduction of particles in close proximity to one another cause the zero net force method to fail at long times, resulting in a constantly accelerating system.

To test the Crowley instability and the periodic rotational motion of pairs of anisotropic particles in the DPD simulations, a series of simulations of multiple colloid systems were performed. These are:

- 5 spherical colloids in a row
- 2 adjacent rod colloids
- 5 rod colloids in a rod
- 25 rod colloids in a 2D lattice.

Crowley instability-like behavior is observed in the 5 sphere, 5 rod, and 25 rod simulations. In the case of 5 spheres, at high enough Péclet numbers (Pe >100) the central colloids sediment faster than those on the sides and the colloids form v-shaped clusters. At lower Pe (Pe $\sim$ 26 and 52), the spheres seem to move under the influence of line of center forces, but they do not form v-shaped clusters.

For 2 adjacent rod colloids, the rods rotate from their vertical to horizontal in a manner similar to the periodic rotation described in [11], however they do not rotate back into a vertical orientation as expected. In [9] it was found that if two vertical disks are placed at an initial separation above a certain threshold value and allowed to sediment, the disks will rotate into a horizontal position, but will not rotate back into a vertical position. This may be the reason why the rods in our simulations do not complete full periodic rotational motions. Alternatively, it could be a result of the periodic boundary conditions of the simulation box.

In simulations of 5 rod colloids in a row, it is observed that the rods initially spread outwards, just as shown experimentally in [9] for particles in the hydrodynamic domain. When the system was simulated at high Pe ($\sim$ 230), the rods spread out initially and as the sedimentation continues the rods rotate into a vertical position and cluster together. This behavior seems to be the result of a combination of the effects that cause the Crowley instability and those that cause the periodic rotational

motion of anisotropic particles.

Finally, in simulations of a sedimenting lattice of 25 rods, the formation of instabilities and clusters is observed in high Pe systems (Pe $\sim$ 22, 44, and 90), but not in lower Pe systems (Pe $\sim$ 5.6 and 11). This seems to contradict what we see in our experimental systems, however since the instabilities observed in experiments tend to be on the order to 5 to 10 rod lengths in width, it is likely that the simulation systems are simply too small to reproduce the clustering observed in experiments.

Future Work

Future experimental work would likely focus on performing more experiments to compare the clustering of vertical and freely rotating rod colloids for the same colloid concentration to verify if the observation that the colloids tend to cluster more closely when held in a vertical orientation is true. In order to better characterize the clustering instabilities in the colloids one may wish perform sedimentation experiments for systems of different colloid concentrations and different temperatures (and therefore different Péclet numbers), as these seem to be variables that affect the formation and sizes of the clustering instabilities. On the other hand, if one were looking to study the dynamics of individual colloids during sedimentation in greater detail, one would need to perform experiments using a much faster imaging system or slow the

motion of the colloids by increasing the viscosity of the solvent (e.g. by decreasing the temperature). This would allow for better time resolution information and give better information on the rotational and diffusive motion of the colloids and make it much easier to track the trajectories of the colloids over time.

Future work on simulations of sedimenting colloids might involve finding ways to limit the acceleration of simulation colloids. This is because one of the issues we ran into in our simulations is that the colloids accelerate constantly in systems in which there is more than one colloid, even when the net force on the system is zero. If this problem can be resolved, it would allow for much longer simulations.

In the case of our 2 rod simulations, we found that the rods do not undergo periodic rotational motion as expected. Performing simulations of two rods sedimenting using different simulation box sizes, initial rod separations, and particle geometries would be useful in determining the cause of the behavior we observed in our current two rod simulations.

Finally, though we observed the formation of clusters in our 25 rod simulations, the systems are too small to be able to truly observe instabilities like those we observe in our experimental systems. Improving our understanding of the behavior of the Crowley instability at different Pe would require significantly larger systems of particles. Given that the current 25 rod simulations are performed in $(60\sigma)^3$ simula-

tion boxes and take around 10 hours to complete 2×10^5 time steps, increasing the size of the systems is quite feasible as the execution time scales roughly linearly with the number of fluid particles and the number of time steps. Since we observe instabilities in experimental systems with Pe $\sim$ 3 to 5, future simulations would likely focus on simulating systems with 0.1 <Pe $<$ 10 to find how significant Brownian motion must be to prevent the Crowley instability.

Bibliography

[1] J. M. Crowley. Viscosity-induced instability of a one-dimensional lattice of falling spheres. *Journal of Fluid Mechanics*, 45(1):151, 1971.

[2] R. G. Jones, J. Kavovec, R. Stepto, E .S. Wilks, M. Hess, T. Kitayama, and W. Val Metanomski. *Compendium of Polymer Terminology and Nomenclature*. International Union of Pure and Applied Chemistry, 2008.

[3] The Editors of Encyclopdia Britannica. Colloid. *Britannica Online Encyclopedia Britannica* http://www.britannica.com/science/colloid.

[4] A. Yethiraj. Tunable colloids: control of colloidal phase transitions with tunable interactions. *Soft Matter*, 3:1099, 2007.

[5] J. Dobnikar, A. Snezhko, and A. Yethiraj. Emergent colloidal dynamics in electromagnetic fields. *Soft Matter*, 9:3693, 2013.

[6] S. Ramaswamy. Issues in the statistical mechanics of steady sedimentation. *Advances in Physics*, 50(3):297, 2001.

[7] A. Kuijk, D. V. Byelov, A. V. Petukhov, A. van Blaaderen, and A. Imhof. Phase behavior of colloidal silica rods. *Faraday Discussions*, 159:181, 2012.

[8] D. Mukhija and M. J. Solomon. Nematic order in suspensions of colloidal rods by application of a centrifugal field. *Soft Matter*, 7(2):540, 2011.

[9] R. Chajwa. Instabilities in sedimentation at low reynolds numbers. Master's thesis, Department of Physical Sciences Indian Institute of Science Education and Research, 2015.

[10] R. Lahiri and S. Ramaswamy. Are steadily moving crystals unstable? *Physical Review Letters*, 79(6):1150, 1997.

[11] S. Jung, S. E. Spagnolie, K. Parikh, M. Shelley, and A-K. Tornberg. Periodic sedimentation in a stokesian fluid. *Physical Review*, 74:035302, 2006.

[12] Wikipedia Community. Péclet number. https://en.wikipedia.org/wiki/P%C3%A9clet_number, March 2016. Accessed - March 28, 2016.

[13] S. Hansen. Translational friction coefficients for cylinders of arbitrary axial ratios estimated by monte carlo simulation. *Journal of Chemical Physics*, 121(18):9111, 2004.

[14] A. Kuijk, A. van Blaaderen, and A. Imouf. Synthesis of monodisperse, rod-like silica colloids with tunable aspect ratio. *Journal of the American Chemical Society*, 133:2346, 2011.

[15] A. Kuijk, A. Imouf, M. H. W. Verkuijlen, T. H. Besseling, E. R. H. van Eck, and A. van Blaaderen. Colloidal silica rods: Material properties and fluorescent labeling. *Particle and Particle Systems Characterization*, 31(6):706, 2014.

[16] Glycerine Producers' Association. *Physical properties of glycerine and its solutions.* New York : Glycerine Producers' Association, 1963.

[17] T. H. Besseling, M. Hermes, A. Kuijk, B. de Nijs, T.-S. Deng, M. Dijkstra, A. Imhof, and A. van Blaaderen. Determination of the positions and orientations of concentrated rod-like colloids from 3d microscopy data. *Journal of Physics: Condensed Matter*, 27:194109, 2014.

[18] N. Moshtagh. Minimum volume enclosing ellipsoids. www.mathworks.com-/matlabcentral/fileexchange/9542-minimum-volume-enclosing-ellipsoid, 2006. Accessed - March 27, 2016.

[19] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62, 1975.

[20] P. P. Klein. On the ellipsoid and plane intersection equation. *Applied Mathematics*, 3:1634, 2012.

[21] D. Allan, T. A. Caswell, N. C. Keim, and C. van der Wel. trackpy v0.2.4. github.com/soft-matter/trackpy, 2014. Accessed - March 27, 2016.

[22] J. C. Crocker and D. G. Grier. Methods of digital video microscopy for colloidal studies. *Journal of Colloid and Interface Science*, 179(1):298, 1996.

[23] R. Rottenfusser, E. E. Wilson, and M. W. Davidson. The point spread function. Zeiss Education in Microscopy and Digital Imaging - http://zeiss-campus.magnet.fsu.edu/articles/basics/psf.html. Accessed - April 18, 2016.

[24] Hoomd-blue homepage. https://codeblue.umich.edu/hoomd-blue/.

[25] C. L. Phillips, J. A. Anderson, and S. C. Glotzer. Pseudo-random number generation for brownian dynamics and dissipative particle dynamics simulations on gpu devices. *Journal of Computational Physics*, 230:7191, 2011.

[26] T. D. Ngyuen, C. L. Phillips, J. A Anderson, and S. C. Glotzer. Rigid body constraints realized in massively-parallel molecular dynamics on graphics processing units. *Computer Physics Communications*, 182(2307), 2011.

[27] M. B. Liu, G. R. Liu, L. W. Zhou, and J. Z. Chang. Dissipative particle dynamics (dpd): An overview and recent developments. *Archives of Computational Methods in Engineering*, 22(4):529, 2015.

[28] D. S. Bolintineanu, G. S. Grest, J. B. Lechman, F. Pierce, S. J. Plimpton, and P. R. Schunk. Particle dynamics modeling methods for colloid suspensions. *Computational Particle Mechanics*, 1(3):321, 2014.

[29] W. Dzwinel, D. A.Yuen, and K. Boryczko. Mesoscopic dynamics of colloids simulated with dissipative particle dynamics and fluid particle model. *Journal of Molecular Modeling*, 8(1):33, 2002.

[30] J. Zhou and F. Schmid. A dissipative-particle-dynamics model for simulating dynamics of charged colloids. In W. E. Nagel et. al., editor, *High Performance Computing in Science and Engineering '13*, page 5, 2013.

[31] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1, 2005.

[32] N. A. Teanby. An icosahedron-based method for even binning of globally distributed remote sensing data. *Computers & Geosciences*, 32(9):1442, 2006.

[33] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. on Mathematical Software*, 22(4):469, 1996.

[34] Scipy Community. Scipy - gaussian kernel density function. docs.scipy.org-/doc/scipy-0.16.1/reference/generated/scipy.stats.gaussian_kde.html, 2015. Accessed - March 27, 2016.

[35] H. Hasimoto. On the periodic fundamental solutions of the stokes equations and their application to viscous flow past a cubic array of spheres. *Journal of Fluid Mechanics*, 5:317, 1959.

Appendix A

Ellipsoid Fitting Code

A.1 Minimum Volume Enclosing Ellipsoid calculation

In order to determine the position and orientation of the rod colloids in our images, we find the minimum volume enclosing ellipsoid (MVEE) for the rods. This is done using a Python implementation of Nima Moshtagh's MATLAB code [18]. The code we wrote is presented below:

```
# Python implementation of Nima Moshtagh's
# Minimum Volume enclosing ellipsoids MATLAB code
# http://www.mathworks.com/matlabcentral/fileexchange/9542-minimum-
    ↪ volume-enclosing-ellipsoid
import numpy as np
import numpy.linalg as linalg
# Returns A matrix for an ellipse and the center c
# such that (x-c)^T * A * (x-c) = 1
# defines the ellipse.
# P = d by N numpy array of N points in R^d
def MVEE(P,tol = 0.01):
  d, N = P.shape
  Q = np.zeros((d+1,N))
  Q[0:d,:] = P
```

```
    Q[d,:] = np.ones((1,N))
    count = 1
16  err = 1
    u = (1.0/N)*np.ones(N)
    u = u.transpose()

    while err > tol:
21    X = np.dot(np.dot(Q,np.diag(u)),Q.T)
      try:
        invX = linalg.inv(X)
        M = np.diag(np.dot(np.dot(Q.T,invX),Q))
        maxM = np.amax(M)
26      j = np.argmax(M)
        step_size = (maxM - d - 1)/((d+1)*(maxM - 1))
        new_u = (1-step_size)*u
        new_u[j] = new_u[j] +step_size
        count += 1
31      err = linalg.norm(new_u-u)
        u = new_u
      except:
        print "Error"
        return None
36  MatU = np.diag(u)

    u.transpose()
    c = np.dot(P,u)
    try:
41    A = (1./d)*linalg.inv(np.dot(np.dot(P,MatU),P.T)-np.multiply.
        ↪ outer(c,c))
    except:
      print "Error"
      return None
    return [A,c,count]
```

A.2 Ellipsoid Fitting Main Code

The main code used to read, pre-process, and find the MVE ellipsoids for the rod colloids. The code includes an algorithm used to find the intercepts of a plane with

the ellipsoids using a method described in [20]. It is run using the command:

```
import EllipsoidFit
E = EllipsoidFit.EllipsoidFit('image.tif', ... optional arguments
    ↪ ...)
E.dofit()
```

The EllipsoidFit.py code:

```
import numpy as np
import time
import tifffile as tiff
from scipy import ndimage, spatial
import sys
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import patches
import itertools as it
import MVEE
import EF_imageprocess as IP
'''
class: EllipsoidFit
Main class used to interact with the ellipsoid fitting code.
Includes the initialization, execution, and output code.
See MVEE.py for the math and HH_imageprocess.py for the image
    ↪ processing.

INPUTS:
image     [str/array] 3D image input. Either a string to specify the
    ↪  location of a .tiff image which is opened using tifffile, or
    ↪ an image array.
OPTIONS: (See default dictionary in EllipsoidFit for default values
    ↪ .)
rodwid    [float] expected rod width in pixels
rodlen    [float] expected rod length in pixels
threshold     [str, float] specifies the method used to determine
    ↪ the threshold intensity value used to identify voxels in a rod
    ↪ . Either a number which is used as the threshold or a string
    ↪ which specifies the method used. SEE: EF_imageprocess.
    ↪ thresholding() for more information on inputs. Makes heavy use
    ↪  of the skimage package.
voxelscale    [3 element list of floats] specifies the shape of the
    ↪ voxels. If the voxels are not cubic, the image will be
    ↪ stretched and interpolated so that the voxels become cubic.
outfile   [str] specifies the filename used to save the data and
    ↪ graph outputs. Default is either the name of the input file or
    ↪  the date, depending on the input method for the image.
```

```
    tol     [float] error tolerance used to end the MVEE fitting
    outputdata    [bool] toggles output of data files
    outputhist    [bool] toggles output of histograms
    displayhist   [bool] toggles display of histograms
    hull    [bool] toggles using the hull of groups of voxels during the
        ↪ MVEE
    smooth    [str] specifies the smoothing method used in pre-
        ↪ processing. SEE: EF_imageprocess.imagesmooth()
    quiet     [bool] toggles printing of debug information
    maxsize   [float] specifies maximum expected object size. Used for
        ↪ threshold refinements.
    minsize   [float] specifies minimum expected object size. Used for
        ↪ threshold refinements.
    refinethresh  [str] specifies type and setting for threshold
        ↪ refinements. SEE: EF_imageprocess.refinethresh()
    histtype  [list of str] specifies what values to print histograms
        ↪ for. Strings must be valid tags for the data dictionary.
    histnorm  [bool] toggles normalization of the output histrograms
    outzslice     [str, list of int] specifies which z-slices to use
        ↪ output with along with ellipsoid-plane intersections. If
        ↪ maxintens, the code chooses the slice with the highest mean
        ↪ intensity.
    '''

    class EllipsoidFit:
      default = {"rodwid": 5,
        "rodlen": 20,
        "threshold": "otsu␣global",
        "voxelscale":[1,1,1],
        "outfile":None,
        "tol": 0.01,
        "outputdata": True,
        "outputhist": True,
        "displayhist":False,
        "hull": True,
        "smooth": "gaussian␣1.0",
        "quiet": False,
        "maxsize": None,
        "minsize": None,
        "refinethresh": None,
        "histtype": ["len","theta","phi","orderparam","zcent","object␣
            ↪ vol"],
        "histnorm": True,
        "outzslice": "maxintens"}
```

```
  def __init__(self,image,**kwargs):
62    #Initialization function. Computes any values that are left
        ↪ unspecified.
      # Checks for invalid inputs and ends execution if any are found.
      for k in kwargs:
        if not k in self.default:
       print 'ERROR:␣'+k+'␣IS␣NOT␣A␣VALID␣INPUT.␣PLEASE␣CHECK␣FOR␣
          ↪ MISSPELLED␣OR␣INVALID␣INPUTS.'
67     sys.exit()

      # If not provided, set to default values
      for k in self.default.keys():
        if not k in kwargs:
72  kwargs[k] = self.default[k]

      # process outfile name
      if kwargs["outfile"] is None:
        if isinstance(image,str):
77  kwargs["outfile"] = image.split(".")[0]
        else:
    kwargs["outfile"] = time.asctime(time.localtime(time.time()))

      # calculate min/max sizes
82    expectedvol = np.pi*(kwargs["rodwid"]**2)*kwargs["rodlen"]/4
      if kwargs["maxsize"] is None:
        maxfactor = 1.5
        kwargs["maxsize"] = expectedvol*(maxfactor)**3
      if kwargs["minsize"] is None:
87      minfactor = 1./3.
        kwargs["minsize"] = expectedvol*(minfactor)**3

      # clean up refinethresh input
      if not isinstance(kwargs["refinethresh"],list):
92      kwargs["refinethresh"] = [kwargs["refinethresh"],1]

      # process image input
      if isinstance(image,str):
        imgtiff = tiff.TIFFfile(image)
97      image = imgtiff.asarray()
        image = image.swapaxes(0,2) # swap z and x such that coords
          ↪ are [x,y,z]
      # else if input is list convert to array
      elif isinstance(image,list):
        image = np.array(image)
102   # else if input is not already a numpy array, throw an error
```

```
    elif not isinstance(image, np.ndarray):
      sys.exit("Invalid input for image, must be a string or numpy
         ↪ array!")

    # set some values for convenience
    self.image = image
    self.ishape = image.shape
    self.dims = len(self.ishape)
    self.opts = kwargs
    self.minsize = kwargs['minsize']
    self.vosc = kwargs['voxelscale']
    self.quiet = kwargs['quiet']
    self.outfile = kwargs['outfile']
    self.vox = min(self.vosc)
    self.tol = kwargs['tol']
    self.info = {}
    del image,kwargs

  def dofit(self):
    #Executes ellipsoid fitting code and outputs.
    # Do image pre-processing SEE: imageprocess
    time1 = time.time()
    self.imageprocess()

    # Do ellipsoid fitting
    time2 = time.time()
    self.get_ellipsoids()
    self.info['Ellipsoid Count'] = len(self.data['x'])

    # Calculate various information on the ellipsoids (length,
       ↪ orientation, etc.)
    time3 = time.time()
    self.elldatacalc()
    time4 = time.time()

    # plot histograms of ellipsoid data
    if self.opts['outputhist'] or self.opts['displayhist']:
      self.graphdata()

    # output z-slices will ellipsoids overlayed
    if self.opts['outzslice']:
      self.overlayellipses()
    time5 = time.time()

    # Calculate the time required for various steps
```

```
      self.info['total time'] = time5-time1
147   self.info['image process time'] = time2-time1
      self.info['fit ellipsoids time'] = time3 - time2
      self.info['calc data time'] = time4-time3
      self.info['plotting time'] = time5-time4

152   # print output files
      if self.opts['outputdata']:
        self.metadataprint()
        self.data.to_csv(self.outfile+"_featuredata.csv")

157 def imageprocess(self):
      #Runs image processing
      # smooth
      if not self.quiet: print 'Smoothing ',self.opts['smooth'],'...'
      self.image = IP.imagesmooth(self.image,self.opts['smooth'])
162
      # rescale using voxelscale values
      if len(set(self.vosc))>1:
        if not self.quiet: print "Original image dimensions: ",self.
           ↪ ishape, "\n Rescaling ..."
        self.image = IP.cubifyvoxels(self.image,self.vosc)
167
        # save new image info
        self.vosc = [self.vox for i in range(self.dims)]
        self.ishape = self.image.shape
        self.info['rescaled voxel'] = self.vosc
172     self.info['rescale image size'] = self.ishape
        if not self.quiet: print "New Image dimentions: ", self.ishape

      # set outzslice:
      if 'outzslice' in self.opts and self.opts['outzslice'] =='
         ↪ maxintens':
177     self.opts["outzslice"] = [np.argmax(np.mean(self.image,axis
           ↪ =(0,1)))]

      # Threshold
      if not self.quiet: print "Thresholding, Method: ", self.opts["
         ↪ threshold"], "..."
      # find binary image
182   binimg, self.info['Threshold'] = IP.thresholding(self.image,self
         ↪ .opts['threshold'], minsize = self.minsize,quiet = self.
         ↪ quiet)
      # label regions of voxels above the threshold
      self.lbimg, self.objnum = ndimage.label(binimg)
```

```
    # Calculate the size of the regions
187 self.objsizes = IP.get_blobsize(self.lbimg,self.objnum)
    if not self.quiet: print "Threshold: {} \nObjects: {}".format(
      ↪ self.info['Threshold'],self.objnum)
    self.info['Initial Object Count'] = self.objnum

    # Refine threshold if required
192 if self.opts['refinethresh'][0]:
      if not self.quiet: print "Refining Thresholding, Method:",self
        ↪ .opts["refinethresh"]
      self.lbimg,self.objsizes = IP.refinethresh(self.image,self.
        ↪ lbimg,self.objsizes,self.minsize,self.opts["maxsize"],
        ↪ self.opts["refinethresh"][0],self.opts["refinethresh"
        ↪ ][1])
      self.objnum = len(self.objsizes)
      self.info["Refined Object Count"] = self.objnum
197   if not self.quiet: print "Done! New object count: ", self.
        ↪ objnum

  def get_ellipsoids(self):
    #Runs fitting of ellipsoids to the labelled thresholded regions
    hull = self.opts['hull']
202 # Stores ellipsoid Matrices and centroids
    tempdict = {'ellmat':[],'x':[],'y':[],'z':[],'zextent':[],'
      ↪ objsize':[],'MVEE_iter':[]}

    # loop through regions larger than the minimum size
    for i in [o[0] for o in self.objsizes if o[1] > self.minsize and
      ↪  not o[0] == 0]:
207   # Counter for the user
      if not self.quiet:
        sys.stdout.write("\r {0} / {1}".format(int(i),self.objsizes.
          ↪ shape[0]))
        sys.stdout.flush()

212   # Create binary image of the object in question only
      objindices = np.argwhere(self.lbimg == i)
      zextent = (np.amin(objindices[:,2]),np.amax(objindices[:,2]))
      # Check against the minimum size
      if hull:
217     # Attempt to get convex hull for the object
        try:
          hull = spatial.ConvexHull(objindices)
          points= np.transpose(objindices[hull.vertices])
```

```
            tempE = MVEE.MVEE(points,tol = self.tol)
222         # If this fails, get the ellipsoid of the object itself
            except:
              print "\nConvex Hull error"
              # Saves the matrices and centroid of the fitted ellipsoid
              tempE = MVEE.MVEE(np.transpose(objindices),tol=self.tol)
227           # MVEE returns None if there is an error

        else:
          tempE = MVEE.MVEE(np.transpose(objindices),tol=self.tol)
        if tempE:
232       # append ellipsoid data to data dictionary
          tempdict['objsize'].append(objindices.shape[0])
          tempdict['zextent'].append(zextent)
          tempdict['ellmat'].append(tempE[0])
          tempdict['x'].append(tempE[1][0])
237       tempdict['y'].append(tempE[1][1])
          tempdict['z'].append(tempE[1][2])
          tempdict['MVEE_iter'].append(tempE[2])

      self.data = pd.DataFrame(tempdict)
242   if not self.quiet:
        sys.stdout.write("\r {0} / {1}".format(self.objsizes.shape[0],
            ↪ self.objsizes.shape[0]))
        sys.stdout.flush()

    def elldatacalc(self):
247   #Calculation of information on the fitted ellipsoids.
      data = []
      for i in range(len(self.data['ellmat'])):
        E = self.data['ellmat'].ix[i]

252     U,Q,V = np.linalg.svd(E) # single value decomposition
        eivals,eivecs = np.linalg.eig(E) # eigenvalues
        primax = eivecs[:,np.argmin(eivals)] # determine longest axis
        lengths = np.sqrt(1/Q) # lengths of the axes

257     # input data into data dictionaries in a pandas friendly
            ↪ format
        datadict = {}

        # Calculate orientations
        datadict['costheta'] = primax[2]/np.linalg.norm(primax)
262     datadict['theta'] = np.arccos(datadict['costheta'])
        datadict['phi'] = np.arctan(primax[1]/primax[0])
```

```
        # transpose eigenvectors so that the vectors are on the along
            ↪ the columns
        datadict["eigenvectors"] = np.transpose(eivecs)
267     datadict["eigenvalues"] = eivals
        datadict["rotation␣matrix"] = V
        datadict["principal␣numbers"] = Q
        datadict["lengths"] = lengths
        datadict["len"] = max(lengths)*2 # lengths of the longest axis
272     datadict["ellipsoid␣vol"] = 4*np.pi*lengths[0]*lengths[1]*
            ↪ lengths[2]/3

        # a comparison of the volumes of the regions and the ellipsoid
        # is a measure of how "close" to an ellipsoid the regions are
        datadict["ellipsoidiness"] = self.data["objsize"].ix[i]/
            ↪ datadict["ellipsoid␣vol"]
277     datadict["aspect␣ratio"] = datadict['len']*2/(lengths[1]+
            ↪ lengths[2])

        # Average of the shorter lengths
        datadict["width"] = (lengths[1]+lengths[2])/2
        data.append(datadict)
282   self.data = pd.merge(self.data,pd.DataFrame(data),left_index=
          ↪ True,right_index=True)

  def graphdata(self):
  #Plot histograms for specified values
    for graph in self.opts['histtype']:
287     print "Plotting␣",graph,"..."
        try:
          fig = plt.figure()
          ax = fig.add_subplot(111)
          n,bins,patchs = ax.hist([d for d in self.data[graph] if
              ↪ isinstance(d,(float,int,long))],normed=self.opts['
              ↪ histnorm'],bins=50)
292       ax.set_ylabel("Quantity")
          ax.set_xlabel(graph+"(bin␣size␣=␣"+str(bins[1]-bins[0])+")")
          ax.set_title('Histogram␣of␣'+graph)
          if self.opts['outputhist']: fig.savefig(self.outfile+"_"+
              ↪ graph+"hist.png")
          if self.opts['displayhist']: fig.show()
297       plt.close(fig)
        except:
          print 'Invalid␣'
          pass
```

```
302   def metadataprint(self):
      #Print image information and average values from the ellipsoids.
        hline = "\n␣
           ↪ ------------------------------------------------------------"
        f = open(self.outfile+"_metadata.dat","w")
        f.write(self.outfile+"_metadata.dat␣...␣"+time.asctime(time.
           ↪ localtime(time.time())))
307     f.write(hline)
        f.write("\n␣USER␣INPUTS:␣")
        for i in self.opts.keys():
          f.write("\n␣{0:40s}␣{1:10s}".format(str(i),str(self.opts[i])))
        f.write(hline)
312     f.write('\n␣IMAGE␣INFO:␣')
        for i in self.info.keys():
          f.write("\n␣{0:40s}␣{1:10s}".format(str(i),str(self.info[i])))
        f.write(hline)
        f.write("\n␣FITTING␣AVERAGES:")
317     for i in self.data:
          if isinstance(self.data[i][0],(float,int,long,np.int64)):
            f.write("\n␣{0:40s}␣{1:10f}".format("Mean␣"+i+":␣",self.data
               ↪ [i].mean()))
            f.write("\n␣{0:40s}␣{1:10f}".format("Median␣"+i+":␣",self.
               ↪ data[i].median()))
            f.write("\n␣{0:40s}␣{1:10f}".format("Standard␣Deviation␣"+i+
               ↪ ":␣",self.data[i].std()))
322         f.write("\n␣
               ↪ ------------------------------------------------------------
               ↪ ")
        f.close()

      def overlayellipses(self):
        #Print ellipsoids intersections with selected z-slices.
327     #Based on "On the Ellipsoid and Plane Intersection Equation" by
           ↪ Peter Paul Klein (2012)
        z = self.opts['outzslice']
        if isinstance(z,int):
          z = [z]
        for i in z:
332       ellinds = range(len(self.data['x']))
          img = self.image[:,:,i]
          fig = plt.figure()
          ax = fig.add_subplot(111)
          # plot selected z-slice
337       ax.imshow(img,interpolation = "nearest",cmap="gray")
```

```
        ax.set_ylabel("y")
        ax.set_xlabel("x")
        ells = []
        for e in ellinds:
342       # collect data to use for calculating the ellipsoid-plane
             ↪ intersection
          cent = [self.data['x'].ix[e],self.data['y'].ix[e],self.data[
             ↪ 'z'].ix[e]]
          A = self.data["ellmat"].ix[e]
          eivecs = self.data["eigenvectors"].ix[e]
          eivals = self.data["eigenvalues"].ix[e]
347       rot = self.data["rotation␣matrix"].ix[e]

          # reorganize eigenvectors and eigenvalues
          eivecs,eivals = reorg_eig(eivecs,eivals,rot)
          roti = np.linalg.inv(rot)
352
          # Find a point on the selected plane that is also in the
             ↪ ellipsoid
          q = findellpnt(A,i,cent,eivecs,eivals)
          if not q is None:
            n = rot[:,2]
357
            # Gets the ellipses from the ellipsoid place intersection
            vecs, m, elens = ellplaneintersect(eivals,np.dot(rot,np.
               ↪ array(q)),n)

            # determine which direction the major axis is in.
362         l1 = 0 if elens[0] > elens[1] else 1
            l2 = 1 if elens[0] > elens[1] else 0

            # rotates the vectors
            anglevec = np.dot(roti,vecs[l1])
367         angle = np.arctan(anglevec[1]/anglevec[0])*180/np.pi
            m1 = np.dot(roti,m)
            angle = -1*(angle)

            # Sets up matplotlib patches the ellipsoids on the image
372         ells.append(patches.Arc((m1[0:2]+np.array(cent)[0:2])
               ↪ [[1,0]],elens[l2],elens[l1],angle=angle,color
               ↪ =[1,0,0]))
        for e in ells:
          # Plots ellipsoids on the image
          ax.add_patch(e)
          e.set_clip_box(ax.bbox)
```

```
377         # save image
            fig.savefig(self.outfile+"_overlay_b"+str(i)+".png",
               ↪ bbox_inches='tight')


    def reorg_eig(eivecs,eivals,rot):
382       # Reorganizes the eigenvalues and eigenvectors
        a = np.argmax(np.absolute(rot.dot(eivecs[0])))
        b = np.argmax(np.absolute(rot.dot(eivecs[1])))
        c = np.argmax(np.absolute(rot.dot(eivecs[2])))
        t = [a,b,c]
387     t = np.argsort(t)
        tempval = np.zeros(3)
        tempvec = np.zeros((3,3))
        tempvec[0] = eivecs[t[0]]
        tempvec[1] = eivecs[t[1]]
392     tempvec[2] = eivecs[t[2]]
        tempval[0] = eivals[t[0]]
        tempval[1] = eivals[t[1]]
        tempval[2] = eivals[t[2]]

397     return tempvec, tempval

    def findellpnt(A,z,cent,eivecs,eivals):
        # Finds a point in the plane that is also in the ellipsoid.
        eivecs = eivecs/eivals
402     # ellipsoid matrix A, z coord z, centroid cent
        # Find bounding rectangle
        xr = int(np.amax(np.absolute(eivecs[:,0])))+2
        yr = int(np.amax(np.absolute(eivecs[:,1])))+2
        zc = z - cent[2]
407     area = it.product(range(-1*xr,xr),range(-1*yr,yr))

        # check if any of the points are in the ellipsoid
        for p in area:
          point = {2:np.array(list(p)+[zc]),
412          1:np.array([p[1],zc,p[0]]),
             0:np.array([zc]+list(p))}[2]
          check = np.dot(point,np.dot(A,point))
          if check < 1:
            return point
417     return None

    def ellplaneintersect(eivals,q,n):
        # Calculates ellipsoid-plane intersection.
```

```
    # q is a point in the ellipsoid, n is the direction of the main
      ↪ axis
422 # Extended version of Klein 2012 algorithm
    if np.linalg.norm(n) != 1.0:
      n= np.array(n)/np.linalg.norm(n)
    D1 = np.diag(np.sqrt(eivals))
    s1 = np.cross(n,[1,0,0])
427 r1 = np.cross(n,s1)

    D1s1 = np.dot(D1,s1)
    D1r1 = np.dot(D1,r1)
    DRDS1 = np.dot(D1r1,D1s1)
432 DRDR1 = np.dot(D1r1,D1r1)
    DSDS1 = np.dot(D1s1,D1s1)

    omega = np.arctan(2*DRDS1/(DRDR1 - DSDS1))/2.
    r = np.cos(omega)*r1+np.sin(omega)*s1
437 s = -1*np.sin(omega)*r1+np.cos(omega)*s1
    r = r/np.linalg.norm(r)
    s = s/np.linalg.norm(s)

    DQ = np.dot(D1,q)
442 DR = np.dot(D1,r)
    DS = np.dot(D1,s)
    DQDR = np.dot(DQ,DR)
    DQDS = np.dot(DQ,DS)
    DSDS = np.dot(DS,DS)
447 DRDR = np.dot(DR,DR)
    r0 = -1*DQDR/DRDR
    s0 = -1*DQDS/DSDS
    d = np.dot(DQ,DQ)-DQDR**2/DRDR-DQDS**2/DSDS
    A = np.sqrt((1-d)/DRDR)
452 B = np.sqrt((1-d)/DSDS)
    m = q+r0*r+s0*s
    return (r,s),m,(A,B)
```

A.3 Image Processing

Python functions used to pre-process and find the threshold for ellipsoid fitting.

```
from scipy import ndimage, spatial
from skimage import filters, morphology
import numpy as np
'''
FUNC: refinethresh
PURPOSE: iteratively refines the thresholding of parts of the image
    ↪ in order to separate rods with overlapping PSFs.
INPUT:
  image [np.ndarray]: Numpy array containing the image info
  lblimg [np.ndarray]: numpy array with thresholded objects labeled
     ↪ by scipy.ndimage.label
  objsizes [list]: List of information on the sizes of the objected
     ↪ labeled (from get_blobsizes).
  minsize [double]: Minimum object threshold. Objects under this
     ↪ threshold are ignored
  maxsize [double]: Maximum object threshold. Objects over this
     ↪ threshold are re-thresholded
  threshtype [str]: Defines what type of threshold is used. (SEE:
     ↪ thresholding)
  iterations [int]: Defines the number of iterations the refinement
     ↪ goes through
OUTPUT:
  lbimg [np.ndarray]: A new array containing the new labelled
     ↪ regions
  objsizes [list]: list containing the sizes of the new labelled
     ↪ regions added to the old ones

OUTLINE:
       For the given number of iterations, identify any objects that
          ↪  are above the maxsize. Then isolate them, set the
          ↪ background to their minimum value and calculate the new
          ↪  threshold. Add the new objects to the lblimg and
          ↪ objsizes.
NOTES:
  Generally only worthwhile if the PSFs of the rods are severely
     ↪ overlapped. One should be careful not to set the maxsize to
     ↪ small, or good rod objects might get re-thresholded. This
     ↪ will cause a lot of rods to shrink in size, so if size
     ↪ statistics are important to you, I would advise finding some
     ↪  other method to fix issues with the PSF.
'''
def refinethresh(image,lblimg,objsizes,minsize,maxsize,threshtype,
   ↪ iterations):
  for j in range(iterations):
    # Max label index, used to add new objects
```

```
    lbind = max([o[0] for o in objsizes])+1

    # Identify which objects are above the maxsize threshold
    bigparticles = [o[0] for o in objsizes if o[1]>maxsize and o[0]>
       ↪  0]
    print "Iteration: ",j," Re-thresholding: ",len(bigparticles)

    initlbind = lbind
    refthreshs = []
    for i in bigparticles:
      # Find the points in the object
      blobpnts = lblimg == int(objsizes[i][0])

      # Set background appropriate to the minimum intensity in the
         ↪ blob
      # note the background can't be zero for the histogram based
      # thresholding methods
      blobimg = np.full(image.shape,np.amin(image[blobpnts]))

      # Get image of only the object in question
      blobimg[blobpnts] = image[blobpnts]
      # thresholding, labelling, etc.
      blobbin, thresh = thresholding(blobimg,threshtype,minsize=
         ↪ minsize)
      bloblab, objnum = ndimage.label(blobbin)
      refthreshs.append(thresh)

      blobobj = get_blobsize(bloblab,objnum)
      lblimg[blobpnts] = 0 # Set pixel intensities for the object in
         ↪  question to zero
      firstblob = True # the first blob is placed in the ith
         ↪ position, the rest are appended to the end of the list
      for j in [o[0] for o in blobobj if o[0]>0 and o[1] > minsize]:
        # Add new objects to the labelled image
        if firstblob:
          lblimg[bloblab == j] = i
          objsizes[i][1] = blobobj[j][1]
          firstblob = False
        else:
          lblimg[bloblab == j] = lbind
          objsizes = np.append(objsizes,[(lbind,blobobj[j][1])],axis
             ↪ =0)
          lbind += 1
    if len(bigparticles): print "Threshold Avg: ",np.mean(np.array(
       ↪ refthreshs))," Num Added: ",lbind-initlbind
```

```
  return lblimg, objsizes
# formats the histogram for the labelled objects
def get_blobsize(lbimg,objnum):
  objsizes = np.histogram(np.ravel(lbimg),bins=range(objnum+2))
  objsizes = np.array([[objsizes[1][i],objsizes[0][i]] for i in
     ↪ range(objnum+1)])
  return objsizes

#
   ↪ ----------------------------------------------------------------------
   ↪
'''
FUNC: imagesmooth
PURPOSE: handles the smoothing of the image
INPUT:
  image [np.ndarray]: input image as a numpy array
  smooth [str]: Sets smoothing type. (median, maximum, minimum,
     ↪ gaussian, or None). If gaussian, add a double to the string
     ↪ to set the sigma. If median, maximum, minimum, add an int to
     ↪  the string to set the radius
OUTPUT:
  image [np.ndarray]: The smoothed image
NOTES: Uses the smoothing functions provided by scipy.ndimage.
   ↪ filters
'''

def imagesmooth(image,smooth):
  smooth = smooth.split()
  if smooth[0] == "median":
    if len(smooth) == 1:
      rad = 1
    else:
      rad = int(smooth[1])
    image = ndimage.filters.median_filter(image,footprint =
       ↪ morphology.ball(rad))
  elif smooth[0] == "maximum":
    if len(smooth) == 1:
      rad = 1
    else:
      rad = int(smooth[1])
    image = ndimage.filters.maximum_filter(image,footprint =
       ↪ morphology.ball(rad))
  elif smooth[0] == "minimum":
    if len(smooth) == 1:
      rad = 1
```

```
    else:
      rad = int(smooth[1])
    image = ndimage.filters.minimum_filter(image,footprint =
       ↪ morphology.ball(rad))
  elif smooth[0] == "gaussian":
    if len(smooth) == 2:
      sigma = float(smooth[1])
    elif len(smooth) == 4:
      sigma= [float(smooth[i+1]) for i in range(3)]
    else:
      sigma = 1.0
    image = ndimage.filters.gaussian_filter(image,sigma)
  return image

#
   ↪ ----------------------------------------------------------------------
   ↪
'''
FUNC: cubifyvoxels
PURPOSE: Uses given voxel scaling and tricubic interpolation to
   ↪ convert voxel sizes to cubes
'''
def cubifyvoxels(image,voxelscale):
  vox = min(voxelscale)
  zoomimg = ndimage.interpolation.zoom(image,[v/vox for v in
     ↪ voxelscale])
  return zoomimg
#
   ↪ ----------------------------------------------------------------------
   ↪
'''
FUNC: thresholding
PURPOSE: calculates a suitable threshold and returns a binary 3D
   ↪ image
INPUT:  image [np.ndarray] - the image of interest
        threshold [str or int] - SEE: threshold in EllipseFit
OUTPUT: binimg - binary thresholded image
        threshold - calculated threshold
NOTES:  Currently uses threshold finding methods available through
   ↪ skimage. May add more methods later.
'''
def thresholding(image, threshold,minsize = 10,quiet = False):
  if isinstance(threshold,int) or isinstance(threshold,float):
    thresh = threshold
  elif isinstance(threshold,str):
```

```
    # Assume its Ok to use the same threshold for each layer.
    parsestr = threshold.split(" ")
    parsestr = [i.lower() for i in parsestr]
    if "otsu" in parsestr:
      if "local" in parsestr:
        ind = np.argmax([np.mean(i) for i in image])
        if len(parsetr) > 1:
          radius = int(parsestr[2])
        else:
          radius = 20
        mask = morphology.disk(radius)
        thresh = filters.rank.otsu(image[0],mask)
        return thresh, 0
      else:
        thresh = filters.threshold_otsu(image)
    if "li" in parsestr:
      thresh = filters.threshold_li(image)
    if "yen" in parsestr:
      thresh = filters.threshold_yen(image)
  threshinds = image<thresh
  binimg =  np.ones(image.shape)
  binimg[threshinds] = 0
  return binimg, thresh
```

Appendix B

Simulation System Generation

B.1 Geodesic Generation

Python implementation of Nick Teanby's geodesic spherical grid generation algorithm [32]. This is used to generate spherical colloidal particles for our simulations.

```
# generates a geodesic grid
# implementation of N.A. Teanby's algorithm
import numpy as np
import numpy.linalg as la

class geodesic:
    tol = 1E-3
    phi = 2*np.cos(np.pi/5)
    # Coordinates of vertices of original Icosahedron
    icos = np.array([[0,phi,1],[0,-phi,1],[0,phi,-1],[0,-phi,-1],
        [1,0,phi],[-1,0,phi],[1,0,-phi],[-1,0,-phi],
        [phi,1,0],[-phi,1,0],[phi,-1,0],[-phi,-1,0]])/(1+phi**2)
            ↪ **(0.5)
    icostriangles = np.array([[icos[0],icos[2],icos[8]],
        [icos[0],icos[2],icos[9]],
        [icos[0],icos[4],icos[5]],
        [icos[0],icos[4],icos[8]],
        [icos[0],icos[5],icos[9]],
        [icos[1],icos[3],icos[10]],
        [icos[1],icos[3],icos[11]],
        [icos[1],icos[4],icos[5]],
        [icos[1],icos[4],icos[10]],
        [icos[1],icos[5],icos[11]],
```

```
      [icos[2],icos[6],icos[7]],
      [icos[2],icos[6],icos[8]],
      [icos[2],icos[7],icos[9]],
      [icos[3],icos[6],icos[7]],
      [icos[3],icos[6],icos[10]],
      [icos[3],icos[7],icos[11]],
      [icos[4],icos[8],icos[10]],
      [icos[5],icos[9],icos[11]],
      [icos[6],icos[8],icos[10]],
      [icos[7],icos[9],icos[11]]])

  def __init__(self):
    # initializes points and triangles
    self.points = self.icos
    self.triangles = self.icostriangles
  def trianglediv(self,t,l):
    # Finds new triangle vertices
    newp = [(t[0]+t[1])/la.norm(t[0]+t[1]),
        (t[2]+t[0])/la.norm(t[2]+t[0]),
        (t[1]+t[2])/la.norm(t[1]+t[2])]

    # Adds points to new
    addpoints = np.array(newp)
    keepflag = [1,1,1]
    for p in self.points:
      for n in range(len(addpoints)):
        if la.norm(addpoints[n]-p) < self.tol:
          keepflag[n] = 0
    addpoints = addpoints[np.where(keepflag)]
    if len(addpoints > 0): self.points = np.append(self.points,np.
       ↪ array(addpoints),axis=0)
    if l > 1:
      self.trianglediv([t[0],newp[0],newp[1]],l-1)
      self.trianglediv([t[1],newp[0],newp[2]],l-1)
      self.trianglediv([t[2],newp[1],newp[2]],l-1)

  def geodesicgen(self,radius,level,cent=[0,0,0]):
    print "Generating␣a␣Geodesic␣Grid␣with␣", 2+10*4**level,"␣points
       ↪ ."
    for t in self.triangles:
      self.trianglediv(t,level)
    self.points = self.points*radius+cent
```

B.2 Rod Particle Generation

Generates rod colloid surface interaction points.

```
import geodesicgen
import numpy as np
import numpy.linalg as la

def genrod(rad,length,level):
  # Generate Geodesic for end caps
  g = geodesicgen.geodesic()
  g.geodesicgen(rad,level)
  circlepnts= np.linspace(0,2*np.pi,round(2*np.pi*rad*2))
  rr,cc = np.cos(circlepnts)*rad,np.sin(circlepnts)*rad

  cylinder = []
  count = 0
  for i in np.arange(-length/2+rad,length/2-rad+0.1,1/2.):
  for j in range(len(rr)):
    cylinder.append([rr[j],cc[j],i])
    #cylinder.append([rr[j]*(-1)**(count%2),cc[j]*(-1)**(count%2),i
      ↪ ])
  count+=1
  cylinder = np.array(cylinder)

  topcap = g.points[np.where(g.points[:,2]>=0)]+[0,0,length/2-rad]
  botcap = g.points[np.where(g.points[:,2]<=0)]+[0,0,-length/2+rad]
  print len(topcap),len(topcap)
  return np.concatenate((cylinder,topcap,botcap),axis=0)
```

B.3 Simulation System Generation

Sample code used to generate the initial .xml file for a 5 vertical rod DPD HOOMD-Blue simulation.

```
import numpy as np
```

```
import genrod
from numpy import random
# Generated rotation matrix from given lists of theta and phi
# Where theta is the rotation from the xy plane about the x axis and
# phi is the rotation about the z-axis.
def get_rotmat(THETA,PHI,num=1):
  rotmat = []
  if num == 1:
    if not isinstance(THETA,(int,float,long)):
      THETA = THETA[0]
    if not isinstance(PHI,(int,float,long)):
      PHI = PHI[0]
    cosph = m.cos(PHI)
    sinph = m.sin(PHI)
    costh = m.cos(THETA)
    sinth = m.sin(THETA)

    return [np.array([[cosph*costh,-1*sinph,cosph*sinth],
           [sinph*costh,cosph,sinph*sinth],
           [-1*sinth,0,costh]])]
  else:
    for i in range(num):
      cosph = m.cos(PHI[i])
      sinph = m.sin(PHI[i])
      costh = m.cos(THETA[i])
      sinth = m.sin(THETA[i])
      rotmat.append(np.array([[cosph*costh,-1*sinph,cosph*sinth],
           [sinph*costh,cosph,sinph*sinth],
           [-1*sinth,0,costh]]))
  return rotmat
#-----
def config(Lx = 60,Ly = 60,Lz = 30,length = 9.,rad=2.,level=2,theta
   ↪ =0,phi=0,rodmass=500.0):
  nrod = 5
  pos = np.array([[0,12,0],[0,6,0],[0,0,0],[0,-6,0],[0,-12,0]])

  rotmat = get_rotmat(theta,phi)
  NLJ = 3*(length-2*rad)
  L = length-2*rad
  L,L2,L3 =L,L**2, L**3
  R,R2,R3,R4 = rad, rad**2, rad**3, rad**4

  # Adjust random number counter to account for empty space left
     ↪ around colloids/walls
  V1 = nrod*np.pi*R3/3+2*np.pi*R*(L-2*R)
```

```
45  V2 = nrod*np.pi*(R+1.2)**2/3+2*np.pi*(R+0.7)*(L-2*R)
    V3 = 2*Lx*Ly
    V = Lx*Ly*Lz
    nrand = int(3*V**2/(V-V2+V1-V3))

50  # Generate central LJ point and surface interaction point
       ↪ coordinates
    LJpnts = np.array([np.zeros(NLJ),np.zeros(NLJ),np.linspace(-length
       ↪ /2+rad,length/2-rad,NLJ)]).T
    g = genrod.genrod(rad,length,level)
    LJpnts = np.append(LJpnts,[[0,0,0]],axis=0)

55  if rodmass is None:
          rodmass = V1*3/nrod
    ngeo = 2+10*4**level+16

    # Calculate masses of surface points
60  H = length - 2*rad
    mhemi = (rodmass*(3/20.)*(15*H**3+104*H**2*R+180*H*R**2+96*R**3)
       ↪ /((3*H+4*R)*(7*H**2+18*H*R+12*R**2)))
    capdens = 2*mhemi/ngeo
    mcyl = (rodmass*(3/10.)*H*(25*H**2+58*H*R+36*R**2)/((3*H+4*R)*(7*H
       ↪ **2+18*H*R+12*R**2)))
    cyldens = mcyl/(len(g[:,0])-ngeo)
65  centmass = rodmass - mcyl - mhemi*2

    # Rotate colloid
    for i in range(len(g)):
      g[i] = rotmat[0].dot(g[i])
70  for i in range(len(LJpnts)):
      LJpnts[i] = rotmat[0].dot(LJpnts[i])

    n = len(g[:,0])
    nlj = len(LJpnts[:,0])
75  print n,nlj
#-----
    # Write Wrapping
    f = open("init.xml","w")
    f.write("<?xml␣version=\"1.0\"␣encoding=\"UTF-8\"?>␣\n"+
80 "<hoomd_xml>␣\n"+
"<configuration␣time_step=\"0\">\n")

    # box size
    f.write("<box␣Lx=\""+str(Lx)+"\"␣Ly=\""+str(Ly)+"\"␣Lz=\""+str(Lz)
       ↪ +"\"/>\n")
```

```
85
    # point positions
    f.write("<position>\n")
    for i in range(nrod):
      for p1 in g:
90      p = p1+pos[i,:]
        f.write(str(p[0])+"␣"+str(p[1])+"␣"+str(p[2])+"\n")
      for p1 in LJpnts:
        p = p1+pos[i,:]
        f.write(str(p[0])+"␣"+str(p[1])+"␣"+str(p[2])+"\n")
95  for i in range(nrand):
      p = random.rand(3)*[Lx,Ly,Lz]-[Lx/2.,Ly/2.,Lz/2.]
      flag1 = True
      for j in range(nrod):
        if np.min(np.linalg.norm(p-LJpnts-pos[j,:],axis=1)) < rad+1.2
           ↪ or p[2] <-Lz/2.+1 or p[2] > Lz/2.-1:
100       flag1 = False
      if flag1: f.write(str(p[0])+"␣"+str(p[1])+"␣"+str(p[2])+"\n")
      else: nrand -=1
    f.write("</position>\n")

105 # Define particle type
    f.write("<type>\n")
    for j in range(nrod):
      for i in range(n):
        f.write("C\n")
110   for i in range(nlj):
        f.write("CLJ\n")
    for i in range(nrand):
      f.write("F\n")
      f.write("</type>\n")
115
    # Define Particle Diameter
    f.write("<diameter>\n")
    for j in range(nrod):
      for i in range(n):
120     f.write("0.0\n")
      for i in range(nlj):
        f.write(str(R*2)+"\n")
    for i in range(nrand):
      f.write("0.0\n")
125 f.write("</diameter>\n")

    # Define Particle body
    f.write("<body>\n")
```

```
  for j in range(nrod):
    for i in range(n+nlj):
      f.write(str(j)+"\n")
  for i in range(nrand):
    f.write("-1\n")
  f.write("</body>\n")

  # Define Particle mass
  f.write("<mass>\n")
  for j in range(nrod):
    for i in range(n-ngeo):
      f.write(str(cyldens)+"\n")
    for i in range(ngeo):
      f.write(str(capdens)+'\n')
    for i in range(nlj-1):
      f.write("0.0\n")
    f.write(str(centmass)+"\n")
  for i in range(nrand):
    f.write("1.0\n")
  f.write("</mass>\n")

  # Write Wall
  f.write('<wall>\n')
  f.write('<coord␣ox=\"0.0\"␣oy=\"0.0\"␣oz=\"-'+str(Lz/2.)+'\"␣nx
     ↪ =\"0.0\"␣ny=\"0.0\"␣nz=\"1.0\"/>\n')
  f.write('<coord␣ox=\"0.0\"␣oy=\"0.0\"␣oz=\"'+str(Lz/2.)+'\"␣nx
     ↪ =\"0.0\"␣ny=\"0.0\"␣nz=\"1.0\"/>\n')
  f.write('</wall>\n')

  # Write Wrapping
  f.write("</configuration>\n"+
  "</hoomd_xml>\n")
  f.close()
```

Appendix C

HOOMD-Blue Simulation Code

Sample code used to run a HOOMD-Blue DPD colloid sedimentation simulation. This particular code is used for a simulation of 5 rod particles. Note: We first generate the initial .xml files ("init.xml"), then run the HOOMD code. To visualize the simulation results using vmd, one needs to generate a separate initial .xml file with only the points in the colloids and using the command `vmd -hoomd rodpoints.xml rodpoints.dcd`

```
from hoomd_script import *
from numpy import random
import numpy as np
import os

# system settings
Lx, Ly, Lz = 40,90,60
length = 12.
rad = 2.
level = 2

theta = np.pi/2

t_warmup = 5E3
t_run = 195E3
t_step = 0.005
n = int(t_run/t_step)
nrod = 5
T = 0.2
forcez = -0.04*163/628

```

```
dirname = 'mar10_16_5rod_T'+str(T)
os.chdir(dirname)

# Open output .csv file and write column headers
flist = []
for k in range(nrod):
  flist.append(open('colloid_data_rod'+str(k)+'.csv','w'))

  flist[k].write("time,num,mass,MIx,MIy,MIz,")
  flist[k].write("COMx,COMy,COMz,velx,vely,velz,orientRe,orientIm_i,
     ↪ orientIm_j,orientIm_k,")
  flist[k].write("AMx,AMy,AMz,forcex,forcey,forcez,torquex,torquey,
     ↪ torquez\n")
f1 = open('fluiddata.csv','w')
f1.write('fvelx,fvely,fvelz\n')
def writebody(t):
# Writes information on the rod to a .csv file
  if t >0:
    for k in range(nrod):
      col = system.bodies[k]
      flist[k].write(str(t)+",␣"+str(col.num_particles)+",␣"+str(col
         ↪ .mass)+",␣"+str(col.moment_inertia[0])+",␣"+str(col.
         ↪ moment_inertia[1])+","+str(col.moment_inertia[2])+",␣")
      flist[k].write(str(col.COM[0])+",␣"+str(col.COM[1])+",␣"+str(
         ↪ col.COM[2])+",␣"+str(col.velocity[0])+",␣"+str(col.
         ↪ velocity[1])+",␣"+str(col.velocity[2])+","+str(col.
         ↪ orientation[0])+",␣"+str(col.orientation[1])+",␣"+str(
         ↪ col.orientation[2])+",␣"+str(col.orientation[3])+",␣")
      flist[k].write(str(col.angular_momentum[0])+",␣"+str(col.
         ↪ angular_momentum[1])+",␣"+str(col.angular_momentum[2])+"
         ↪ ,␣"+str(col.net_force[0])+","+str(col.net_force[1])+",␣"
         ↪ +str(col.net_force[2])+",␣"+str(col.net_torque[0])+",␣"+
         ↪ str(col.net_torque[1])+","+str(col.net_torque[2])+"\n")

      del col
    ss = system.take_snapshot()
    fvelmean = np.mean(ss.particles.velocity,axis=0)
    f1.write(str(fvelmean[0])+','+str(fvelmean[1])+','+str(fvelmean
       ↪ [2])+'\n')
    del ss

# read in the initial configuration

system = init.read_xml(filename="init.xml",restart='restart.xml')
```

```
dpd = pair.dpd(r_cut = 1.0,T=T,seed=random.randint(0,100000))
dpd.pair_coeff.set('F', 'F', A=0.0, gamma = 5.0)
dpd.pair_coeff.set('F', 'C', A=0.0, gamma = 10.0)
dpd.pair_coeff.set('C', 'C', A=0.0, gamma = 0)
dpd.pair_coeff.set('C', 'CLJ', A=0.0, gamma = 0)
dpd.pair_coeff.set('CLJ', 'CLJ', A=0.0, gamma = 0)
dpd.pair_coeff.set('F', 'CLJ', A=0.0, gamma = 0)

# Define lennard jones force between fluid and central particles
lj = pair.slj(r_cut=2**(1./6))
lj.pair_coeff.set('F', 'CLJ',epsilon = 1.0,sigma=1.0)
lj.pair_coeff.set('F', 'C',epsilon = 0.0,sigma=0.0)
lj.pair_coeff.set('C', 'C',epsilon = 0.0,sigma=0.0)
lj.pair_coeff.set('CLJ', 'C',epsilon = 0.0,sigma=0.0)
lj.pair_coeff.set('CLJ', 'CLJ',epsilon = 1.0,sigma=1.0)
lj.pair_coeff.set('F', 'F',epsilon = 0.0,sigma=0.0)

integrate_mode = integrate.mode_standard(dt=t_step)
non_rigid = integrate.nve(group=group.nonrigid())

dump.xml(filename='restart.xml',restart=True,period = 1000,all=True)
dump.dcd(filename='rodpoints.dcd',period=500,group=group.rigid())
analyze.log(filename='thermo.log',quantities=['temperature','
    ↪ pressure'],period=500)

# Relaxation time
run(t_warmup,callback = writebody,callback_period=20)

# Start integration for colloid particles
rigid = integrate.nve_rigid(group=group.rigid())
force.constant(fx=0.0,fy=0.0,fz=forcez,group = group.rigid())

# apply force to fluid
ffluid =  -1*(forcez/628)*nrod/(len(system.particles)-628*nrod)
force.constant(fx=0.0,fy=0.0,fz=ffluid,group = group.nonrigid())

run(t_run,callback = writebody,callback_period=20)
for k in range(nrod):
        flist[k].close()
del flist
f1.close()
del lj, dpd, system, k,integrate_mode,rigid,non_rigid
init.reset()
os.chdir('..')
```