2018

Application of Bayesian Inference to Analysis of Dynamic Light Scattering Data to Determine Particle Size Distributions

Caroline Martin
cmarti10@wellesley.edu

Application of Bayesian Inference to Analysis of Dynamic Light Scattering Data to Determine Particle Size Distributions

Caroline Martin

Submitted in Partial Fulfillment of the

Prerequisite for Honors in Physics

April 2018

Contents

List of Figures

Acknowledgments

This thesis would not exist without the never-ending encouragement, support, and hard work of my advisor, Professor Jerome Fung. I cannot express how thankful I am for all his time and effort. Thank you for all the hours you spent going over my code, driving back and forth to Harvard, and aligning lasers into single mode fibers. I cannot imagine a better thesis advisor, and I feel so lucky that I had the chance to work with you this year.

Many thanks also to the Manoharan Lab at Harvard University, especially to Professor Vinothan Manoharan for allowing a strange Wellesley student to use his DLS instrument, and for his time talking with me about my project. Huge thanks to Nabila Tanjeem, for her time debugging the instrument with me, and for the use of her gold nanoparticles. Thanks also to Tim Chiang, Victoria Hwang, and LaNell Williams, as well as the entire colloids group for being so open and welcoming.

Thanks to my professors here at Wellesley for encouraging me and challenging me, and especially to Professor Ducas, for advising me and asking me the hard questions.

Huge thanks to my friends here at Wellesley and beyond, especially to the astro group and the physics majors. I would not have made it through without you.

Finally, thank you to my family, for their support and for listening to me complain about my code even when they don't understand a word I'm saying. I love you all.

Abstract

Dynamic light scattering (DLS) is a technique used to determine the size distribution of particles suspended in a fluid, that undergo Brownian motion through random collision with the molecules of the fluid. When light is incident on the suspension, each particle scatters the light, creating a complex interference pattern that randomly fluctuates in time as the particles move. Over long periods of time, we can use these flunctuations in the measured intensity of light to determine the composition of the sample. This thesis develops and tests a Bayesian inference approach to this problem. The Bayesian approach to statistics describes probability as a degree of belief, and incorporates prior information about the system into the inference. Recently, Bayesian inference has been incorporated into other fields with limited data sets, such as astronomy, and other techniques in soft matter, such as holographic microscopy; some preliminary efforts have been made to apply this to dynamic light scattering. This thesis expands on these efforts, and develops an open source algorithm in Python that is able to efficiently and accurately describe the size distribution of particles in a nearly monodisperse suspension with a mono-modal size distribution. This algorithm is tested on several sets of DLS; while these results are promising, further testing will need to be done before a conclusion can be drawn about the precision of this method compared to the traditional least squares fit. The results of this thesis do indicate that Bayesian inference provides comparable results to the traditional least squares fit, with additional information provided about the system in the probability density function for the inferred parameters.

1 Introduction

Dynamic light scattering (DLS) is a widely used technique in biophysics and soft matter physics that uses the time-dependent nature of fluctuations in scattered light to extract information about a range of colloidal suspensions. Because DLS data is used to infer the size distribution of the particles in the system, it is well suited to the application of a Bayesian inference model. The work presented in this thesis expands on work done by Professor Vinothan Manoharan's group at Harvard University to apply Bayesian inference to soft matter systems [5], and includes both experimental and computational work. This chapter includes an introduction to DLS and a justification for the application of Bayesian inference; it also includes an outline of the thesis in its entirety.

1.1 Dynamic Light Scattering

In order to observe the kinetics of biological or soft matter systems, such as an enzyme binding to a substrate or the stability of a colloidal suspension, the technique of dynamic light scattering can be used. This method has the advantage of being noninvasive, as it does not alter the sample with each probe, and dynamic, as it is able to monitor the real time changes and motions of the particles. This technique is able to determine the size distribution of particles suspended in a fluid, freely moving in Brownian motion through random collisions with the molecules of the fluid. When an incident laser is shone on the suspension, light is scattered by those moving particles; a detector measures the intensity of the scattered light, which is the sum of scattered light from each suspended particle. Because these particles are constantly in motion, however, the pattern of light scattered is not constant. Instead, it is randomly fluctuating. Over long periods of time, we can use these fluctuations in the measured intensity of light to determine the composition of the sample.

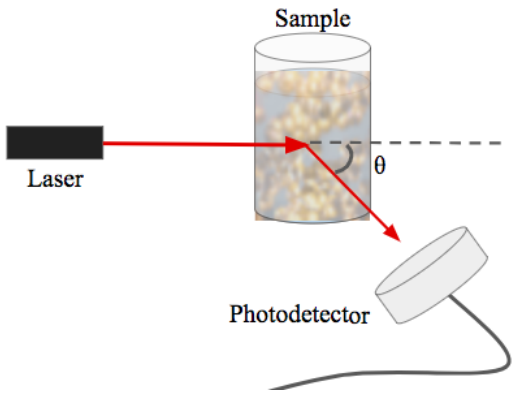


Figure 1: In dynamic light scattering experiments, incident laser light scatters off of moving particles prepared in a glass cuvette. The intensity of the scattered light is measured by a photo-detector and analyzed.

DLS makes use of scattering theory developed by Rayleigh in the late 19th century, which describes the scattering of light by particles smaller than the wavelength of the incident light being scattered; this model assumes that each particle in the suspension reradiates light as a dipole. For typical DLS experiments, the particles are generally nanometers to microns in diameter, placing them solidly in a non-quantum regime [6]. The light scattered by these particles can be used to probe the diffusive behavior of the suspended particles; this behavior is described by Brownian motion, completed by Einstein in 1910 and based on the work of Robert Brown. Einstein found that larger particles subject to random collisions with solvent particles undergo random motion that produce a mean squared displacement that is proportional to time. That motion is dependent on not only the temperature of the solution, but also the friction the particles experience. That friction is described by a Stokes drag model, where the frictional force is proportional to the viscosity of the solution and the radius of the particle.

Using these theoretical models, a connection can be made between the scattering of light and the hydrodynamic radius of the scatterers. The hydrodynamic radius can be thought of as the radius of an equivalent hard sphere that experiences the same diffusion as the particle due to Stokes drag; using the hydrodynamic radius, we can approximate the shape of irregular particles as a hard spheres. While the particles in DLS experiments are not exactly hard spheres, they can be modeled as such. With the technological developments of the laser and the digital correlator to measure the autocorrelation of the intensity of the scattered light, it became possible to quantify how the scattered light changes as a function of time, and quantify the pattern of fluctuations. [1] To find the autocorrelation function of the measured intensity, the function is shifted by increasing time intervals, called τ, and multiplied by itself. At a shift of 0, the functions exactly add, resulting in a maximum value. From there, this autocorrelation function decays, as the function starts canceling itself out. At long time, the functions become entirely uncorrelated. This produces an autocorrelation function that decays as an exponential. The steepness of that exponential decay is determined by the hydrodynamic radius of the particle. Thus, using DLS data, it is possible to determine the hydrodynamic radius of the nanoparticle suspended in the solution.

DLS has several advantages over other methods used to characterize the size of colloidal particles, including the speed of the data acquisition and the range of sizes that can be measured [6]. Additionally, because it probes a system without altering it, DLS allows for the investigation of the dynamics of a soft matter or biological system. Thus, it can be used to investigate processes of self-assembly, describe complex particle interactions, and probe the stability against aggregation of particles over time [1]. As such, dynamic light scattering is a widespread and powerful technique in soft matter physics for particles in the nanometer to micron scale.

1.2 Applying a Bayesian Inference Approach

Because the particles analyzed in DLS experiments are moving in random Brownian motion, there is an inherent probabilistic nature to the data. Inference of the size distribution from DLS data is an ill-conditioned problem, however, as small amounts of noise can generate many possible unrelated solutions, producing large amounts of error [7]. By instead applying a Bayesian inference framework to this analysis, I hope to constrain the inferred size distribution using prior knowledge and decrease this sensitivity to experimental noise.

The Bayesian approach to statistics describes probability as a degree of belief, and uses data as a way to update that degree of belief. It further constrains the models by using prior information to model parameters, which is highly applicable to model fitting in which several possible models exist. It does this using Bayes' Theorem, which is discussed in depth in Chapter 5. Bayesian statistics includes a way to formalize the information already known about the system, including its constraints and likely composition, which is one of the strengths of this inference method. The real power in Bayesian inference comes from defining the probability that the model fits the data in terms of the probability that the data fits the hypothetical model, as the latter is much easier to define. [3]

For all its power, Bayesian inference does have several drawbacks. The largest is the computational demand. Because Bayesian calculations for all but the simplest of problems involve integrations without an analytical solution, they are impossible to solve without computational methods; these methods, including Markov chain Monte Carlo algorithms (discussed in Chapter 5) have made Bayesian inference feasible in the past few decades. Another potential issue is the definition of the prior probability of the hypothesis, known as the prior. Because the inference method explicitly requires some human assumption, it is often the subject of skepticism. The effect of choosing priors is important to keep in mind, but becomes less influential when there are sufficient data (as discussed in Chapters 5 and 6).

With the rise of robust computational methods starting in the 1990s, Bayesian inference has become more and more popular in many fields, including astronomy, psychology, medicine, and even marketing [8] [9] [10] [11] [12]. Recently, Bayesian inference has been incorporated into other techniques in soft matter such as holographic microscopy [5]. Some preliminary efforts have been made to apply this to dynamic light scattering [13] [14] [15]. These initial applications of a Bayesian framework found an increase in precision compared to traditional statistical methods. This thesis seeks to expand on these efforts, and to develop an open source algorithm in Python that is able to efficiently and accurately describe the size distribution of particles in a monodisperse suspension, which currently does not exist. By incorporating a Bayesian framework into dynamic light scattering analysis, I hope to develop a more precise inference of size distribution of a sample within a limited data set.

1.3 Work Presented

This thesis explores DLS from both an experimental and analytical perspective. Chapter 2 presents a derivation of the governing equations of DLS, as well as a discussion of the fundamental physics on which it depends. All experimental work was conducted in the Manoharan Lab at Harvard University; the setup of the DLS instrument is detailed in Chapter 3. Chapter 3 also details the procedure for creating a solution with a known size distribution, and work done to investigate noise sources in the instrument. After the experimental setup is explained, Chapter 4 describes the standard data analysis approach to DLS data, including doing a nonlinear least squares fit to a single exponential across a single angle and multiple angles, and the method of cumulants. This chapter also details the application of these methods to the data collected. Once the standard methods are explored, Chapter 5 introduces the alternative statistical approach that this thesis focuses on: Bayesian Inference. Bayes' theorem is explored, and an introduction to problem solving with Bayesian inference is presented. Chapter 6 details the development of the model for data analysis, and the Python code written to apply that model. Finally, Chapter 7 details the results of that model, and compares the findings to those of the standard data analysis techniques. The Python code developed for both the standard method of analysis and the Bayesian inference method is listed in the appendices.

2 Dynamic Light Scattering Theory

The systems probed by dynamic light scattering experience Brownian motion; because the particles exist in the length scale between nanometers and micrometers, they are large enough that an in-depth consideration of quantum mechanics is superfluous, but small enough that advanced electromagnetic theory is also unnecessary. This chapter derives the governing equations of the systems probed by DLS, describes the fundamentals of light scattering, and introduces the autocorrelation function, which is the method used to analyze the light scattered by these systems. In this chapter, the scatterers are assumed to be uniform solid spheres, with small length scales in the Rayleigh scattering range, such that each scatters reradiates light as a single dipole. For these derivations, situations with a single type of scatterer with a set radius are considered first. After the equations are derived, an expansion is done to allow for a small spread within the size distribution of the particles; this method of cumulants is detailed in the final section.

2.1 Brownian Motion

Because DLS experiments are conducted on soft matter systems of particles in nanometer to micron scale suspended in fluids, we can consider the systems as a collection of macromolecules that are colliding with the much smaller solvent molecules. The random motion of a small but macroscopic particle suspended in a fluid is described by Brownian motion. Also called diffusion, Brownian motion results from random collisions between the atoms or molecules of the fluid and the suspended particles. These collisions are fundamentally the result of thermal energy, which causes the molecules to vibrate and move. Because Brownian motion is fundamentally random, it is impossible to predict the exact position of any given particle after a set amount of time. In general, however, the net displacement can broadly be quantified by the mean squared displacement of the particle.

2.1.1 Mean Squared Displacement

At its simplest, this mean squared displacement can be derived by considering a one-dimensional particle subject both to an external field and internal interactions with the particles of the fluid. Because of the constant motion of these fluid particles, those internal interactions are highly complex and difficult to describe analytically. Instead, we can consider the average forces on the particle by breaking them up into internal and external interactions

$$m\frac{dv}{dt} = \mathcal{F}(t) + F(t) \tag{1}$$

where $\mathcal{F}$ is the total external forces on the particles from an external field (be it magnetic or gravitational), and F is the total internal interaction between the particle and all the other particles in the solution [16]. This statement of Newton's Second law in Equation 1 appears simple enough, but it is obviously nearly impossible to write down $\mathcal{F}$ and F; we instead must consider another way to describe the broad trends of F. While it is impossible to write down the exact forces on a particle at any given instant, over time the force on the particle can be broken up into two forces

$$F = \overline{F} + F' \tag{2}$$

Here, $\overline{F}$ represents a slowly varying force, which we can think of as the drag that a particle experiences. This friction term has the tendency to reduce the motion of the particles to zero over time. That drag force on the particle can be written as

$$\overline{F} = -\alpha v \tag{3}$$

assuming that this scatterer exists in a low Reynolds number regime, and thus experiences a Stokes drag force that is dependent only on some constant α and the velocity of the scatterer. The other term in the internal forces, F', can be thought of as the rapidly changing force from spontaneous collisions. To make sense of these forces, we must consider them over a long amount of time. Over a long time interval, the particle experiences collisions from all sides; thus, the average of F' must be zero.

When we set the net sum of these forces equal to the change in momentum of the particle by Newton's second law, we can write what is called the Langevin equation,

$$m\frac{dv}{dt} = \mathcal{F} - \alpha v + F' \tag{4}$$

where v is the velocity of the particle, and α is the Stokes drag constant.

If we assume that no external force is applied, then the motion of the particle suspended in the fluid is determined solely by its interaction with that fluid. Although these particles do exist in the external field of gravity, they are sized such that the thermal energy of the particles greatly overpowers the potential difference due to the height of the small cuvette. Thus, we can consider for our derivation that the external forces on the sample are zero.

Since the mean displacement $\langle x \rangle$ approaches 0 when the system is in thermal equilibrium, as the random motion has no preferred direction, we instead can calculate the average of the square of the displacement.

To do so, we multiply both sides of the Langevin equation by x to get

$$mx\frac{dv}{dt} = -\alpha x\frac{dv}{dt} + xF' \tag{5}$$

By rewriting these derivatives, we can write this as

$$m\frac{d}{dt}(x\dot{x}) - m\dot{x}^2 = -\alpha x\dot{x} + xF' \tag{6}$$

To find the mean square displacement, we must take the average. Recall that the average of the small fluctuations of F' is equal to 0, as these fluctuations have no preferred direction. Thus, we can simplify to

$$m\frac{d}{dt}\langle x\dot{x}\rangle = m\langle\dot{x}^2\rangle - \alpha\langle x\dot{x}\rangle \tag{7}$$

Using the equipartition theorem, which states that $m\langle\dot{x}^2\rangle = kT$, we can relate the kinetic energy of the particle to its thermal energy, such that the expression becomes the simple differential equation

$$\frac{d}{dt}\langle x\dot{x}\rangle = \frac{kT}{m} - \frac{\alpha}{m}\langle x\dot{x}\rangle \tag{8}$$

Solving and plugging in initial conditions that the particle begins at the origin, we get

$$\langle x\dot{x}\rangle = \frac{kT}{\alpha}(1 - e^{-\frac{\alpha}{m}t}) \tag{9}$$

Although we now have an equation for $\langle x\dot{x}\rangle$, we can rearrange to get an expression for the mean squared displacement $\langle x^2\rangle$, which is the typical measurement of Brownian motion. By rearranging this derivative, and using the fact that the time derivative and time average operators commute (and thus can be interchanged), we can set

$$\langle x\dot{x}\rangle = \frac{1}{2}\frac{d}{dt}\langle x^2\rangle \tag{10}$$

Finally, integrating away the derivative, and rearranging to arrive at $\langle x^2\rangle$, the expression for the mean square displacement of a particle undergoing Brownian motion becomes

$$\langle x^2\rangle = \frac{2kT}{\alpha}(t - \frac{m}{\alpha} - \frac{m}{\alpha}e^{-\frac{mt}{\alpha}}) \tag{11}$$

If we consider the particles as hard spheres in a fluid, the friction coefficient α is given by Stokes' equation,

as the particles are in a regime of a small Reynolds number [1]. The drag coefficient for a hard sphere is

$$\alpha = 6\pi\eta R \tag{12}$$

where the particle is approximated as a hard sphere with a hydrodynamic radius R and suspended in a fluid with viscosity η [6].

At longer times, the limiting behavior of this becomes the more recognizable form of mean squared displacement for a diffusive random walk, where the friction constant is given as above,

$$\langle x^2 \rangle = \frac{kT}{3\pi\eta a} t = 2Dt \tag{13}$$

where D is defined by this equation to be the diffusion coefficient. To expand this to three dimensions, the expression for the mean square displacement $\langle |\Delta \vec{r}^2| \rangle$ would increase by a factor of three to be

$$\langle |\Delta \vec{r}^2| \rangle = 3\frac{kT}{3\pi\eta a} t = 6Dt \tag{14}$$

2.2 Light Scattering and Autocorrelation Functions

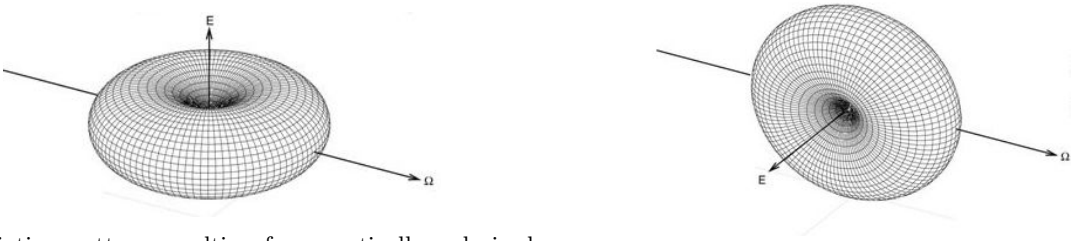
In order to extract information about the makeup of these systems undergoing Brownian motion, it is possible to observe how an incoming electromagnetic wave is scattered by the suspended particles. This incoming light is here assumed to be a plane wave, and complex electromagnetic theory is not considered.

2.2.1 Rayleigh Scattering

When the particles observed in DLS measurements are smaller than the wavelength of the incident laser, the scattered light can be described by Rayleigh scattering [1]. When an electromagnetic wave is incident on a particle, the oscillating electric field induces an oscillation of the electrons in the particle. As these electron accelerate, the accelerating charges emit electromagnetic waves in all directions. These particles can be assumed to act as individual electric dipoles, reradiating light as their charges oscillate. This scattered light, the sum of all the re-radiated light of all the particles within the fluid, fluctuates with time as the waves constructively and destructively interfere. This fluctuation pattern has measurable properties that can reveal the size, shape, and interactions of the molecules within the fluid.

The light reradiated by the induced dipoles depends on the polarization of the incident light, as seen in Figure 2. If the incident light is horizontally polarized, the horizontally oscillating electric field will induce horizontal oscillations in the dipole, marked with an E in the diagram. This creates a strong angle dependence

in the horizontal plane, with a sharp decrease in scattering intensity at 90 degrees. Vertically polarized light, on the other hand, will induce vertical oscillations in the dipole. The light will be reradiated without angle dependence in the horizontal plane. Because of the lack of angle dependence in the plane measured by the DLS instrument, it is useful for the incident light to be vertically polarized in DLS experiments.



(a) Radiation pattern resulting from vertically polarized light

(b) Radiation pattern resulting from horizontally polarized light

Figure 2: The resulting direction of reradiated light emitted by a dipole, where the incident plane wave is moving in direction Ω and the induced dipole is oriented in direction E. When light is incident on a dipole, the direction of the induced dipole is dependent on the initial polarization of the light. For incident horizontally polarized light, there is a drop in the intensity of scattered light at 90 degrees; vertically polarized light does not have this angular dependence in the horizontal plane. [1]

Broadly, how rapidly the intensity pattern fluctuates is an indication of the size of the particles within the fluid. For particles of small radius, the fluctuation pattern changes much more rapidly; this is because the smaller particles experience less drag, and thus change positions more rapidly. For larger particles, the large amount of drag slows down the particles, producing a fluctuation pattern that changes slower. The resulting pattern can schematically be seen in Figure 3.

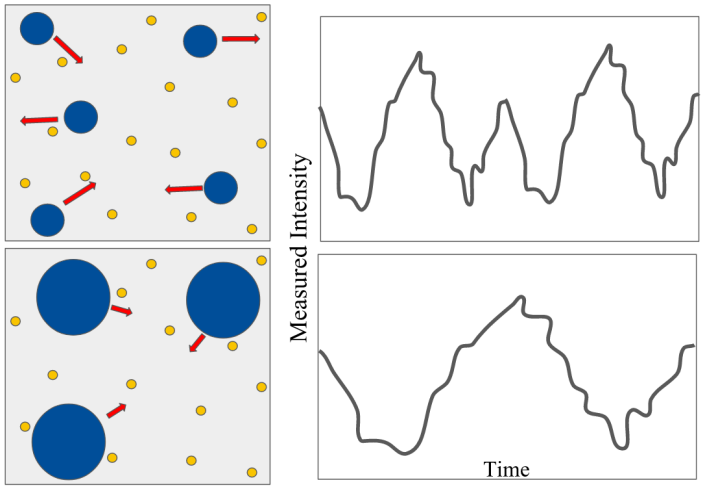


Figure 3: In DLS experiments, light scatters off of moving particles (in blue). The size of these particles determines their motion, which in turn affects the fluctuation of the measured intensity.

2.2.2 Autocorrelation Function

To draw information out of this fluctuating signal, the autocorrelation function of the intensity is examined. A correlation function describes the degree to which two properties are correlated. An autocorrelation function of a property, then, is how correlated that property is to itself at different times [7]. The autocorrelation function of a function $f(t)$ can be written as

$$\langle f(t)f(\tau)\rangle = \lim_{T\to\infty}\frac{1}{T}\int_0^T dt f(t)f(t+\tau) \tag{15}$$

where τ is the amount by which the function has been shifted. This autocorrelation function of the intensity depends on a wide range of τ.

For example, the autocorrelation function of a periodic square wave, as shown in Figure 4, is a triangular wave. When the square waves are at a shift of $\tau = 0$, they completely add, resulting in a maximum value of the autocorrelation function. When they are shifted by an amount equal to the width of the wave, they completely cancel, resulting in a minimum value of the autocorrelation function. Here, this function is periodic, leading to a periodic autocorrelation function, and linear, leading to a triangular shape between these minimum and maximum points.

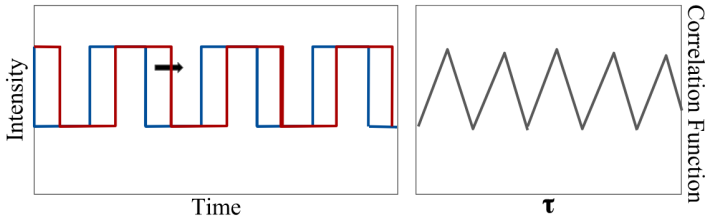


Figure 4: A square wave interferes constructively with itself until the squares are completely in phase; it then interferes destructively until they are completely out of phase. This leads to a periodic autocorrelation function that oscillates between complete addition and complete subtraction: the triangle wave shown above.

The autocorrelation function of the intensity of scattered light, on the other hand, is not periodic. When τ is 0, the autocorrelation function is simply

$$\langle f(0)f(\tau)\rangle = \lim_{T\to\infty}\frac{1}{T}\int_0^T dt f(t)f(t) = \langle f^2\rangle \tag{16}$$

At $\tau = 0$, the autocorrelation function is at a maximum, as the random fluctuations of the two identical functions do not cancel, and are completely additive. When τ is small, the intensity function measured is still highly correlated, as the particles have not yet shifted much from their initial positions. The autocorrelation function must be smaller at $\tau > 0$ than at $\tau = 0$, however, as some parts of the random fluctuations begin

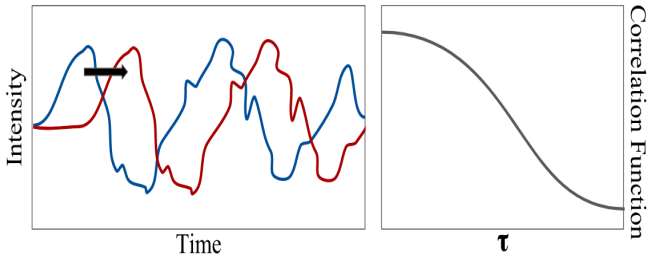


Figure 5: To find the decaying autocorrelation function of the intensity, the function is shifted by greater τ until it reaches a minimum value. The resulting autocorrelation function is a decaying exponential, shown here schematically on a semilogarithmic plot.

to cancel as a peak begins to align with a trough. As τ increases, the autocorrelation function decays as the intensity becomes less and less correlated. At long τ, the autocorrelation function decays to

$$\lim_{\tau \to \infty} \langle f(0) f(\tau) \rangle = \langle f(0) \rangle \langle f(\tau) \rangle = \langle f \rangle^2 \tag{17}$$

This decay is schematically represented in Figure 5.

Using this minimum value for the autocorrelation function, we can normalize the autocorrelation function of the electric field of the scattering light, written as

$$g^{(1)}(\tau) = \frac{\langle E(t) E(t+\tau) \rangle}{\langle E(t) \rangle^2} \tag{18}$$

This autocorrelation function decays exponentially to the minimum point, with some decay constant given by

$$g^{(1)}(\tau) = A e^{-\Gamma \tau} \tag{19}$$

where A is the amplitude of the autocorrelation function and Γ is a decay constant [7].

While it is theoretically simpler to consider the autocorrelation of the electric fields, experimentally this is not an easily measurable value. Instead, the autocorrelation function of the intensity is considered. The autocorrelation of the intensity $g^{(2)}(\tau)$ can be related to the autocorrelation of the electric field $g^{(1)}(\tau)$ through the Siegert relationship [17]

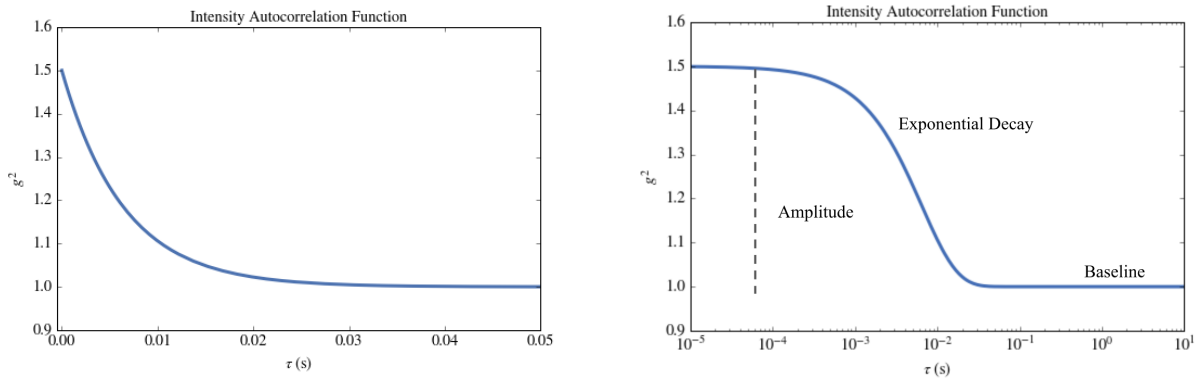
$$g^{(1)}(\tau) = (g^2(\tau) - 1)^{0.5} \tag{20}$$

For the simplest solution in DLS experiments, of a single type of spherical particle of uniform size, the

autocorrelation function of the electric field is given by

$$g^{(2)}(\tau) = Ae^{-2\Gamma\tau} + B = Ae^{-2Dq^2\tau} + B \tag{21}$$

where A is the amplitude, D is the diffusion coefficient, q is the scattering vector, and B is the baseline.



(a) The autocorrelation function of intensity plotted on a linear plot

(b) The autocorrelation function of intensity plotted on semilogarithmic plot

Figure 6: The autocorrelation of the intensity function given in Equation 21 is plotted above, on both a linear scale and a semilogarithmic scale. The semi-log scale is advantageous to see detail over the wide time scales probed by typical DLS experiments. The height of the decay is given by A, the baseline is B, and the steepness of the decay is determined by Γ.

Of these parameters that determine the behavior of this autocorrelation function, B is usually set to be 1, as given in the Siegert relationship, and D is dependent on the Brownian motion, which is considered above [6]. Using the expression derived in Equation 14, the Stokes-Einstein relationship, the diffusion coefficient can be written as

$$D = \frac{kT}{6\pi\eta R} \tag{22}$$

where k is Boltzmann's constant, T is the temperature of the solution, and η is the viscosity of the solution.

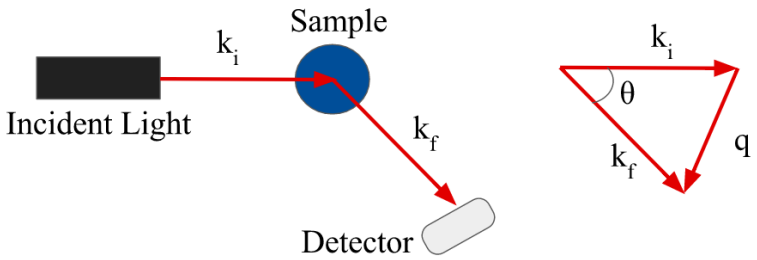


Figure 7: The incident laser light scatters off the sample, and is detected at a known angle. The scattering angle q for the given angle θ is determined by the wave numbers k_i and k_f.

This leaves the scattering vector $\vec{q}$, which represents the difference between the initial vector of incident light $\vec{k_i}$ and the final vector of the scattered light $\vec{k_f}$, such that

$$\vec{q} = \vec{k_f} - \vec{k_i} \tag{23}$$

as seen in Figure 7. The magnitude of $\vec{k}$ is the wave number of the incident and scattered light, such that

$$|\vec{k_i}| = \frac{2\pi n}{\lambda_i} \tag{24}$$

Since the magnitude of the initial vector $\vec{k_i}$ is about equal to the magnitude of the final vector $\vec{k_f}$, as the change in the wavelength of the light before and after scattering is negligible, given that the scattering off the particle is elastic, this vector subtraction forms an isosceles triangle, with the length of $\vec{k_i}$ and $\vec{k_f}$ being the same. Using the law of cosines, the relationship can be written as

$$q^2 = k_i^2 + k_f^2 - 2k_f k_i \cos(\theta) \tag{25}$$

where θ is the angle between $\vec{k_f}$ and $\vec{k_i}$, and k_i and k_f are the magnitudes of the vectors. Since $k_f = k_i$, as we are assuming elastic scattering, this becomes

$$q^2 = 2k^2(1 - \cos(\theta)) \tag{26}$$

Through trigonometric identities, this can be rewritten as

$$q = 2k \sin\left(\frac{\theta}{2}\right) = \frac{4\pi n}{\lambda} \sin\left(\frac{\theta}{2}\right) \tag{27}$$

With these expressions for the values in the autocorrelation function $g^{(2)}(\tau)$, it becomes possible to extract the hydrodynamic radius R of the particle in a monodisperse solution of spherical particles by observing the intensity of the scattered light and knowing the parameters of the experiment, including the temperature, viscosity of the fluid, angle of measurement, wavelength of the incident light, and index of refraction of the fluid.

In general, the time constant for the exponential decay can be given by

$$\frac{1}{\Gamma} = \frac{1}{2Dq^2} \tag{28}$$

For a particle in the size regime probed by DLS experiments, assuming room temperature and water based solutions, this time constant usually ranges from 1 to 100 ms. In order to sample across both very small and very large τ in comparison to this time constant, the intensity of the scattered light is observed for between 0.5 and 10 minutes.

2.3 Method of Cumulants

When the suspended particles are no longer completely monodisperse, it is not appropriate to fit a single exponential to the decay of the intensity autocorrelation function. If the particles have a finite spread of radii centered around some mean and that spread can be considered Gaussian, then it is possible to expand the single exponential fit into a sum of exponential terms. These terms describe a distribution of decay rates Γ, and the moments of this distribution [18].

In general, the autocorrelation function of the electric field of a polydisperse sample can be written as

$$g^{(1)}(\tau) = \int G(\Gamma) \exp(-\Gamma\tau) d\Gamma \tag{29}$$

which is an integral sum of decay curves, where $G(\Gamma)$ is a normalized function that describes this distribution of the decay rates. Unfortunately, it is difficult to measure $G(\Gamma)$, as small noise in the data leads to huge uncertainties in recovered values of $G(\Gamma)$ [19]. It is possible to simplify this probability function of decay constants if we make the assumption that distribution is peaked around some value $\overline{\Gamma}$, an average decay constant. We can then rewrite the value of the exponent in terms of that average decay constant as

$$g^{(1)}(\tau) = \int G(\Gamma) \exp\big(-\overline{\Gamma}\tau\big) \exp\big(-(\Gamma - \overline{\Gamma})\tau\big) d\Gamma \tag{30}$$

Because the distribution is assumed to be sharply peaked around this mean decay constant, the value $\Gamma - \overline{\Gamma}$

is small. Using this assumption that the value in the exponential is much less than one, we can expand the above equation using a power series expansion,

$$g^{(1)}(\tau) = \int \exp\big(-\overline{\Gamma}\tau\big) G(\Gamma)[1 - (\Gamma - \overline{\Gamma})\tau + \frac{(\Gamma - \overline{\Gamma})^2}{2!}\tau^2 - \frac{(\Gamma - \overline{\Gamma})^3}{3!}\tau^3 + ...]d\Gamma \tag{31}$$

We can write this more generally by defining what is known as a moment about the mean,

$$\mu_m = \int G(\Gamma)(\Gamma - \overline{\Gamma})^m d\Gamma \tag{32}$$

Using this definition and integrating, the autocorrelation function of the electric field becomes

$$g^{(1)}(\tau) = \exp\big\{-\overline{\Gamma}\tau\big\}(1 + \frac{\mu_2}{2!}\tau^2 - \frac{\mu_3}{3!}\tau^3...) \tag{33}$$

Finally, recalling the Siegert relation between $g^{(1)}$ and $g^{(2)}$ [Equation 20], we can relate this to the autocorrelation function of the intensity to get

$$g^{(2)}(\tau) = B + \beta \exp\big(-2\overline{\Gamma}\tau\big)(1 + \frac{\mu_2}{2!}\tau^2 - \frac{\mu_3}{3!}\tau^3...)^2 \tag{34}$$

If $\overline{\Gamma}$ is determined by a fit of Equation 34, the value for $\overline{\Gamma}$ can be used to find the average hydrodynamic radius in a similar method as the single exponential, where

$$\overline{\Gamma} = Dq^2 = \frac{kT}{6\pi\eta R}(\frac{4\pi n}{\lambda}\sin\left(\frac{\theta}{2}\right))^2 \tag{35}$$

which can be used to solve for the hydrodynamic radius R, given the known experimental parameters.

This cumulant expansion is generally only expanded out to the 2nd order cumulant, which describes the polydispersity index, or width of the distribution. The 3rd order cumulant can be included to give information on the skew of the solution, but is often unnecessary or uninformative, as the skew for manufactured particles often is 0. [20] In general, the 2nd order expansion is sufficient to describe a polydisperse solution with a small Gaussian spread about some central mean size, where the 2nd order cumulant is related to the width of the Gaussian as

$$\sigma^2 = \frac{\mu_2}{\overline{\Gamma}^2} \tag{36}$$

where σ is the half-width of the Gaussian curve. When this value for the polydispersity index exceeds the order of 0.6, the sample becomes too polydisperse for the assumptions of the expansion to hold [19].

When the suspension is made up of a spread of particles with multiple peaks, where $G(\Gamma)$ is not mono-

modal, both the single exponential method and the method of cumulants fail. The standard method for these populations is a constrained regularization algorithm called CONTIN, which discards solutions that it considers unlikely [21]. Inference of the size distribution from the autocorrelation function is an ill-conditioned problem, however, as small amounts of noise can generate many possible unrelated solutions, producing large amounts of error. Because of these complications, this algorithm is beyond the scope of this project; it is possible, however, that the same principles could be applied later to this type of fit.

3 Experimental Setup

In addition to the development of a Bayesian framework for the inference of size distribution of particles from DLS data, a substantial effort in this project was experimental work to take data to be analyzed. While some data analyzed in subsequent chapters were taken elsewhere, the majority of the data were taken in the Manoharan Lab, after work with alignment and optimization of the instrument. This chapter presents an overview of the design of the DLS instrument used, the process for alignment and data collection, and the process for optimizing the signal from the scatterers. The data collected will be analyzed in subsequent chapters.

3.1 Design

The DLS instrument used for the data analyzed in this thesis is an ST100 Variable Angle Light Scattering Instrument from Scitech, with a minimum scattering angle of 5 degrees, located in the Manoharan Lab in Harvard University. Broadly, the DLS instrument uses a 658 nm diode laser steered into a sample submerged in an oil bath. The scattered light is measured at an adjustable angle by an optical fiber connected to a photodetector that is rotated around the sample. This detector is connected to a digital autocorrelator which outputs the autocorrelation function of the intensity through software. The output is the autocorrelation function rather than the direct intensity measurements because of the sheer amount of data gathered. The instrument measures the intensity roughly every 10^{-7} seconds; directly outputting the intensity for a 10 minute measurement would then be 6 billion points of data, which is computationally very expensive.

In order for the instrument to measure the fluctuating intensity of the scattered light, there must be enough scattered light for the photodetector to measure. To create the incident plane waves on the scatterers, a laser must be carefully steered into a single-mode fiber that is connected to the instrument. The optical setup used to achieve the necessary alignment is shown in Figure 8. First, light is emitted from a red diode laser and collimated by a lens; the diode has the advantage of having its power controlled by adjustable

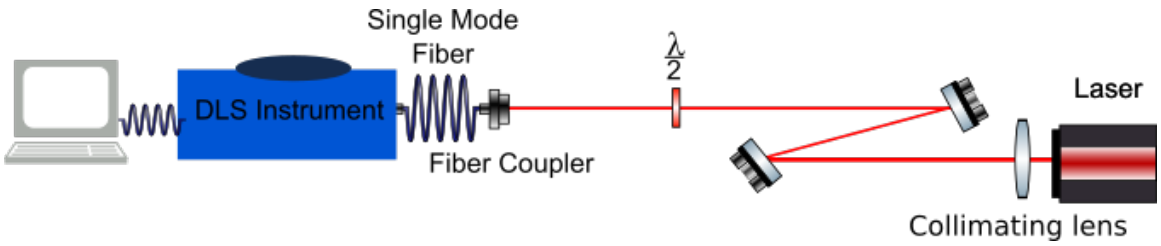


Figure 8: 658 nm light is steered into a fiber coupler using two steering mirrors. The coupler transmits light into a single-mode optical fiber, which is connected to the DLS instrument. The autocorrelation function of the signal is output through software on a PC.

current input, rather than sets of attenuating filters. The beam is then directed by two steering mirrors through an adjustable half-wave plate into a fiber coupler. The fiber coupler, from Oz Optics, couples the beam to a single-mode fiber, and is adjustable through fine tilt controls. An internal aspheric gradient refractive index lens in the fiber coupler allows for the necessary adjustable focus. Once the beam is perfectly aligned, the single-mode fiber produces a Gaussian beam that approximates as plane waves near the center. These plane waves are directed through the sample; the scattered light is then measured at adjustable angles by an avalanche photodetector. Finally, the measured intensity of the light is run through a correlator interior to the DLS instrument. The output of that correlator, the intensity autocorrelation function $g^{(2)}$, is read through software on a PC.

Internal to the DLS instrument is a circulating bath of Dow Syltherm XLT oil. It is a low weight blend of polymethyl siloxanes with a kinematic viscosity of about 2 cSt [22]. This oil bath is essential for two reasons. First, it maintains the temperature of the sample. Because temperature is a variable in the equation for the hydrodynamic radius for a sample suspended in water and because the viscosity of water is highly temperature-dependent, it is necessary to precisely maintain the temperature of the scattering sample. This instrument does so by submerging the sample in the circulating oil bath that is connected to temperature controls. Second, the oil minimizes reflection off the glass cuvette containing the sample, because the index of refraction of the oil is comparable to the index of the glass. This prevents a large amount of the laser light from being lost to reflection. While the oil is essential to the instrument, it must be filtered and kept clean; otherwise, it is possible that scattering could occur off particles suspended in that oil, which could interfere with the signal from the sample.

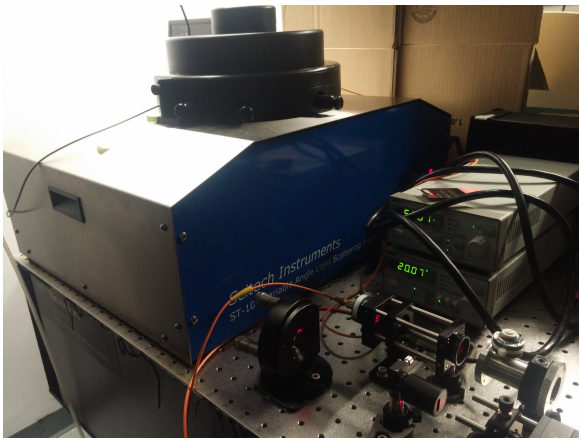


Figure 9: The DLS instrument used for data collection, made by Scitech, model ST100. The sample is submerged in oil within instrument, and light is scattered and measured internally.

When a sample is measured, the black cap seen in Figure 9 is removed and the sample is submerged in the oil bath. The sample is held in place by a clamping ring and collet, with at least a few millimeters of the

sample extending past this clamp and into the oil bath. After the sample is allowed to come to equilibrium and its temperature is stable, data can be taken. The laser is directed from the incoming fiber through the oil and sample. Internally, there is an attenuator that allows the software to adjust the laser power. The scattered light is then captured by a rotated optical fiber that directs the light into a photon counting photomultiplier tube. [22] This count rate is analyzed by an internal digital correlator, which outputs the the autocorrelation function of the intensity, which can then be analyzed as described in Chapters 4 and 6.

The rotation of the scattered light fiber is controlled by software connected to the instrument that sets both the angle and the observation time. The duration of observation is determined both by the general time constant of the decay for the sample, as well as the count rate measured by the photomultiplier. If the count rate is low, it is necessary to increase the observation time. The photon count uncertainty is proportional to the square root of the number of counts.

3.2 Laser Alignment

Because the DLS instrument requires a single-mode fiber, the difficulty in the optical set-up comes from the precision required by that fiber. In order for the beam to be transmitted, it must couple to the fiber. If the beam enters the fiber at too great an angle, it will not be transmitted. This precision is achieved by adjusting several degrees of freedom: the two tilt screws on each of the two steering mirrors, the three tilt screws on the fiber coupler, and the z adjustment in the end of the single mode fiber that adjusts the distance between the end of the fiber and the coupler lens. Through iterations of of these adjustments, the necessary precision can be achieved. It is useful, however, to align in intermediary steps, outlined below.

The laser is first roughly aligned using the steering mirrors and a series of irises to establish a constant height. Once the beam is roughly directly straight into the fiber coupler, it is necessary to align through a multi-mode fiber before aligning to a single mode fiber. Multi-mode fibers transmit more than a single transverse mode of the electromagnetic wave in the incoming beam, and because of this can transmit incident light even when they are at an angle with respect to the fiber. Figure 10 compares the resulting beam shape of single and multi-mode fibers. Because a single-mode fiber transmits only one mode, it results in a Gaussian beam profile. This Gaussian profile is essential, as the assumption that the sample is undergoing Rayleigh scattering is only valid if the incoming electromagnetic wave is effectively a plane wave. This approximation holds near the center of a Gaussian beam, but fails when the transmitted wave has multiple higher order modes.

To optimize the alignment through a multi-mode fiber, it is necessary to observe the resulting beam profile at the other end of the fiber. To do so, the light from the beam is projected on a screen a fixed,

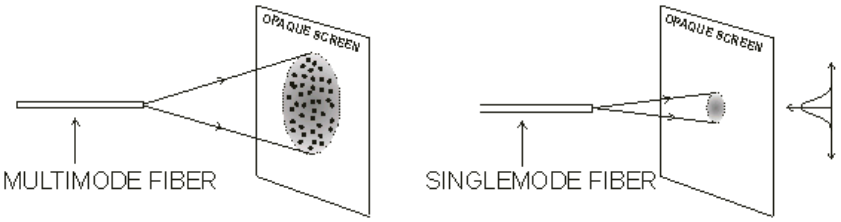


Figure 10: A multi-mode fiber allows for many electromagnetic modes to be transmitted, while a single-mode fiber only allows for one; this produces a Gaussian beam profile. [2] The Gaussian profile can be approximated as a plane waves near the center of the beam, which is a necessary assumption in the light scattering theory.

large distance away; the large distance allows the diverging beam to appear wide enough to the eye to see the features of the beam shape. Because the multi-mode fiber is being used as an intermediate step towards the single-mode fiber, the goal of the alignment is to minimize the diameter of the beam and maximize the size of the speckles. This indicates that the lower order modes are being excited, as shown in Figure 11. This alignment is not the same as maximizing the power of the out-coming signal; it is possible to maximize power through exciting higher-order modes, which would disappear when the single-mode fiber is connected. [2]

After the beam profile through a multi-mode fiber has been optimized, a single-mode fiber can be connected to the fiber coupler. Because the end of the DLS instrument's single-mode fiber is inaccessible, is is helpful to use an additional single-mode fiber for alignment purposes. When the beam is misaligned by even a small amount, the output power is too low to see using the DLS software, due to the amount of scattering through the oil of the instrument. By using a separate single-mode fiber, the output power can be monitored directly using a power meter.

Small adjustments to the tilt screws on the back of the coupler change the angle between the fiber and the beam, as shown in Figure 12. Because the single-mode fiber requires the incident beam to have a very

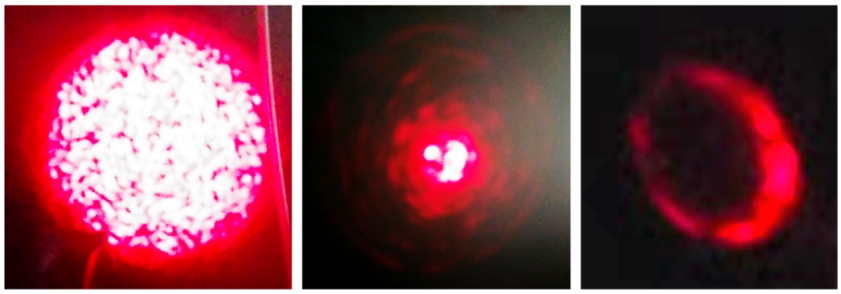


Figure 11: The beam profile out of a multi-mode fiber is optimized (left). When the beam is not optimized, only certain modes of the fiber are excited. The center beam shows higher order modes slightly excited, with a wide beam profile. The right beam shows a beam profile with only higher order modes excited.

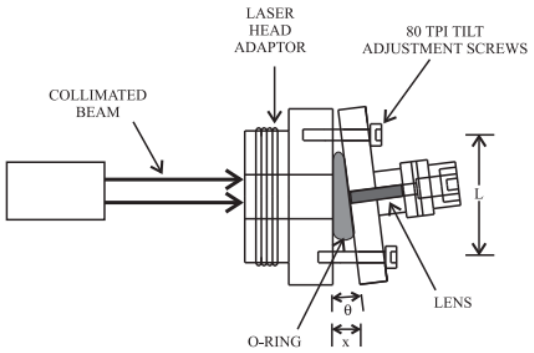


Figure 12: The fiber coupler is controller through a set of three adjustable tilt screws. This allows for the beam to be precisely aligned to the single-mode fiber, allowing the beam to be transmitted. The lens internal to the coupler is a non-spherical gradient refractive index lens. [2]

small angle with the fiber, many small adjustments had to be made to the three tilting screws and the steering mirrors before a sizable signal was seen in the power meter. Once the signal was maximized, the loose single-mode fiber was replaced with the single-mode fiber attached to the DLS instrument without losing the previously achieved signal.

The final alignment step was adjusting the focus, which is controlled by a tunable length adjustment on the end of the single-mode fiber. By rotating a small ring around the end of the fiber, the end of the fiber is shifted either backward or forward, changing the distance that the ferrule will extend past the connector. This changes the spacing between the fiber and lens inside the fiber coupler very precisely. By adjusting this ring, it is possible to shift the end of the fiber to the exact focal point of the lens, which increases the coupling efficiency. After the output power from the single-mode fiber is optimized, it is possible to make more small adjustments to the tilt screws and steering mirrors to finalize the alignment. Through these adjustments, the laser was aligned, resulting in a signal large enough to be read by the DLS instrument. The typical coupling efficiency for the incident light is expected to be between 30% and 60%. Typically, the transmitted count rate must be larger than 10 kHz as read by the DLS software, with the scattered count rate at 90 degrees between 5 and 60 kHz. By the end of the alignment process, before adjusting for the polarization, we achieved a count rate of 40 kHz transmitted with no sample in the instrument to scatter. [22]

3.2.1 Polarization Adjustment

After the laser has been aligned, it is necessary to adjust the polarization by rotating the half wave plate, shown in Figure 8; the importance of the vertical polarization is discussed in Chapter 2. To achieve this polarization, a polarizer was inserted into the DLS instrument, in the space where a sample is placed for scattering. The incident beam will strike the polarizer at a normal incidence; by monitoring how much

light is transmitted, it is possible to adjust the polarization of the beam. The polarizer inserted into the instrument blocks vertically polarized light, so the polarization is optimized when the transmitted power is minimized. [22]. To minimize the transmitted beam power, the adjustable half-wave plate is loosened and rotated by adjusting a lever. This in turn rotates the polarization of the light relative to the polarizer in the instrument, as linearly polarized light incident on a half-wave plate at an angle θ to its fast axis will be transmitted rotated by an angle of 2θ. After this adjustment was made to produce vertically polarized light entering the fiber, the fiber maintains the polarization as it is transmitted to the sample. The adjustment did result in a drop in the count rate to around 18 kHz transmitted, which was still high enough to produce reliable measurements.

3.3 Sample Preparation

3.3.1 Cuvette Cleaning

Because the scatterers in the DLS samples are typically in the nanometer range, dust presents a huge potential source of noise. If a sample contains dust, which has a wide range of sizes but is typically in the micron scale, the scattering from the dust can overwhelm the scattering from the nanometer scale particles. To minimize dust, it is necessary to thoroughly clean the cuvettes that contain the sample. The process used for cuvette cleaning is listed below:

1. Submerge the cuvette and cap in a 1% concentration of Micro 90 soap and water solution, and cover with tin foil. Bake in oven at 65 degrees Celsius for 40 minutes.

2. Rinse thoroughly with tap water at least 10 times. The salts in the unfiltered water provide additional cleaning properties to help rinse the soap from the glass.

3. Rinse thoroughly with deionized water at least 10 times.

4. Rinse thoroughly with Millipore water at least 10 times.

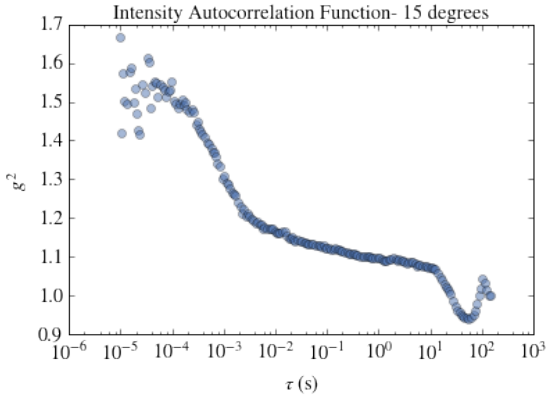
5. Dry cuvettes upside down on a sheet of clean tinfoil, covered with glass container in oven at 65 degrees Celsius for 2-3 hours, or until dry.

If the cuvettes are properly rinsed, there should be no residue from the Micro 90 remaining on the glass. An additional step to minimize dust in the samples is to filter the fluid through a 0.1 micron filter as it is being transferred into the cleaned cuvette.

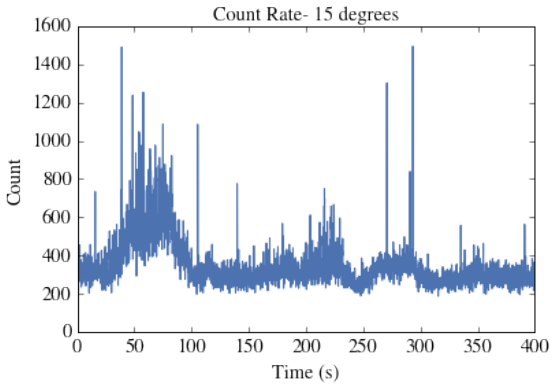
3.3.2 Calibration with Small Scatterers

In order to get reliable data to compare methods of analysis, it was necessary to create a sample of known, monodisperse size. First, a sample of $C_{12}E_8$, or octaethylene glycol monododecyl ether, was prepared. $C_{12}E_8$ forms micelles of a known hydrodynamic radius of 3 nm, making it a good surfactant for calibration measurements. Because its radius is in the nanometer scale, its scattering is much stronger than water, which should theoretically produce a strong enough signal to detect. A solution of the chemical is also stable around room temperature, and has an index of refraction very close to the index refraction of water. Further, the solution is stable at long times, and will not aggregate [23] . 3 mL of a solution of $C_{12}E_8$ and water were transferred via a pipette and a syringe filter with a 0.1 micrometer pore diameter into the cleaned cuvette to remove any dust in the solution.

The analysis of this sample can be seen in Figure 13. Unfortunately, both the count rate detected by the photomultiplier and the autocorrelation function indicated a possible source of contamination within the sample or the instrument. For an uncontaminated monodisperse sample, the count rate of the scattered light should be roughly constant, without large spikes or fluctuations. If these fluctuations exist, it indicates that a larger scatterer has moved into the scattering region, and overwhelmed the signal from the sample. Because of the large spikes in the count rate in Figure 13 (b), this sample was considered contaminated. These spikes were largest in the forward scattering range (small angles), which indicated that the contaminants were larger scatterers, most likely dust. This can also be seen in the double decay of the autocorrelation function measured, in Figure 13 (a). The decay at larger times τ was produced by a scatterer much larger than the 3 nm micelles.



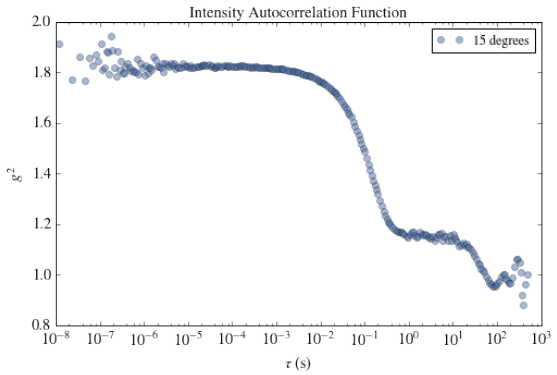
(a) Autocorrelation function of $C_{12}E_8$ sample



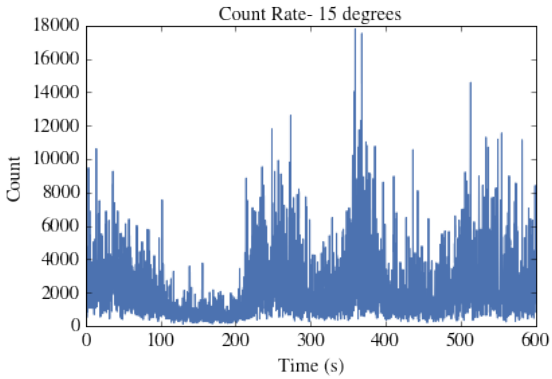
(b) Count rate of sample found by photon detector

Figure 13: The autocorrelation function of the intensity of light (a) scattered by a solution of $C_{12}E_8$ with a diameter of 3 nm, along with the count rate recorded by the photodetector (b). The signal is overwhelmed by the forward scattering of larger particles, likely dust suspended in the instrument. The spikes in the count rate indicate this contamination, as does the additional decay at larger τ, around 10^1 seconds, corresponding to a larger radius.

To combat this additional signal seen in the $C_{12}E_8$ sample, the first effort was to increase the hydrodynamic radius of the scatterers to 10 nm. Before the sample was prepared, the process for cleaning was repeated on new cuvettes in order to minimize the risk of contamination. This sample was made of gold nanoparticles suspended in water, measured at 23 degrees Celsius for 10 minutes. Although this sample did have a much clearer autocorrelation decay at the expected decay time, as seen in Figure 14 (a), the secondary decay at high times was still present. The count rate, as seen in Figure 14 (b), also displayed clear signs of sample contamination, with large spikes and fluctuations characteristic of secondary large scatterers. Additionally, this sample also had similar angular dependence, with small angles showing a much larger count rate; this again was most likely the result of large scatterers, which preferentially scatter in the forward direction.



(a) Autocorrelation function of contaminated sample



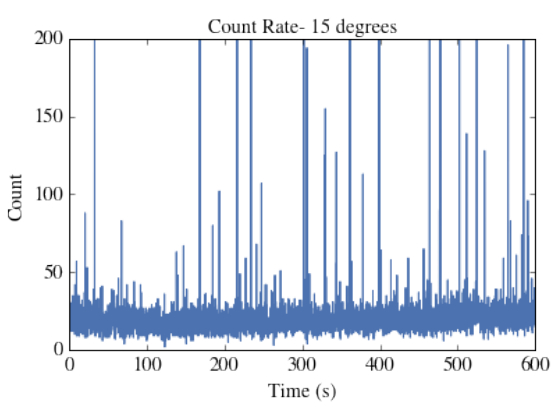
(b) Count rate of sample found by photon detector

Figure 14: DLS data taken of a solution of gold particles with a diameter of 10 nm. The signal is again overwhelmed by the forward scattering of likely dust in the instrument, with the same signs of contamination as seen in Figure 13.

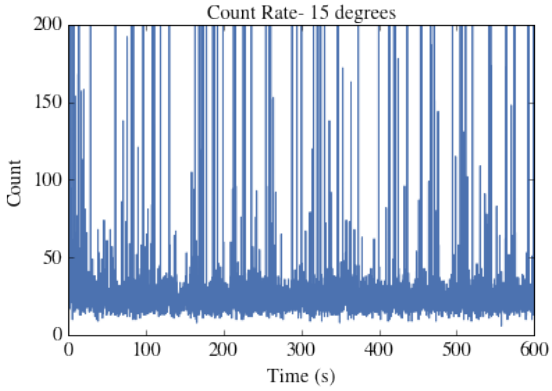
3.3.3 Dust Search

In order to isolate the cause of this contamination, two potential sources were investigated: contaminated cuvettes from improper cleaning and contaminated oil within the instrument itself. First, a cuvette containing only filtered Millipore water, prepared the same way as the two sample mentioned above, was measured. The count rate can be seen in Figure 15 (a). This sample was also rotated, with the count rates remaining unchanged; this indicated that the issue was not the result of scratches or soap residue on parts of the glass vial. Second, the sample was removed and the scattered count rate was recorded of just the oil bath that the sample was submerged in, as seen in Figure 15 (b). The fact that the scattered count rate remained very high even without any known scatterers submerged in the instrument was strong evidence that the dust is in the oil itself rather than inside the sample. This was not entirely unexpected, as the oil has not been changed or cleaned since the installation of the instrument into the lab in 2012, and the oil is exposed to

contamination any time a sample is inserted or removed.



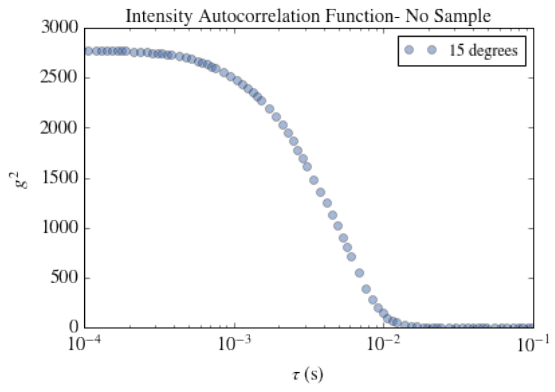
(a) Count rate of filtered millipore water in cleaned cuvette



(b) Count rate of DLS oil, with no sample inside the instrument

Figure 15: Count rate of scattered light at 15 degrees for two neutral samples: a cuvette of filtered water and an empty instrument. The high number of spikes in the count rate in both indicates contamination in the oil, rather than the cuvettes themselves.

Further information about the likely contaminants in the dust can be seen in the autocorrelation function of the intensity of light scattered from the oil inside the DLS instrument in Figure 16. The autocorrelation functions measured were highly angle dependent in both count rate and calculated radius. The count rate was much higher at lower angles, indicating that the particles are likely larger forward scatterers. The calculated hydrodynamic radius ranged from 8 nm to 441 nm, at angles ranging from 15 to 165. While this data is not a definitive calculation of the size of the contaminants, it does indicate the presence of large additional scatterers.



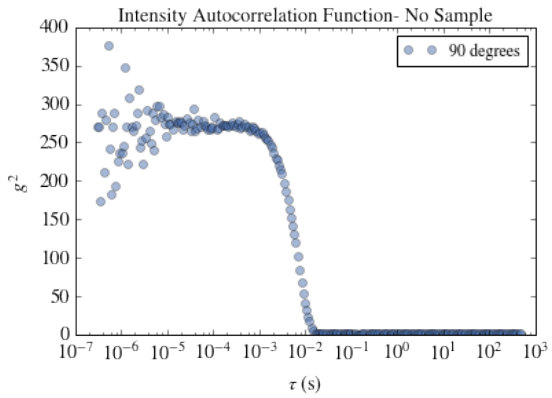


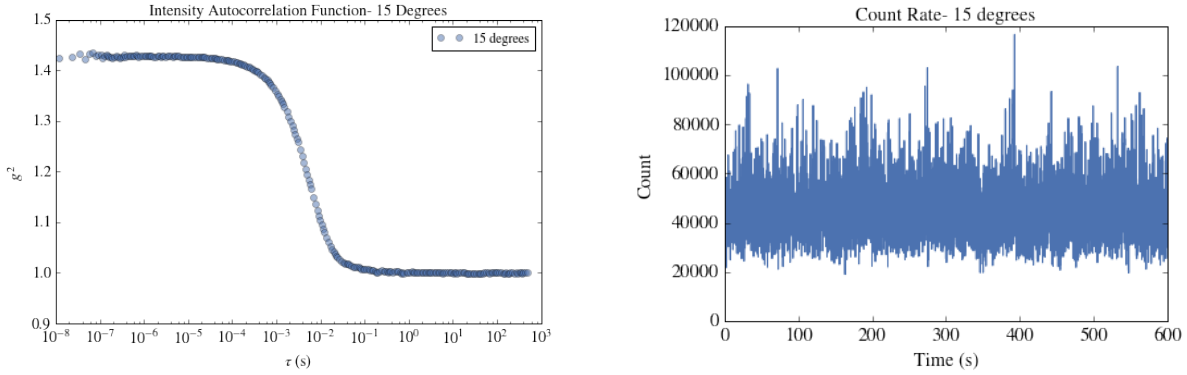
Figure 16: The autocorrelation function of the intensity of scattered light with no sample inside the instrument, such that the laser is scattering off particles suspended in the oil. The data above were taken at both 15 and 90 degrees; the autocorrelation functions displayed a high amount of angle dependence.

To combat this contamination, there were two possible approaches. The best option for the long term

was to change both the oil and the filter of the DLS instrument. Unfortunately, the required filter proved too difficult to obtain within the time span allowed by this thesis. Because clearer data was necessary in order to fully test the proposed Bayesian method, we proceeded with second option: increasing the size and concentration of the scatterers to the point where the scattered signal is able to dominate over the noise from the dust-contaminated oil.

3.3.4 Polystyrene Beads

To achieve this necessary increase in the hydrodynamic radius and concentration, a sample of 40 nm diameter polystyrene beads was prepared with a concentration of $0.5\,\mathrm{mg\,mL^{-1}}$. They were produced by Invitrogen, lot number 1934078. These beads have a hydrophobic surface due to the arrangement of polystyrene chains and a buildup of sulfate groups on their surface to provide a repulsive electric force to counteract van der Waals attraction at short ranges [24]. To prevent these charges from acting repulsively over a large range, the particles were dispersed in a 10 mM aqueous solution of NaCl. This solution was filtered through a 0.1 micron filter into a cleaned cuvette, and measured for 10 minutes at each of 11 angles. The first angle measured is shown below in Figure 17. As seen in this figure, the sample produced a clear single exponential decay without interference from a larger scatterer at larger τ. The count rate is also much more stable, high enough to get a strong signal with low uncertainty, and not overwhelmed by spikes from the dust inside the oil. This sample provided the bulk of the data that will be used in Chapter 6 to compare the results of standard fitting and Bayesian inference. Because of the issues faced with laser alignment, sample preparation, and noise, some data analyzed in Chapters 4 and 6 have been produced by previous experiments rather than in the course of this work.



(a) Autocorrelation function at 15 degrees

(b) Count rate at 15 degrees

Figure 17: Autocorrelation function and count rate of the 40 nm sample. The single exponential and more stable count rate indicate that the signal from the scatterers was enough to overcome the noise produced by the oil in the DLS instrument.

4 Standard Approach to Analysis

To analyze DLS autocorrelation functions using the equations derived in Chapter 2, the standard method for monodisperse solutions is to use a nonlinear least squares fit algorithm. In this chapter, this method is applied to the analysis of several sets of DLS data using both a single exponential decay and the method of cumulants. The results are outlined and discussed; in subsequent chapters, these will be compared to results found by a Bayesian inference method.

4.1 Single Exponential

Because the decay of the autocorrelation function can be described by a single exponential when the solution is completely monodisperse, as shown in equation 21, the most simple method of analysis is to fit the three unknowns of Equation 21 using a least squares fit, which minimizes the residuals of the function with respect to the data. These unknowns are the amplitude A, the baseline B (which is generally 1, but which some instruments set to be 0), and the diffusion coefficient D. The SciPy function scipy.optimize.curve-fit, a which uses an unweighted non-linear least squares to fit a given model to the data, was used to find these unknowns, given the basic decaying exponential model [25]. This function also outputs the covariance matrix of the fit, which can be used to calculate the uncertainty of the inferred values. The code for this analysis is listed in Appendix A.1.

After developing the code, it was used to analyze data previously obtained by Jerome Fung in 2015 of a roughly spherical enzyme, Guanosine-5'-monophosphate reductase in an aqueous buffer solution. This data was taken by a different DLS instrument than is described in Chapter 3; the instrument used to obtain this data was a ALV-5000, located in Brandeis University. The instrument uses a 632.8 nm HeNe laser. While this data does not have the noise issues discussed in Section 3.3, the data available is limited to only one scattering angle of 90 degrees. The raw data and the exponential fit found from the code in Appendix A.1 are shown in Figure 18. Because the error range of this data is not known, no error bars are assumed.

Because the data seem well described by the fit, as seen in Figure 18, the fit gives a good approximation for the hydrodynamic radius of the enzyme. The diffusion coefficient was fit as $4.746 \times 10^{-11}\,\mathrm{m^2 s^{-1}}$. Using Equation 22 and the known temperature recorded by the instrument to be 297.95 Kelvin, the hydrodynamic radius was found to be $5.166 \pm 0.030\,\mathrm{nm}$. The covariance matrix given by the fit was used to estimate the uncertainty of this value. While the fit does have some residuals, they are largely due the noise from the data rather than a failure in the model. This noise is mostly concentrated in the first points of data, most likely due to initial after-pulsing in the photodetector.

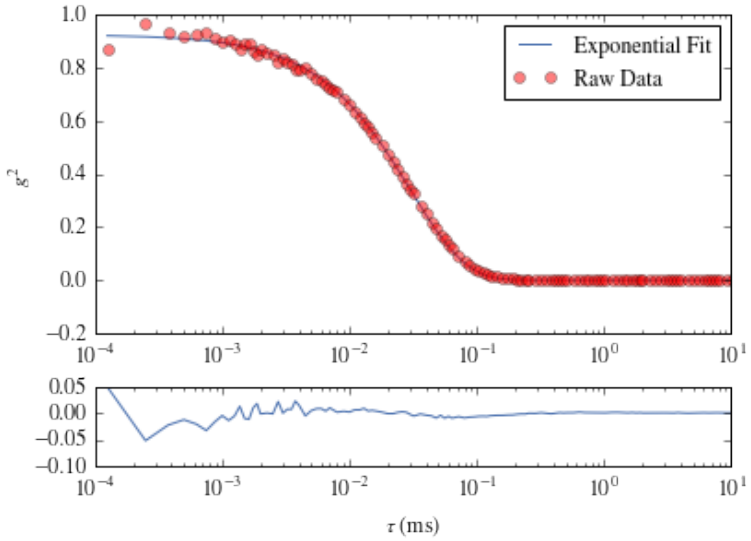
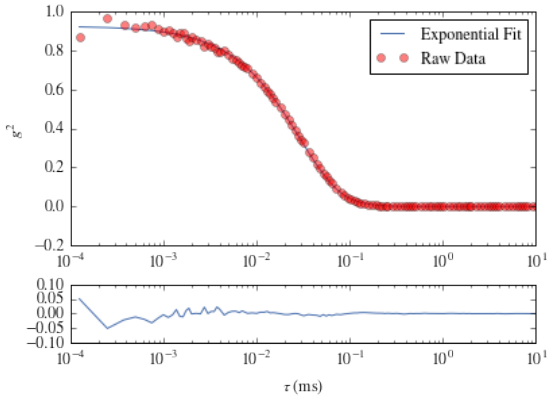


Figure 18: A least squares fit of data taken of an aqueous enzyme solution to a single exponential decay model. The residuals, the values given by the model subtracted by the values of the data, are plotted below the data and fit. No error model is assumed for the data, so no error bars are given.

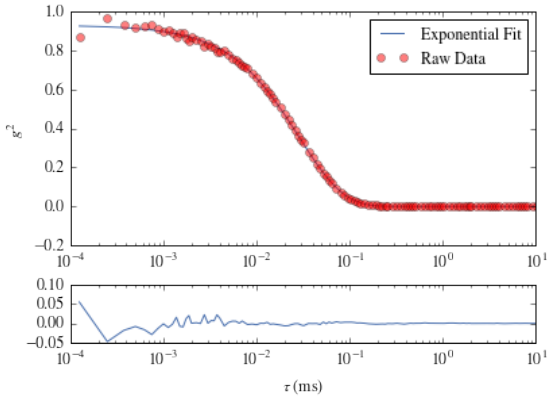
4.2 Method of Cumulants

Using the Python code listed in Appendix A.5, this same data set was analyzed using both the second and third order cumulant expansion of the single exponential decay, outlined in Equation 33. First, a second order expansion was used to fit the data. The results of this fit are shown in 19 (a). The values found by the fit were $\Gamma = 16.74 \pm 0.1\,\mathrm{s}^{-1}$ and $\mu_2 = 104.8 \pm 1.4\,\mathrm{s}^{-2}$. For comparison, the results of the third order fit are shown in Figure 19 (b). The third order method of cumulant includes an additional term that describes the skew of the size distribution. This fit found the values of $\Gamma = 17.60 \pm 0.1\,\mathrm{s}^{-1}$, $\mu_2 = 115.3 \pm 1.4\,\mathrm{s}^{-2}$, and $\mu_3 = -246 \pm 3403\,\mathrm{s}^{-3}$. Unfortunately, the large uncertainty on the value of μ_3 mean that it is almost completely non-informative; because the uncertainty encompasses both positive and negative numbers, the value is consistent with 0 and the direction of the skew cannot be determined. Regardless, the third order cumulant fit still produced physically reasonable values for D and μ_2, and small amounts of residuals.

The results of these three different fits on the same data are shown in the table below. Although increasing the number of parameters to be fit does reduce the sum of the residuals squared, and thus leads to a fit that better describes the data, the difference between the radius found in the second order and third order fits is small compared to the difference between the single order and second exponential. While a third order expansion can be helpful when exploring the value of the skew of the data, it is generally not necessary; in this case especially, the uncertainty in the value fit for μ_3 indicates the unreliability of this value.



(a) Second order cumulant fit



(b) Third order cumulant fit

Figure 19: A least squares fit of data taken of an aqueous enzyme solution to both a second order cumulants expansion and a third order cumulants expansion. The difference in the residuals between these models is very small, and the results found by the fits are comparable.

Method	Calculated Radius	Sum of Residuals Squared
Single Exponential	$4.746 \pm 0.030\,\text{nm}$	0.011389
Second Order Cumulant	$4.480 \pm 0.027\,\text{nm}$	0.010662
Third Order Cumulant	$4.261 \pm 0.049\,\text{nm}$	0.009607

4.3 Exponential Fit Across Multiangle Data

When taking DLS measurements across several angles, the behavior of the exponential is determined by the angle dependence of the scattering vector q. The value for the diffusion constant D should remain constant across these measurements, however. The angle dependence of DLS data can be seen in the multi-angle results of the 40 nm polystyrene bead sample, shown in Figure 20. The data were collected over 11 angles, for 600 seconds each. While the amplitude is roughly constant across these measurements, the steepness of the exponential decay increases with increasing scattering angle. Each angle was fit to a single exponential decay using the least square fit code listed in Appendix A.3.

There are two strong indications of the monodispersity of this data. First, the data at all angles are well described by a single exponential model, with only slight deviation. Another way to examine the monodispersity of a sample is to plot the fitted value for the decay constant Γ against the value for the scattering vector q^2. For a completely monodisperse solution, this relationship is perfectly linear, with the slope of the line equal to the diffusion constant D. This plot for the 40 nm data is shown in Figure 21. This sample is strongly monodisperse, as shown by the strong agreement between the data and the theoretical line. The slope of this line was fit to be $D = (9.34 \pm 0.04) \times 10^{-12}\,\text{m}^2\text{s}^{-1}$; this gives a hydrodynamic radius

of $R = 24.66 \pm 0.10\,\text{nm}$. While the expected value for these beads was 20 nm, the standard deviation of particle size was given by the producer (Invitrogen by Thermo Fisher, Lot number 1934078) as 6 nm. The calculated radius is then within the specifications given by the manufacturer; thus, the 40 nm polystyrene beads will function well as a calibration for the development of the analytical models.

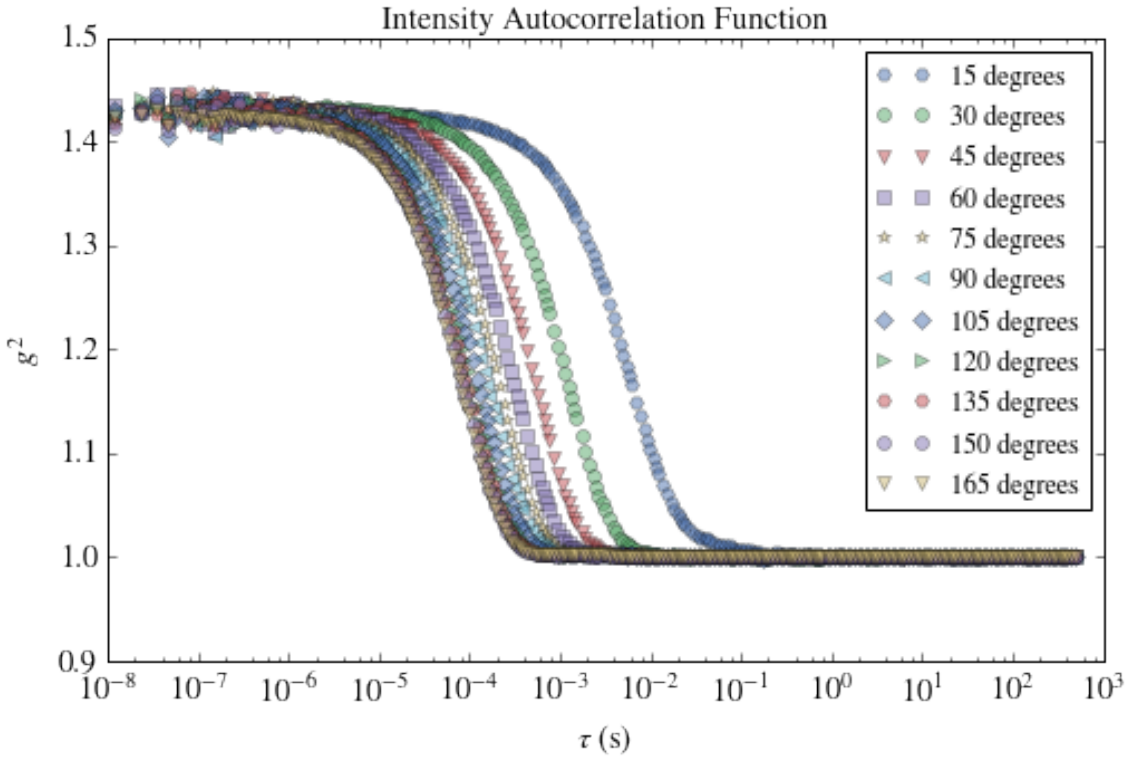


Figure 20: The intensity autocorrelation function of light scatted by 40 nm polystyrene beads suspended in water, taken across several angles. The angle dependence of the decay coefficient can be seen in above; as the value of q^2 increases with increasing angle, the decay time Γ increases, leading to a steeper exponential decay.

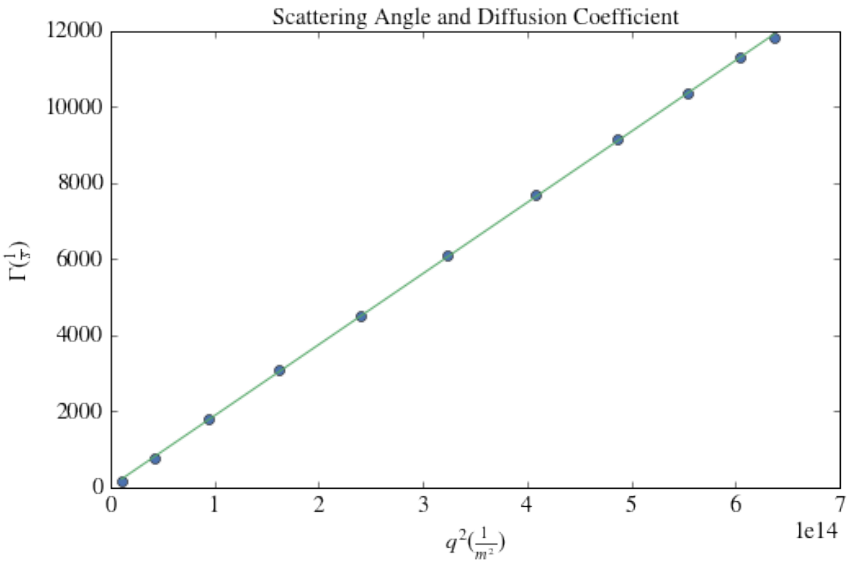


Figure 21: The results of a standard non-weighted least squares fit of the decay times found for each angle of the 40 nm data to a linear model. The diffusion coefficient D is equal to the slope of the line shown. The strongly linear relationship indicates the monodispersity of the sample.

4.4 Method of Cumulants, Multiangle

Like the diffusion coefficient, the polydispersity index, which is the ratio between the second order cumulant μ_2 and the decay time $\overline{\Gamma}^2$, should also be independent of the angle of measurement. The intensity autocorrelation function for each angle of the 40 nm data was fit to a second order cumulant model. The results are listed in the table below. The fit found a huge amount of variation in the fit value for μ_2, which was not unexpected, as the value for Γ also spans orders of magnitude. The value for the polydispersity index, however, was not as constant as expected. Although all the values are low enough that the sample can be considered monodisperse, there was an angle dependence across the values, which indicates that the size distribution of beads in the sample is not as sharply peaked as expected. This could possibly be due to slight aggregation within the sample, or it could be a slight signal from the dust in the oil. Regardless of this deviation from the model for a monodisperse solution, all inferred radii were within the known value, and the sums of the residuals squared were comparable to those of the single exponential fit, which had an average residual value of 0.00245.

Method	μ_2	$\frac{\mu_2}{\bar{\Gamma}^2}$	Calculated Radius	Sum of Residuals
Least Squares Fit: 15 degrees	$4,439\,\mathrm{s}^{-2}$	0.5357	$26.0 \pm 0.19\,\mathrm{nm}$	0.0020
Least Squares Fit: 30 degrees	$43,096\,\mathrm{s}^{-2}$	0.2446	$22.2 \pm 0.10\,\mathrm{nm}$	0.0008
Least Squares Fit: 45 degrees	$152,660\,\mathrm{s}^{-2}$	0.1716	$21.6 \pm 0.12\,\mathrm{nm}$	0.0012
Least Squares Fit: 60 degrees	$336,230\,\mathrm{s}^{-2}$	0.1318	$21.7 \pm 0.18\,\mathrm{nm}$	0.0026
Least Squares Fit: 75 degrees	$556,451\,\mathrm{s}^{-2}$	0.1026	$22.1 \pm 0.15\,\mathrm{nm}$	0.0017
Least Squares Fit: 90 degrees	$910,612\,\mathrm{s}^{-2}$	0.0930	$22.2 \pm 0.16\,\mathrm{nm}$	0.0019
Least Squares Fit: 105 degrees	$1,296,302\,\mathrm{s}^{-2}$	0.0837	$22.2 \pm 0.19\,\mathrm{nm}$	0.0027
Least Squares Fit: 120 degrees	$1,698,098\,\mathrm{s}^{-2}$	0.0780	$22.3 \pm 0.15\,\mathrm{nm}$	0.0018
Least Squares Fit: 135 degrees	$2,471,329\,\mathrm{s}^{-2}$	0.0876	$22.3 \pm 0.16\,\mathrm{nm}$	0.0020
Least Squares Fit: 150 degrees	$1,860,606\,\mathrm{s}^{-2}$	0.0566	$22.6 \pm 0.14\,\mathrm{nm}$	0.0014
Least Squares Fit: 165 degrees	$2,024,584\,\mathrm{s}^{-2}$	0.0563	$22.8 \pm 0.08\,\mathrm{nm}$	0.0005

The method of least squares fitting described in this chapter has the benefit of being computationally cheap, and is more than sufficient for a single angle measurement of strongly monodisperse data. The method becomes less useful, however, when considering a large amount of data, or data that are no longer well described by the model. There are two main limitations to using least squares fitting to multi-angle DLS data. First, the method fits each angle individually, and then fits a straight line to those fits or takes an average. Either method requires an intermediate step to find the final value for the diffusion coefficient. It would obviously be preferential to use the entire data set as a whole to determine the constant value of the diffusion coefficient, rather that finding it by a fit made to points that are already fits themselves. Another limitation of traditional least squares fitting methods is the fact that the methods do not make it clear when there is another likely fit to the data; instead, the functions output only the most likely values and their covariance matrix. If there are sources of noise or uncertainty in the data, it is possible that multiple values for the parameters can well describe the data. By only giving one parameter value with a set uncertainty, the traditional fitting method loses information about other likely parameter values. A method that returns a probability distribution of the fit for D along with the most probable value would address these issues, which are common when analyzing real data of unknown sources. The following chapter will introduce another approach to analysis that will address these problems; this new method will be developed and applied in Chapters 6 and 7.

5 A Bayesian Approach

This chapter introduces the fundamentals of Bayesian inference, which will provide the groundwork for the development of the Bayesian inference approach to DLS analysis. Bayes' theorem is defined, and several example problems are used to explore the implications of the theory. Finally, the Python package emcee, which uses an MCMC algorithm, is introduced; it will subsequently be used in the algorithm for determining the size distribution of particles from DLS data, developed in Chapter 6.

5.1 Introduction to Bayesian Statistics

Because DLS analysis attempts to recreate the size distribution that would produce the pattern of intensity fluctuations by observing those fluctuations, it is an inverse problem rather than a forward problem. Because the problem is ill-conditioned, it is highly sensitive to noise in the data. To analyze DLS data, it is necessary to use a statistical inference to determine the most likely size distribution. In this chapter, we will define and apply a Bayesian inference approach to model fitting, which will later be applied to infer size distributions from DLS data.

Bayesian statistics approaches probability as a degree of belief. The application of a Bayesian approach to data analysis uses data to update the degree of belief as more data become available. It rests on what is known as Bayes' theorem, which states

$$P(x|y,I) = \frac{P(y|x,I)P(x|I)}{P(y|I)} \tag{37}$$

where the notation $P(x|y)$ stands for the probability of x being true given that y is true. Here, I represents all other information known about the system. In words, Bayes' theorem states that the probability of a hypothesis being true given the available data is equal to the probability of the data being true given the hypothesis multiplied by the probability of the hypothesis being true, and divided by a normalizing value of the probability of the data being true. The value $P(x|y,I)$ is known as the posterior probability, and can be written as $P(hypothesis|data,I)$. The value $P(y|x,I)$ is known as the likelihood function, and can be written as $P(data|hypothesis,I)$. Finally, the value $P(x|I)$ is known as the prior, and can be written as $P(hypothesis|I)$ [3].

Of these, the prior has often been the focus of skepticism. It requires that one make an assumption about the hypothesis, and determine a priori whether it is a reasonable hypothesis or not. This can make many users uncomfortable, as it introduces the opportunity for human bias in the assessment. While it is true that a carefully chosen prior can influence the resulting probability density function of the posterior when there is

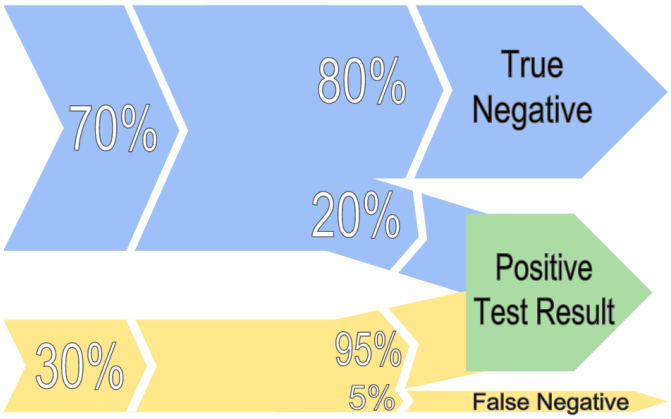


Figure 22: Probability flow chart showing the populations carrying a disease, shown in yellow, and those who are not, shown in blue. While most of the sick population is detected by a positive test result, there are also many false negatives that must be taken into account to determine the true posterior probability that someone with a positive test result actually has the disease.

little data, when large amounts of data are available, the chosen prior has almost no influence. The example in Section 5.1.2 illustrates this point. Before this is explored, however, a simple example of the application of Bayesian statistics is explained.

5.1.1 Application of Bayesian Statistics

It can be difficult to intuitively understand Bayes' theorem without an understanding of the prior probability. One example where the prior is most clear is in medical tests used to determine if someone has a disease; here, the relative odds of having a disease are well known, and provide a concrete and easily understandable prior. In this example, we can clearly define the likelihood and prior, and use them to come to a conclusion about the posterior probability.

For a hypothetical medical test to determine if a patient is carrying the flu, there is a known rate of 5% false negatives, where 5% of flu carriers will not get a positive result. The test is also known to have a high rate of 20% false positives, where 20% of non-carriers are told that they have the flu. If someone receives a positive result on this test, however, we know more information than just this test result. We also know how common the flu is in the population, or the prior odds that someone will actually be sick with the flu. If at any given flu season, 30% of the population will have the flu, then the prior odds that someone will be sick become 0.3.

One way to visualize this is through a flow diagram, as shown in Figure 22. Here, we know that the positive test result puts the patient in the green section; it is possible to work backwards to determine the odds that that person is actually sick. Even though the test has a 95% accuracy for those who are sick, it is clear from the flow chart that many people who do not have the disease also end up with positive test

results. The true posterior probability that the person is actually sick given a positive test result is given by

$$\frac{P(sick)}{P(healthy)} \times \frac{P(positive \mid sick)}{P(positive \mid healthy)} = \frac{P(sick \mid positive)}{P(healthy \mid positive)} \tag{38}$$

which states that the relative probability of being sick given a positive test result is equal to the relative probability of being sick times the relative probability of getting a positive test results given that one is sick. In other words, the posterior probability of being sick given a positive result is equal to the prior probability of being sick times the likelihood of getting a positive test result given that it is true that someone is sick.

Plugging in the numbers given in the hypothetical, the true posterior probability can be found

$$\frac{0.3}{0.7} \times \frac{0.95}{0.2} = 2.04 = \frac{P(sick \mid positive)}{P(healthy \mid positive)} \tag{39}$$

To get the true probability of being sick given a positive test result, we must normalize this probability, using the fact that

$$P(sick \mid positive) + P(healthy \mid positive) = 1 \tag{40}$$

as every patient must be either sick or healthy. After normalizing this probability, the final probability that someone with a positive test result is actually sick with the flu is equal to 67.1%.

This can be seen more clearly if we imagine a population of 1,000 patients. Of the 700 total patients who are healthy, 140 would produce a false positive test result and 560 would produce a true negative test result. Of the 300 patients who are actually ill with the flu, 285 would produce a true positive test result and 15 would produce a false negative test results. This makes it clear that of the population in the "positive test result" section of the flow chart, 285 are actually sick and 140 are not. Thus, the probability of being sick given a positive test result is

$$P(sick \mid positive) = \frac{285}{140 + 285} \times \frac{0.95}{0.2} = 0.671 \tag{41}$$

What we just arrived at using intuitive statistical reasoning can also be found directly using Bayes' theorem, given by

$$P(sick|positive) = \frac{P(positive|sick)P(sick)}{P(positive)} = \frac{(0.95) \times (0.3)}{(0.7 \times 0.2 + 0.3 \times 0.95)} = 0.671 \tag{42}$$

where the probability of positive results given that you are sick is 95%, the probability of being sick as a whole is 30%, and the probability of getting a positive test result is the probability of getting a positive result

if you are healthy times the probability of being healthy plus the probability of getting a positive result if you are sick times the probability of being sick. Although less immediately intuitive, Bayes' theorem greatly simplifies this calculation.

In this example, there is just one point of data, as the test has only been run one time. Even with this limited data set, however, we can use Bayes' theorem to incorporate other prior knowledge to inform our degree of belief in the results. With a larger set of data, the conclusions become more certain.

5.1.2 Priors and Their Effect in Parameter Estimation

In most applications of Bayesian inference, the posterior likelihood is not described by a single number. Instead, Bayes' theorem gives a probability distribution, showing the relative likelihood of each possible value of the parameter. To explore a more complex example and to show the effect of choosing different priors, we can consider flipping a coin to determine how fair it is, meaning how likely it is to land of heads for any given flip (modeled from an example problem in [3]). Here, the posterior probability is not simply stating the probability that it is fair, as the bias of a coin is not a binary variable like sickness or health. Instead, it is possible that the coin would land on heads anywhere between 0% and 100% of the time, giving a possible bias of 0 to 1. The posterior probability then becomes a probability distribution, with the peak of the distribution indicating the most probable bias of the coin.

To write down the likelihood function, recall that it is equal to $P(data|hypothesis)$. In this example, it is the probability of getting a set number of heads in a set number of tosses, given that the coin has a given bias. Because this coin only has two options, heads or tails, this becomes a binomial probability distribution, only determined by the bias of the coin and the number of tosses. It can be written as

$$likelihood = H^n(1-H)^{N-n}$$

where H is the bias of the coin, n is the number of heads, and N is the number of times the coin has been flipped. H represents the probability of getting heads, while $1-H$ is the probability of getting tails.

While those values are easy to determine, choosing a prior might give pause. Given a limited set of coin flips, the choice of priors can strongly influence our conclusion about the bias of the coin, but we must make an assumption about how fair we think the coin is. We might say that most coins are fair, and define the prior as a sharply peaked Gaussian around 50% bias. We might also express strong skepticism of the bias of this coin and think that it is most likely a trick coin; this would lead us to define the prior as strongly peaked at both 0% and 100%, but not between. Most commonly, the prior would be defined as uniform between the range of known values, from 0% to 100%. Each of these possible priors have different levels of assumption.

In general, it is best to chose a bounded uniform prior when lacking information about a system. [3] [26].

The results of choosing each of these possible priors are shown in Figure 23, which are plots of the resulting posterior probability density function. For each, data were randomly generated with a coin biased at 25% heads. In the first plot, the coin has been flipped 0 times, so the posterior function is equal to the prior for each probability function: a uniform, bounded prior between 0 and 1, a Gaussian distribution peaked at 0.5, and a bounded double Gaussian with peaks at 0 and 1. In Figure 23 (b), the coin has been flipped twice, resulting in 1 heads and 1 tails. By 32 flips, in Figure 23 (c), the probability functions have begun to collapse to similar values closer to the true value of 0.25. Finally, by 1000 flips in Figure 23 (d), the probability functions are almost identical, and very sharply peaked.

While the priors are strongly influential when the data set is limited, it only takes on the order of tens of flips for the posterior functions to collapse to a reasonable guess of the bias of the coins, regardless of the chosen prior. It is also evident that the probability function gives a measure of uncertainty in addition to the most likely value. The sharper the peak, the lower the uncertainty of the value. The full width at half maximum of the probability curve is often the best way to describe the shape of a Gaussian probability curve; it becomes more complicated when the resulting curve has multiple peaks, as in subfigure (b) of Figure 23.

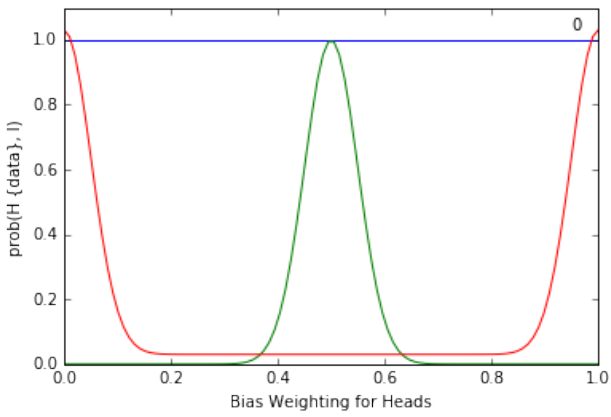
5.2 Probability Density Functions Across Multiple Parameters

When fitting more than one parameter using a set of data, the probability density function becomes a function of multiple variables. The most likely values for the parameters are found by finding the peak of the multidimensional probability surface. This is known as the maximum *a posteriori* probability, which is equal to the mode of the posterior distribution function, and provides a value for an unknown parameter. To describe the result for a single variable, it is possible to integrate out, or marginalize over, the variables not of interest. For example, if one were fitting both the baseline and amplitude of the signal, but were only interested in the amplitude, the desired probability density function can be written as

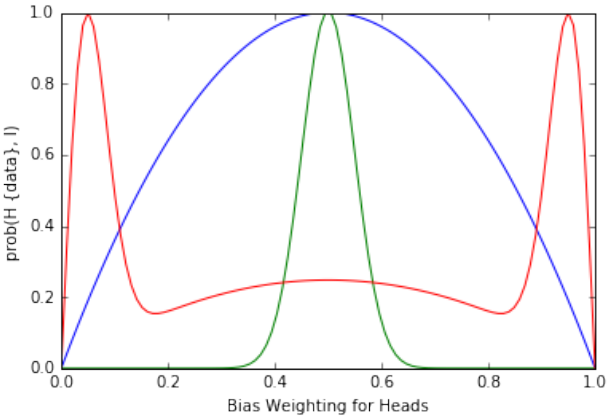
$$P(A \mid data) = \int_0^\infty P(A, B \mid data) dB \tag{43}$$

where A is the amplitude and B is the baseline. By marginalizing out nuisance parameters, it is possible to focus only on the probability distribution of the single parameter of interest.

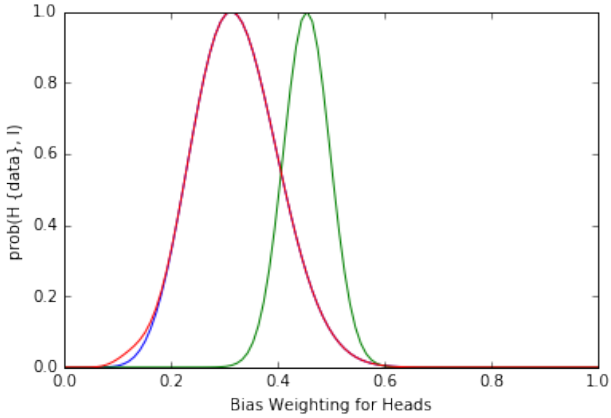
When inferring more than one parameter, however, it is often important to take into account the correlation between the two parameters when determining the the maximum *a posteriori* values. The correlation between two variables can be seen in the skew and elongation of the probability density surface, as shown in Figure 24. Ideally, the estimation of one parameter would have no effect on the other parameter; in this



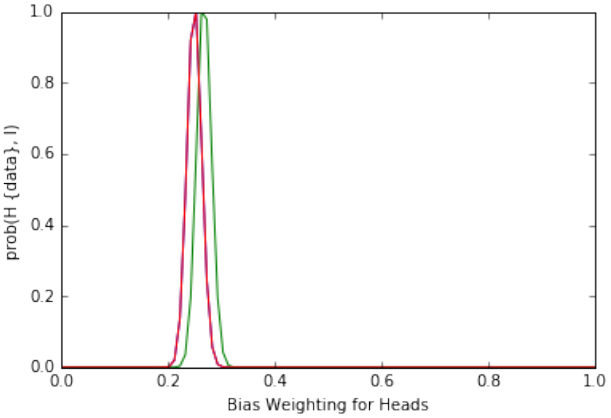
(a) Number of Flips: 0



(b) Number of Flips: 2



(c) Number of Flips: 32



(d) Number of Flips: 1000

Figure 23: The resulting probability density function of the posterior, given a different number of coin flips. As the amount of data increases, the effects of the choice of the prior decreases; eventually, the likelihoods all collapse around one value.

situation, the probability density surface would have a direction parallel to the coordinate axes, with zero skew, as there would be no correlation between the two variables. If the variables are correlated, meaning that the estimation of one parameter influences the inferred value of the other, the resulting probability density surface becomes skewed. The direction of the skew indicates whether the variables have a positive or negative correlation. For a positive skew, the shape indicates that when changing one variable by a slight positive amount, the other correlated variable also must increase to make the parameters statistically probable.

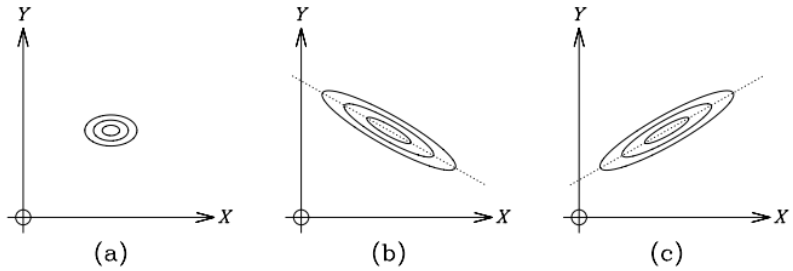


Figure 24: Three probability density surfaces showing the relationship between two variables. (A) shows two variables that are uncorrelated. (B) shows two variables that have a strong negative covariance and correlation. (C) shows two variables that have a strong positive covariance and correlation. [3]

5.2.1 Sampling Probability Functions Using Markov Chain Monte Carlo

When fitting more than three parameters, it becomes impossible to plot the probability density function, and much more difficult to sample the multidimensional surface to find the maximum *a posteriori*. When inferring 7 parameters, for example, even if we were to sample just 100 points along each dimension, this would become 100^7 or 100 trillion points. Rapidly, it becomes prohibitively computationally expensive to fully explore every point on the probability density function. In this case, it is necessary to use computer algorithms to sample points across the surface.

One class of methods, Markov chain Monte Carlo (MCMC) algorithms, uses a chain of steps to sample and explore the probability density function. Monte Carlo methods refer to a sampling technique that can be used to model complex systems and estimate some parameter without directly calculating it. To do so, Monte Carlo simulations sample random points around the parameter space, and use that sample as representative of the space. For MCMC algorithms, this sampling occurs as a directed random walk. The algorithm creates this random walk around the parameter space by sampling points, evaluating those points, and choosing a new direction to move to within the parameter space; these steps are recored and form a Markov chain. After many steps, the algorithm's walk makes a representative sample of the distribution [27]. An algorithm determines how the walker will move for each step. First, the algorithm proposes a step.

For each step, the algorithm decides whether to accept the step into the chain, or reject it; the methods used to do this vary by type of MCMC algorithm, but generally include some probabilistic favoring of steps that move the sample towards the maximum of the probability function, rather than isolated in an uninteresting minimum. [28] Ultimately, because of this favoring, the walkers spend more time in the region of parameter space where the posterior is largest, providing an estimation of the posterior probability and an inference of the most likely value for the parameter.

In Python, there exists a package called emcee, a Python implementation of an affine-invariant MCMC algorithm that allows for efficient exploration of the probability function by using an ensemble of walkers. These multiple walkers explore the posterior probability simultaneously, which both allows for faster results and prevents the walkers from being stuck in some local minimum or maximum. [28] Emcee has recently been used for Bayesian inference in numerous astrophysical papers [29], [30], [31]; here, we are applying it to a much different length scale, though the essentials remain the same.

To explore how emcee can be used to infer parameters from a set of data, we can use it fit a straight line to a typical set of noisy data (modeled off the example problem in [32]). The data were generated randomly with set nonuniform uncertainties. First, a traditional nonlinear least squares fit was performed, using the SciPy function scipy.optimize.curve-fit to fit the slope m and the intercept b. This best fit is plotted with the data in Figure 25.

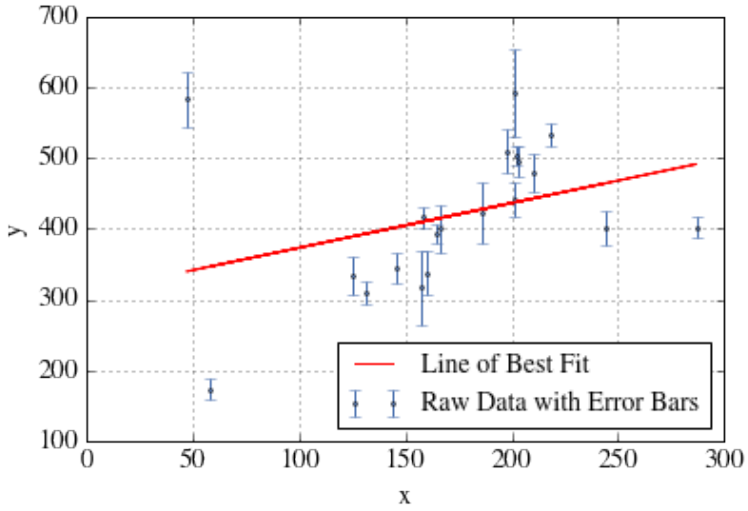


Figure 25: Line of best fit of a linear model to data with random uncertainties found using traditional nonlinear least squares fit method. The noisy data and clear outliers prevent a clear fit.

Next, this same interference was performed using emcee (note that the Python code for this fit is not included, as the exact same method is outlined in Appendix A.2). First, it is necessary to define the prior and the likelihood functions, as emcee takes the natural log of these values as inputs. The prior was assumed

to be uniform and unbounded. The likelihood function has been derived by Hogg [32] for general fits to be

$$\ln \mathcal{L} = K - \sum_{i=1}^{N} \frac{[y_i - mx_i - b]^2}{2\sigma_{y_i}^2} = K - \frac{1}{2}\chi^2 \tag{44}$$

where m and b are the parameters to be fit, σ_{y_i} is the uncertainty of a given data point, χ is equal to the residuals, and K is a constant equal to

$$K = \sum_{i=1}^{N} \ln \frac{1}{\sqrt{2\pi\sigma_{y_i}^2}} \tag{45}$$

The natural log of the posterior probability is given simply by summing the log of the prior and the likelihood functions.

In order for the emcee walkers to explore the posterior probability, they must first be initialized at a starting position in the parameter space. It is important to chose a reasonable starting position; if the walkers are too far from the maximum, they will not be able to find it within the number of steps allotted. The result from the least squares fit was used here to initialize the position of the walkers with respect to b and m. The paths explored by the walkers are shown in Figure 26. They initially are confined to the area around the initial positions, but after around 100 steps they begin to fully explore the posterior distribution.

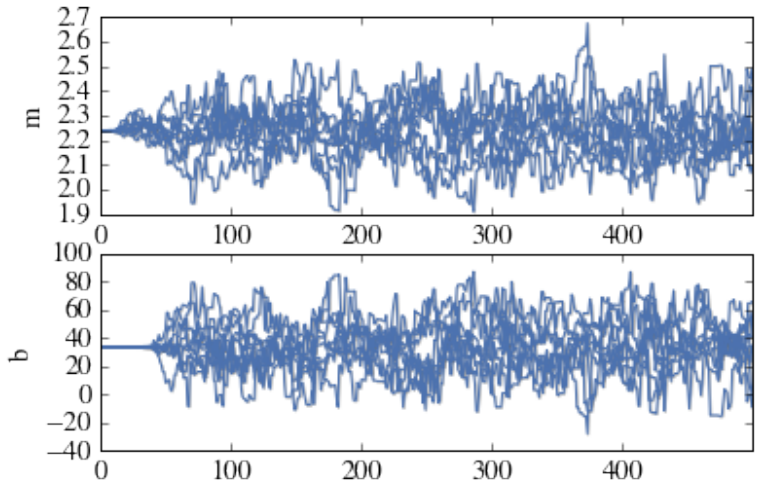
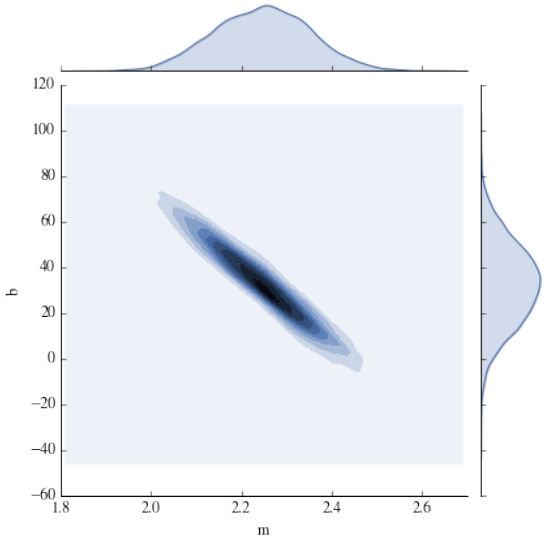


Figure 26: The trace of the values explored by the emcee walkers in the probability surface of m and b, with the values of each parameter on the y axis and the number of steps taken on the x axis. After starting at the initial guess set by the code, it takes around 100 steps for each to equilibrate.

The results of the emcee walkers are shown in Figure 27. The probability surface of the two parameters is shown in subfigure (a). The high skew of the ellipse indicates that b and m are highly correlated, which is expected. If the line of best fit had a slightly different slope, it would require a different intercept to fit the data. The marginalized probability distributions are also shown on the edges of the plot of the

probability surface in subfigure (a); these plots allow for a more complete description of the fits found by emcee. Although the marginalized probabilities are roughly Gaussian, there are some small secondary peaks for the slope m; this is logical, given the spread within the data. The roughly Gaussian shape to the data also give a clear uncertainty to the inferred values, indicated by the width of the curves. The numerical results are shown in subfigure (b); this table shows the values of the 16th and 84th percentile of the marginal distributions for each parameter. These percentiles correspond to $\pm 1.0\sigma$ for a Gaussian distribution.. While the intercept b has a large uncertainty, the value of the slope m is more closely bounded. The most probable values found by emcee are plotted in Figure 28.



(a) Probability surface of the two parameters

	b	m
0.16	14.422424	2.138046
0.50	32.921615	2.248605
0.84	50.938547	2.351682

(b) Most probable fits and confidence intervals

Figure 27: (a) shows the resulting probability surface of the slope and intercept found by the emcee walkers, as well as the marginalized probability of each, with the other parameter integrated out. The probability surface is shown in the center of (a), while the marginalized probabilities are shown on the edges. The two parameters are highly correlated, as shown by the elliptical shape of the surface. (b) shows the numerical results and the confidence interval of each parameter, within 1 σ.

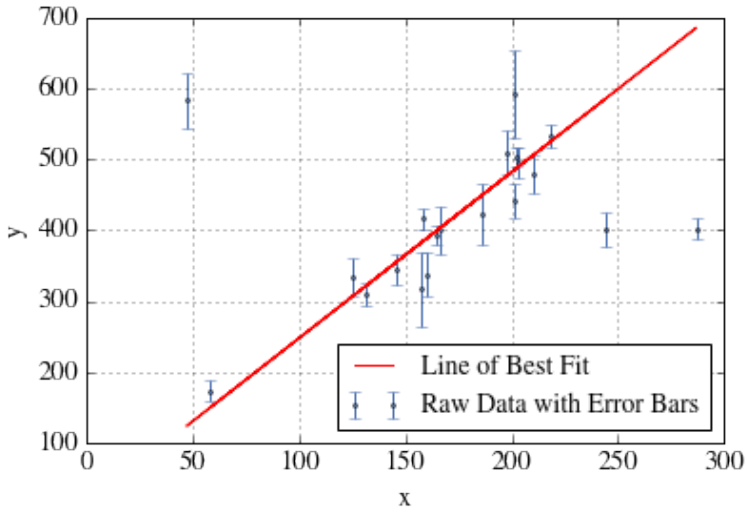


Figure 28: Line of best fit of a linear model to data with random uncertainties found using Bayesian inference with emcee. Although the residuals here are higher that the least squares fit, the Bayesian fit better describes the main trend of the data, instead of skewing to include outliers.

The results of the two fits are described in the table below. When compared to Figure 25, the line found using Bayesian inference is less sensitive to outliers and more accurately describes the main trend of the data. Although the residuals for the Bayesian fit are higher, the individual uncertainties of the values are lower. Further, the benefits of emcee are apparent in the probability surface shown in Figure 27 (a), which not only demonstrates the high level of correlation between the variables, but also shows the shape of the probability function for each marginalized parameter.

	Fit for Intercept	Fit for Slope	Sums of Residuals Squared
Least Squares Fit	310 ± 73	0.24 ± 0.6	175016.92
Bayesian Inference	33 ± 18	2.25 ± 0.1	350541.17

Overall, emcee provides an efficient and robust algorithm to explore a multidimensional probability function, and infer the most probable value for each parameter using Bayesian inference. In the next chapter, this method will be applied to DLS data to describe single and multi-angle data using both a single exponential model and a cumulant model of the decay of the autocorrelation function.

6 Developing a Bayesian Model

Using the computational tools developed in the previous chapter and the governing equations derived in Chapter 2, a Bayesian framework was applied to size inference from DLS data. This chapter discusses the process for the development of that framework, including determining the necessary starting position for emcee walkers and the effect of the choice of priors. Once these developmental issues are discussed, the algorithm is applied to the 40 nm polystyrene calibration data. These data are analyzed using both a single exponential and the method of cumulants with Bayesian inference. Finally, additional data of 100 nm gold particles are analyzed to explore the effect that noise such as aggregation has on the Bayesian inference algorithm developed.

6.1 Fitting Using Emcee: Starting Position and Step Size

To develop a Bayesian inference model, a similar process was used as outlined in the previous chapter to infer a straight line fit to data. The codes developed for single angle DLS inference, multi-angle DLS inference, and method of cumulants inference are listed in Appendices A.2, A.4, and A.6 respectively. First, the Bayesian inference of single angle DLS data will be discussed.

First, it is necessary to define the natural log of the likelihood function. Adapting the approach developed by Hogg (2010) [32], we can replaced the straight line model $y = mx + b$ with the model for the decay of the autocorrelation function for a monodisperse solution $g^{(2)}(\tau) = Ae^{-2Dq^2\tau} + B$. This gives a likelihood function of

$$\ln \mathcal{L} = K - \sum_{i=1}^{N} \frac{[g_i - A\exp\{-2Dq^2\} - B]^2}{2\sigma_{g_i}^2} = K - \frac{1}{2}\chi^2 \tag{46}$$

where A, D, and B are the parameters to be inferred. Here, A is the amplitude of the decay, B is the baseline, and D is the diffusion coefficient. The scattering vector q is angle dependent, but is known given the parameters of the experiment, so it is not necessary to infer the value. The value g_i here is each value of the autocorrelation function of the intensity, and σ_{g_i} is the uncertainty of a given data point. Finally, χ is equal to the residuals, and K is a constant equal to

$$K = \sum_{i=1}^{N} \ln \frac{1}{\sqrt{2\pi\sigma_{g_i}^2}} \tag{47}$$

Because the algorithm emcee requires that the uncertainty on the values be explicitly defined, it was necessary to estimate a noise model. For the purposes of development, the model was assumed to be $\pm 5\%$ for each value. This produced larger uncertainty for the largest values of the intensity autocorrelation function,

which occur at the beginning of the decay when the time τ is the smallest. While it does not completely describe the true uncertainties on these values, which should depend on the number of measurements that go into each point of the autocorrelation function such that the uncertainty would be proportional to $\frac{1}{\sqrt{N}}$, this noise model is reasonable for the development of this model.

Second, it is necessary to define the prior. The effect of using different priors will be explored in the next section, but for the purposes of development, the prior was assumed to be uniform and bounded. The bounds placed on the prior for each value were determined by the physical constraints of the experiment. First, it is known theoretically that the amplitude A, baseline B, and diffusion coefficient D must be positive numbers, so a lower bound of 0 was placed on all the values. It is also known that the amplitude given by this instrument is generally between 1 and 2; thus, a conservative upper bound of 10 was placed on A. No further constraints were placed initially, to allow for a non-assuming model. Because emcee takes the input of the natural log of the likelihood function, any values outside of the bounds on the prior return a value of $\ln(0) = -\infty$, as the value is undefined.

With these functions defined, we can use Bayes' Theorem, which defines the posterior probability as the normalized product of the prior and the likelihood, to define the posterior probability as

$$\ln P(A, B, D|g) = \ln P(g|A, B, D) + \ln P(A, B, D) \tag{48}$$

To sample this posterior probability density, emcee uses the function emcee.EnsembleSampler. This function takes the variables of the number of walkers sampling the posterior probability, the number of parameters being inferred, the value of the log of posterior (as defined above), and the input values of τ, $g^{(2)}$, and σ_g.

Another input required by emcee is the starting position of these walkers; because the walkers need to be relatively close to the maximum *a posteriori* values, the algorithm is highly sensitive to initial conditions. Unless the initial positions of the walkers are close to the values of the maximum *a posteriori* probability, they will not be able to find it within the number of steps allotted. The input values used in this model are found using the traditional least squares fit analysis developed in Chapter 4.

Below are the results of this code applied to the first angle of data from 100 nm gold particles, using the initial values $A = 1.4$, $D = 4.35 \times 10^{-12}\mathrm{m}^2\mathrm{s}^{-1}$, and $B = 1$. While the code ran without issue, the results given in Figures 29 and 30 make no physical sense. A diffusion coefficient of 0.5 would give a radius of $R = 4.28 \times 10^{-19}$ m, which is 4 orders of magnitude smaller than the size of an electron. The source of the issue with this fit can be seen more clearly in the path taken by the walkers, shown in Figure 29. While they start close to 0, they rapidly move orders of magnitude away from the area of maximum probability. This is due to the step size the walkers must take to explore the parameter space of A and B, which are both orders

of magnitude larger than D; to fully explore all the variables, the walkers lose any sensible value of D.

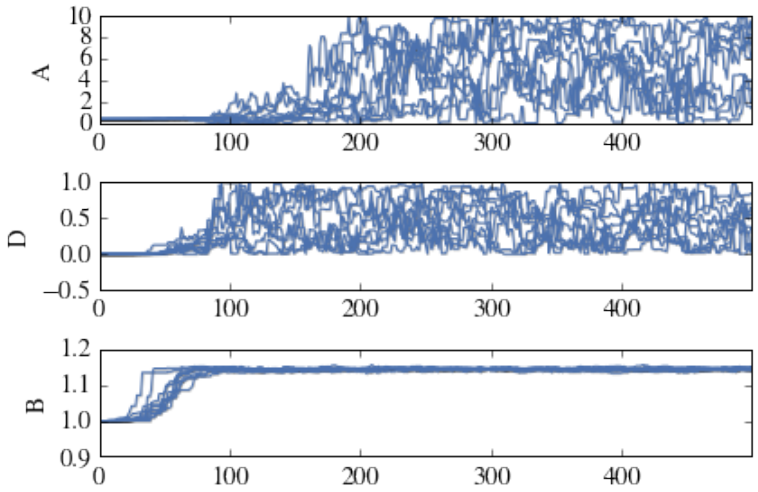


Figure 29: Trace of emcee walkers attempting to infer a value that is smaller than their step size. Because of this, they quickly move away from the region with the most likely value, and obtain results that do not make physical sense.

	A	B	D
0.16	1.160298	1.140638	0.165869
0.50	4.600467	1.144754	0.510908
0.84	8.344930	1.148496	0.850479

Figure 30: Flawed results from emcee inference for 100 nm gold particles. The value given for the diffusion coefficient results in a radius that is several orders of magnitude smaller than an electron.

It is obvious from this result that all the variables to be inferred by emcee must be of similar orders of magnitude. To force this to be the case, we rescaled the variable being inferred. To do so, we took advantage of the fact that the exponential decay constant can be written as

$$2Dq^2 = 2\frac{D}{\lambda^2}(4\pi n \sin\left(\frac{\theta}{2}\right))^2 \tag{49}$$

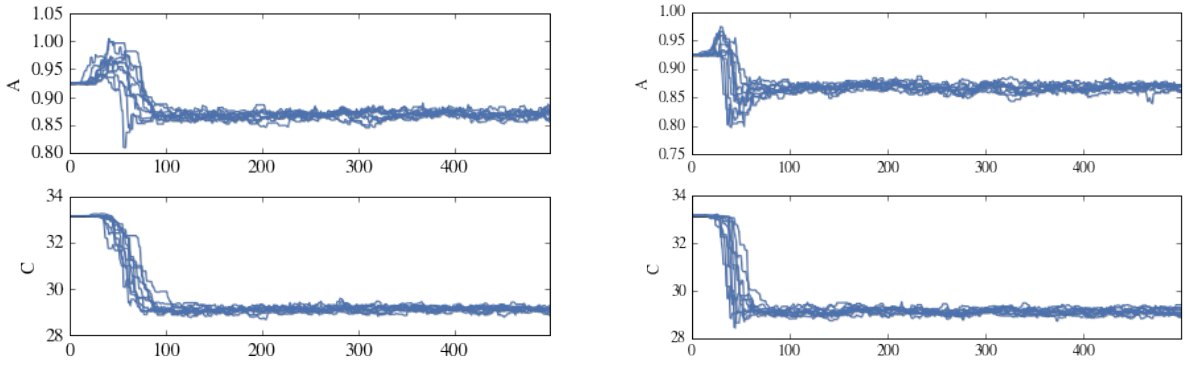
where

$$\frac{D}{\lambda^2} = \frac{4.35 \times 10^{-12}}{(658 \times 10^{-9})^2} = 10.047\,\mathrm{s}^{-1} \tag{50}$$

for this experiment. Even with larger or smaller particles, this fraction generally gives a value of the order of 10, making it a good stand in for inferring with emcee.

6.2 Effect of Priors on Enzyme Data

Next, we investigated the effect of the choice of priors on the Bayesian inference. Previously, we concluded that the choice of prior had minimal effect if there were sufficient data. To determine if this is the case, the same enzyme data used in the previous chapter were analyzed. Because there is only one angle in this data, the angle dependence is not included in the model. Instead, the value $C = 2Dq^2$ was inferred, and used to solve for the diffusion coefficient D. These data were analyzed using two typical choices for prior probabilities. First, a uniform, bounded prior was used. This unassuming prior only required that the parameters all be positive numbers. The initial positions of the walkers were set based on the least squares fit done in Chapter 4. Second, this analysis was repeated using a much less conservative prior: a sharply peaked Gaussian set to peak at $A = 0.92$ and $C = 33.0\,\mathrm{s}^{-1}$.



(a) Uniform Prior

(b) Gaussian Prior

Figure 31: The trace of the emcee walkers using both a uniform prior and a strongly assuming Gaussian prior. While both start in the position found by the least squares fit, the Gaussian prior strongly assumes the values are $A = 0.92$ and $C = 33.0\,\mathrm{s}^{-1}$. However, both sets quickly find the most likely values to be very similar.

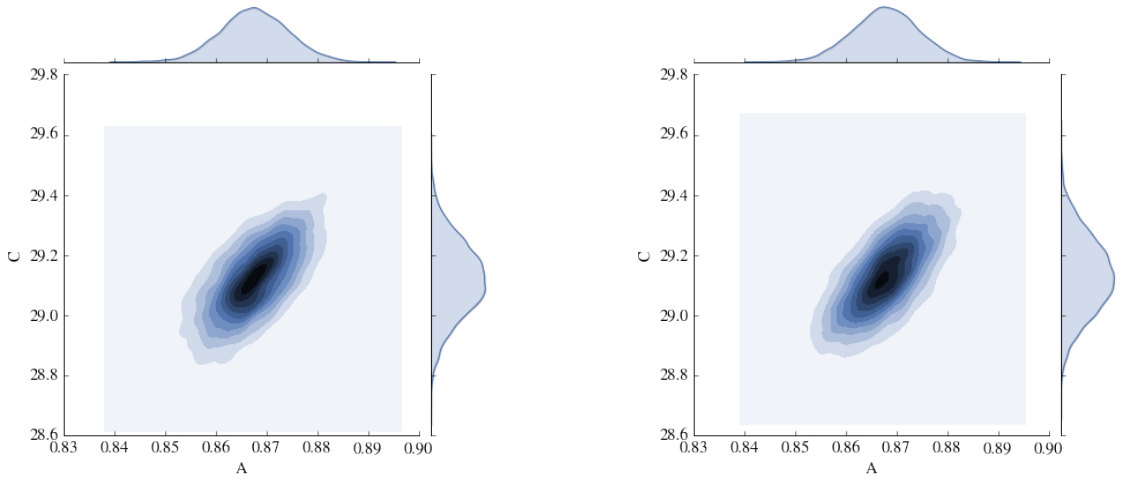
	A	C
0.16	0.861061	28.996831
0.50	0.867542	29.119737
0.84	0.874209	29.245621

(a) Uniform Prior

	A	C
0.16	0.861312	29.004532
0.50	0.867912	29.130150
0.84	0.874319	29.260970

(b) Gaussian Prior

Figure 32: The inferred most likely values using both a uniform prior and a strongly assuming Gaussian prior, where C has units of s^{-1}. The percent difference between the results is negligible, and the width of uncertainty is largely unaffected.



(a) Uniform Prior

(b) Gaussian Prior

Figure 33: The probability surface mapped by the walkers for both a uniform and Gaussian prior. The resulting posterior probability, shown here, is largely unaffected by the choice of priors.

The results of the inference using this prior are shown in Figures 31, 32, 33. Although the walkers begin far from the inferred values, and are strongly encouraged by the prior to stay in this region, both models find the same values after around 100 steps. The inferred values of A and C are very similar; the percent difference between the values found with each prior was 0.012% for the amplitude and 0.068% for the value of C. More importantly, there is not a significant change in the uncertainty of these values. The posterior probability surfaces shown in Figure 33 look very similar, and the confidence intervals are nearly identical. Clearly, even this 30 second measurement at one scattering angle provided sufficient data to overcome the strong influence of the choice of prior.

The values found through these two methods are compared to the values found in Chapter 4 in the table below. Unfortunately, because the data were collected from a solution of enzymes created for biological research instead of lab manufactured surfactants or polystyrene beads, there is no known hydrodynamic radius. All the values presented below are reasonable inferences. This analysis shows that for single angle data, the developed code gives comparable and reasonable results, regardless of the choice of prior.

Method	Calculated Radius	Sum of Residuals Squared
Single Exponential	4.746 ± 0.030 nm	0.011389
Second Order Cumulant	4.480 ± 0.027 nm	0.010662
Third Order Cumulant	4.261 ± 0.049 nm	0.009607
Bayesian- Uniform Prior	5.901 ± 0.025 nm	0.08329
Bayesian- Gaussian Prior	5.899 ± 0.025 nm	0.08198

6.3 Bayesian Inference of Multiangle Data

6.3.1 Single Exponential

The real strength of our Bayesian inference method for DLS analysis is the potential to determine the constant D across multiple angles simultaneously, rather than fitting to each angle individually. The code developed for this is listed in Appendix A.4. First, it was necessary to construct arrays of all the data across angles to put the parameters into a format that could be input into emcee. These included an array of all the times τ, the output autocorrelation functions g, the uncertainty of the autocorrelation values σ_g, and the angle ϕ. For a given position [i] in these arrays, the values of each describes the resulting output g taken at set time and angle.

Because the data are collected at several scattering angles, the autocorrelation function g is now a function of ϕ as well as τ. The likelihood function then is

$$\ln \mathcal{L} = K - \sum_{i=1}^{N} \frac{[g_i - A \exp\Big\{-2D(4\pi n \sin\Big(\frac{\phi}{2}\Big))^2\Big\} - B]^2}{2\sigma_{g_i}^2} \tag{51}$$

where D here is equal to $\frac{D}{\lambda^2}$ to make the value the same order of magnitude as B and A. Using Bayes' Theorem, the posterior probability becomes

$$\ln P(A, B, D|g(\tau, \phi) \propto \ln P(g(\tau, \phi)|A, B, D) + \ln P(A, B, D) \tag{52}$$

In order to fully test the accuracy of the inference method developed in the previous chapter, it was necessary to use a data set with minimal noise and a well-defined hydrodynamic radius. Although we faced several issues in the data collection process, outlined in Chapter 3, the autocorrelation of the intensity of light scattered by 40 nm diameter polystyrene beads across 11 angles, shown in Figure 20, will provide a good test for the Bayesian inference method. While there still exists some noise at short τ, the amplitude of the autocorrelation function is nearly constant across all angles. Further, the solution appears to be strongly monodisperse, without signs of aggregation. This can be seen in Figure 21, which shows the values for D determined by least squares fit method plotted against the value q^2. For completely monodisperse solutions, this relationship should be perfectly linear. Although there is some deviation from the best fit line (in green), the data agree with this trend remarkably well. Overall, the solution will act as an adequate calibration tool in order to compare the traditional method with this developed Bayesian inference model.

Although the intensity of the light scattered by the 40 nm polystyrene beads does not have a strong angle dependence, it is not necessarily the case that the autocorrelation function must have the same amplitude

across angles. For larger particles especially, the amplitude of the signal is strongly dependent on the angle. Because of this, it was necessary to include a separate parameter for the amplitude at each angle. To allow for this, the code in Appendix 4 defines the amplitude A as an array of amplitudes for each angle. This allows for a more accurate fit, as the amplitude is not artificially constrained. Further, because these variables will be marginalized out in the determination of the diffusion coefficient, they are not essential. Because of the increase in parameter space, the run time of the algorithm increased from around 5 seconds to around 10 minutes. This increase was expected, and the run time is still remarkably short given the increase from 3 dimensions to 14 dimensions increases the number of points sampled in parameter space by 11 orders of magnitude. The results of the walkers' exploration is shown below in Figures 34 and 35. Because of the large parameter space, the number of steps taken by each walker was increased from 500 to 1500. Though it does take longer for the walkers to settle, especially in the case of D, the walkers take more than enough steps to fully map the parameter space.

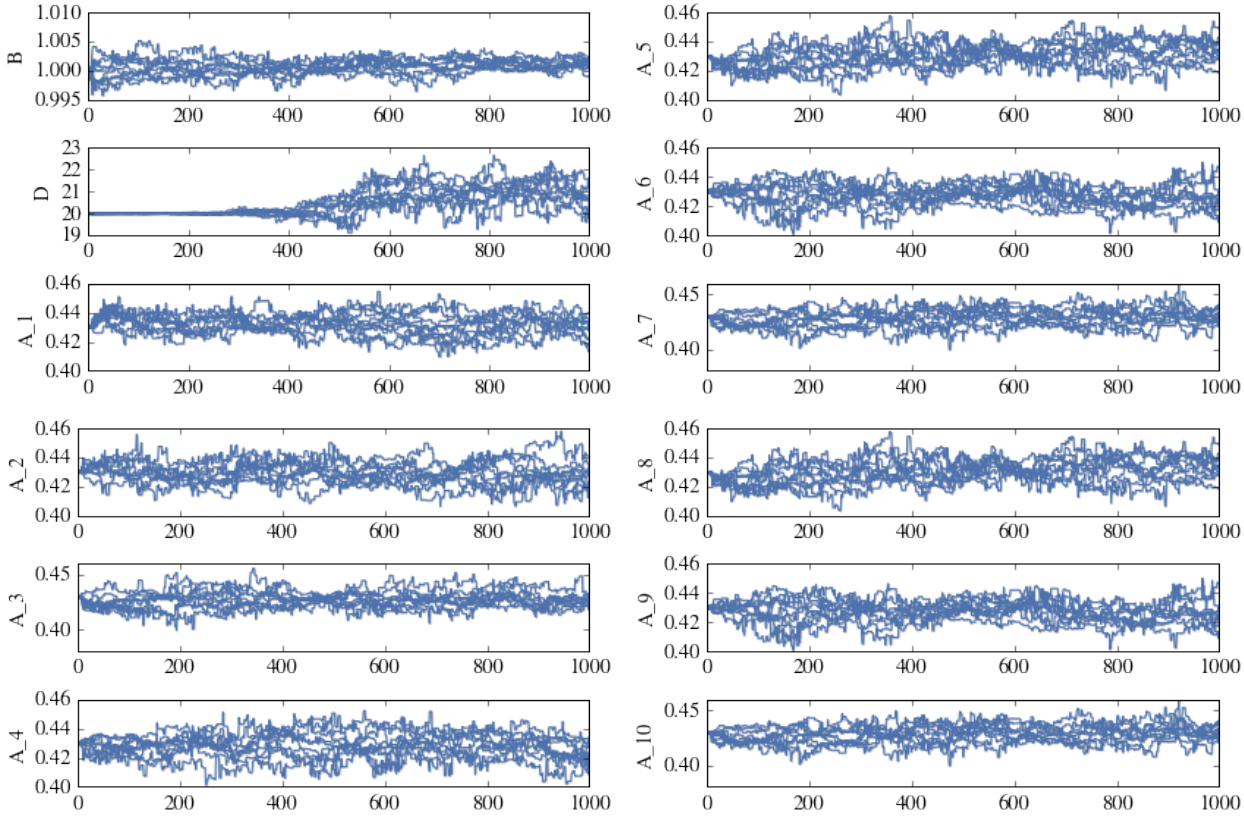


Figure 34: Trace of emcee walkers attempting to infer values for a baseline, diffusion coefficient, and angle dependent amplitudes. Although it takes longer for the variables to equilibrate, they fully explore the parameter space after 1000 steps.

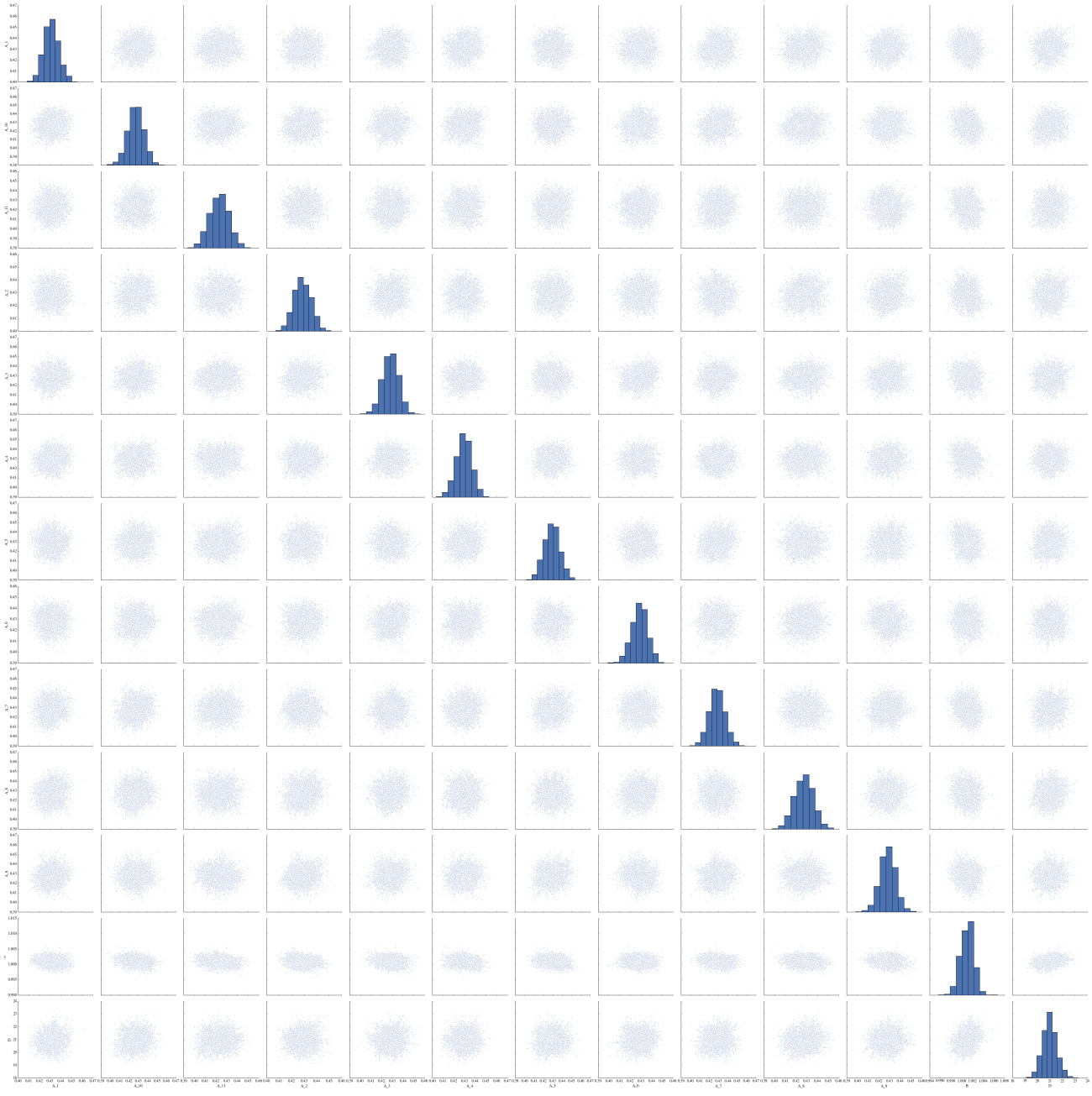


Figure 35: Pair plots for each variable inferred by emcee for the 40 nm polystyrene bead data. Each plot shows the probability surface of the variables in the given row and column, which gives the correlation between those two variables. Across the diagonal are the marginalized probability functions for each variable, which show the probability density function of only that variable. In general, the inferred variables are largely uncorrelated, and the resulting probability densities for each variable are roughly Gaussian in shape.

With such a large number of parameters being inferred, it can be difficult to understand the shape of the posterior probability surface. Pair plots, which show the correlation between each variable as well as their marginalized probability function, are helpful when viewing large trends. The pair plots for this inference are shown in Figure 35. Most of the plots show generally circular probability surfaces, indicating that the

variables are largely uncorrelated with each other. Further, each variable's marginalized probability function is generally Gaussian, which indicates that the values are well defined. These results are shown in more detail in Figure 37, where the pair plots are just between the amplitude associated with 15 degrees and the diffusion coefficient; these plots strongly indicate that the parameters are uncorrelated. Using the results in the figures above, the diffusion coefficient was found to be $D = 9.09 \pm 0.24 \times 10^{-12}\,\mathrm{m^2 s^{-1}}$, giving a hydrodynamic radius $R = 25.34 \pm 0.67\,\mathrm{nm}$.These results are compared to previous results in a table below.

	A_1	A_10	A_11	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	B	D
0.16	0.424766	0.417976	0.414756	0.422926	0.419802	0.420995	0.424096	0.417298	0.419804	0.420143	0.419224	0.999882	20.448155
0.50	0.431127	0.425729	0.422335	0.429760	0.426514	0.429208	0.432335	0.425229	0.427011	0.428886	0.427294	1.001048	20.996179
0.84	0.437888	0.433449	0.430440	0.437344	0.434466	0.436872	0.440217	0.433623	0.434656	0.437421	0.434953	1.002273	21.560523

Figure 36: The results of a Bayesian inference of the baseline, amplitude, and $\frac{D}{\lambda^2}$, shown along with confidence intervals.

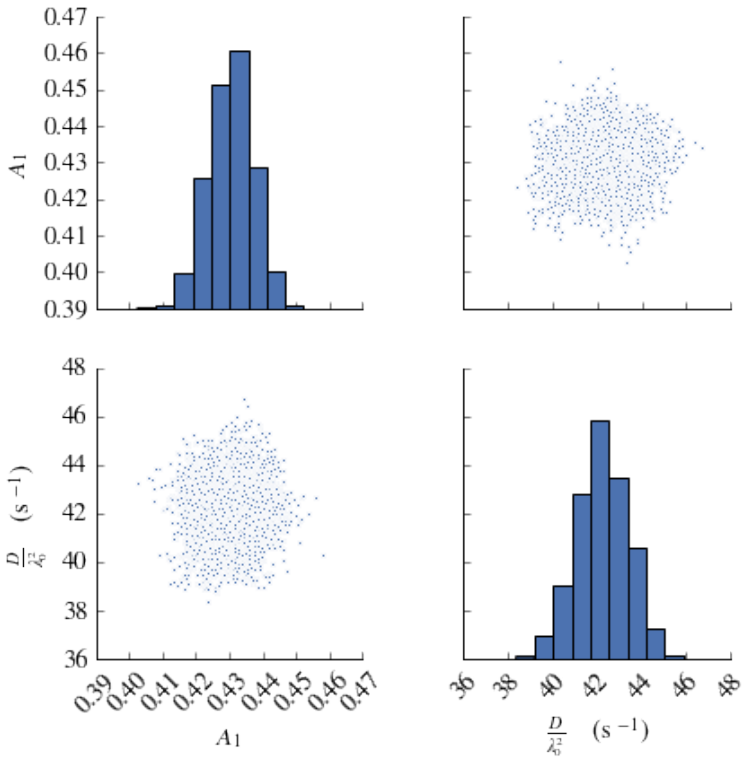


Figure 37: The marginalized probability function of one amplitude value and the value of $\frac{D}{\lambda^2}$. The value of $\frac{D}{\lambda^2}$ was found to be Gaussian; the width of the curve indicates confidence values.

6.3.2 Bayesian Method of Cumulants

The final step in the development of the Bayesian inference model was to extend code in Appendix A.4 to the second-order cumulant expansion; the code developed for this is listed in Appendix A.6. As the 40 nm polystyrene data was previously analyzed using the method of cumulants in Chapter 4, the results from each angle fit are not listed again. The process for including angle dependence in the values of μ_2 is the same process that was used to allow for angle dependence across the amplitude of the autocorrelation function. This increased the parameter space from 13 parameters to 24 parameters, which increased the run time of the code to 45 minutes; this is still a reasonable amount of time given the calculation size. Though the value of μ_2 varies across angles, the value for the polydispersity index should be constant if the size distribution is one Gaussian sharply peaked around a mean radius. The results found in Chapter 4 indicated that the polydispersity index was not constant, however, due to some source of noise in the data.

This result was also found when the Bayesian inference listed in Appendix A.6 was applied to the data. Although the peak of the probability function for the marginalized values of μ_2 are well defined, the uncertainty of the value is a large fraction of the determined value. Again, this value does not provide reliable insight into the polydispersity index of the sample, but it is consistent with the results given by the standard least squares fit.

The values for the polydispersity index are listed in the table below. While there was a large amount of uncertainty in the values found by the Bayesian fit, the results are very similar to those found by the nonlinear least squares fit. This consistency is a strong indication that the results of the Bayesian inference are reliable, despite the large amount of uncertainty. The inferred value for the hydrodynamic radius was $24.7 \pm 0.9\,\text{nm}$, which is compared to the other results in the next section.

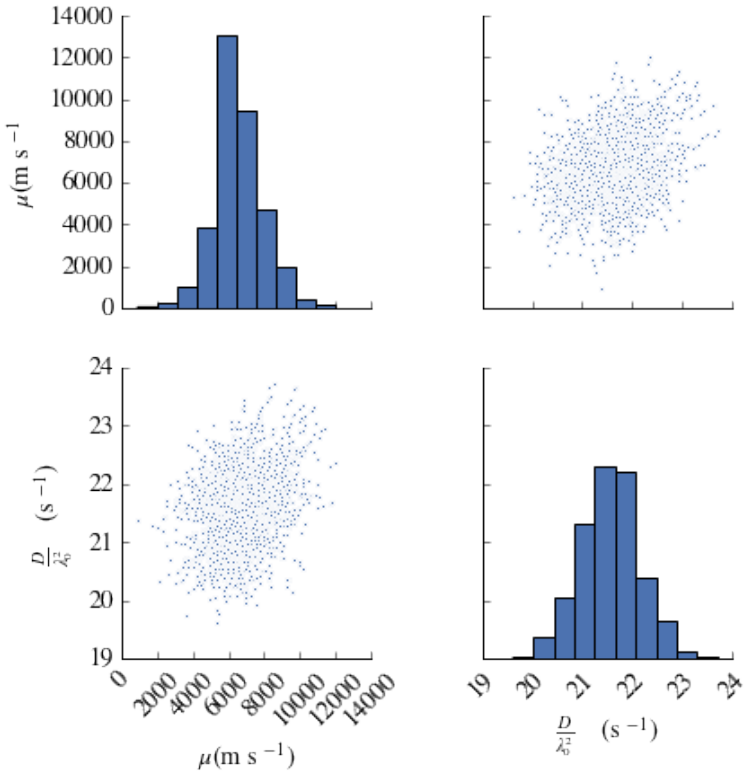


Figure 38: The marginalized probability function for μ_2 and D for 40 nm polystyrene beads, where μ_2 is the second order cumulant associated with the data taken at 15 degrees. Both variables are well defined, although there is large uncertainty on the value of μ_2.

Method	μ_2	$\frac{\mu_2}{\overline{\Gamma}^2}$
Least Squares Fit: 15 degrees	$4,673\,\mathrm{s}^{-2}$	0.4398
Least Squares Fit: 30 degrees	$42,917\,\mathrm{s}^{-2}$	0.2613
Least Squares Fit: 45 degrees	$152,274\,\mathrm{s}^{-2}$	0.1940
Least Squares Fit: 60 degrees	$335,955\,\mathrm{s}^{-2}$	0.1468
Least Squares Fit: 75 degrees	$556,785\,\mathrm{s}^{-2}$	0.1107
Least Squares Fit: 90 degrees	$910,645\,\mathrm{s}^{-2}$	0.0995
Least Squares Fit: 105 degrees	$1,296,316\,\mathrm{s}^{-2}$	0.0894
Least Squares Fit: 120 degrees	$1,697,905\,\mathrm{s}^{-2}$	0.0825
Least Squares Fit: 135 degrees	$2,471,328\,\mathrm{s}^{-2}$	0.0927
Least Squares Fit: 150 degrees	$1,860,505\,\mathrm{s}^{-2}$	0.0584
Least Squares Fit: 165 degrees	$2,024,290\,\mathrm{s}^{-2}$	0.0572

Finally, another inference model was developed in order to take advantage of another value that is theoretically constant across the angle of measurement: the polydispersity index (PDI). The polydispersity index is a measure of the width of the distribution of the radii around the center peak, using the method of cumulants. As derived in Chapter 2, this value is

$$PDI = \frac{\mu_2}{\overline{\Gamma}^2} \tag{53}$$

where μ_2 is the second order cumulant and $\overline{\Gamma}$ is the average decay constant. Because μ_2 varies by orders of magnitude across angle, fitting to a constant value rather than indivudal values of μ_2 would greatly improve the efficiency of the code, as well as decrease the computation time. To do so, we began with the cumulants expansion of the autocorrelation function of the intensity, as derived in Chapter 2

$$g^{(2)} = B + Ae^{-2\overline{\Gamma}\tau}(1 + \frac{\mu_2}{2!}\tau^2)^2 \tag{54}$$

and factored out the value we defined as the polydispersity index.

$$g^{(2)} = B + Ae^{-2\overline{\Gamma}\tau}(1 + \frac{\mu_2}{\overline{\Gamma}^2}\frac{\overline{\Gamma}^2}{\mu_2}\frac{\mu_2}{2!}\tau^2)^2 \tag{55}$$

This results in an equation that is independent of μ_2, and thus only requires inference of the variables B, $\overline{\Gamma}$,

	A_1	A_10	A_11	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	B	D	P
0.16	0.425545	0.416855	0.415303	0.423706	0.421138	0.423033	0.423727	0.420615	0.421113	0.420465	0.419979	0.999382	20.940468	0.053278
0.50	0.432007	0.425555	0.423335	0.430924	0.429321	0.430552	0.431273	0.429215	0.429388	0.429193	0.428388	1.000605	21.830162	0.139900
0.84	0.438646	0.434731	0.431185	0.438692	0.436538	0.438525	0.439023	0.436972	0.437865	0.437025	0.436504	1.001836	22.776572	0.251222

Figure 39: The results of a Bayesian inference of the baseline, amplitude, polydispersity index, and $\frac{D}{\lambda^2}$, shown along with confidence intervals.

PDI, and the angle dependent amplitudes A_i.

$$g^{(2)} = B + Ae^{-2\overline{\Gamma}\tau}(1 + P\frac{\overline{\Gamma}^2}{2!}\tau^2)^2 \tag{56}$$

The code developed based on this equation is listed in Appendix 6. By decreasing the number of parameters being inferred from 24 back to only 13, we were able to develop a Bayesian approach to the method of cumulants while still maintaining a computational time on the order of 10 minutes, rather than 45. The results of that code are given below, in Figures 39 and 40. While the inferred value for D is well defined, and the probability distribution is roughly Gaussian, the value for the polydispersity index is strongly influenced by the boundaries placed on the prior. Because a negative polydispersity index is not physically meaningful, the prior probability only allowed positive values for this index. This causes a sharp boundary on the posterior probability, as shown in the marginalized posterior probability of the polydispersity index, in the upper left hand corner of Figure 40. Despite this, the algorithm was able to determine confidence intervals for the inferred value of the PDI. The posterior probability shown in Figure 40 also indicates that the values for the diffusion coefficient and the polydispersity index are highly correlated, which is expected given that the value of PDI was defined as being proportional to the value of $\overline{\Gamma}$.

That inferred value was found to be $PDI = 0.140 \pm 0.014$. The manufacturer of the polystyrene beads expected a polydispersity index of $PDI = 0.15$, so this inferred value is within the expected result [24]. Unlike the least square fit results, which had a strong dependence on angle (as discussed in Chapter 4) this inference method is able to the data as a whole, providing a more reliable inferred value. The results for the radius will be discussed in the following section.

6.3.3 Comparison of Inference Methods

The results from the four methods of analysis performed on the calibration sample of 40 nm polystyrene beads are listed in the table below. The expected value for the radius of the beads was 20 nm. While none of the methods inferred this value, the expected standard deviation of the size of the bead is is 6 nm; because of this, all of these values are within the error expected by the producer of the particles [24]. Thus, these fits

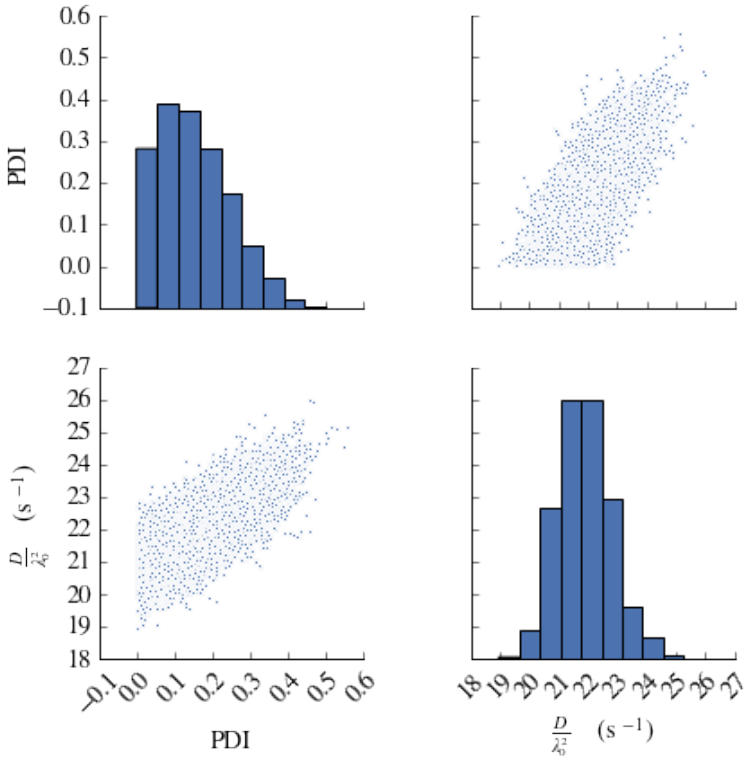


Figure 40: The marginalized probability function for the polydispersity index PDI and D for 40 nm polystyrene beads. While the inferred value for D is well defined, and the probability distribution is roughly Gaussian, the value for the polydispersity index is strongly influenced by the boundaries placed on the prior, which causes a sharp boundary on the allow probability.

are ultimately inconclusive as to whether the Bayesian inference approach leads to a more accurate value for the hydrodynamic radius.

Method	Diffusion Coefficient	Calculated Radius
Least Squares Fit	$9.34 \pm 0.04 \times 10^{-12}\,\mathrm{m^2 s^{-1}}$	$24.66 \pm 0.10\,\mathrm{nm}$
Bayesian Inference: Single Exponential	$9.09 \pm 0.24 \times 10^{-12}\,\mathrm{m^2 s^{-1}}$	$25.34 \pm 0.67\,\mathrm{nm}$
Bayesian Inference: Method of Cumulants	$9.34 \pm 0.35 \times 10^{-12}\,\mathrm{m^2 s^{-1}}$	$24.68 \pm 0.94\,\mathrm{nm}$
Bayesian Inference: Inferring PDI	$9.45 \pm 0.40 \times 10^{-12}\,\mathrm{m^2 s^{-1}}$	$24.37 \pm 1.02\,\mathrm{nm}$

While this calibration does not offer proof that the Bayesian inference method is more accurate, it does offer a proof of concept. Because the results from the Bayesian inference have been found to be repeatedly comparable to the results given by the standard least squares fit, we have shown that Bayesian inference can be reliably applied to both single and multi-angle DLS data. In the following sections, the algorithm will be applied to more complex DLS data taken in the Manoharan lab, and the results will be discussed.

6.4 Analysis of 100 nm Gold Particles

The sample analyzed below was prepared by Nabila Tanjeem of 100 nm diameter gold particles suspended in water. The gold nanoparticles were purchased from Ted Pella, and have a coefficient of variation of 8%; this means that the hydrodynamic radius of 68 % of the particles should fall between 46 nm and 54 nm [33]. The sample was analyzed with the DLS instrument in the Manoharan Lab, described in Chapter 3. The temperature of the sample was set to 20 degrees Celsius, and the data was taken for 120 seconds. The resulting autocorrelation functions, shown in Figure 41, do not indicate the noise issue encountered with smaller scatterers due to the dust inside the instrument, but they do have large amounts of noise at smaller τ values.

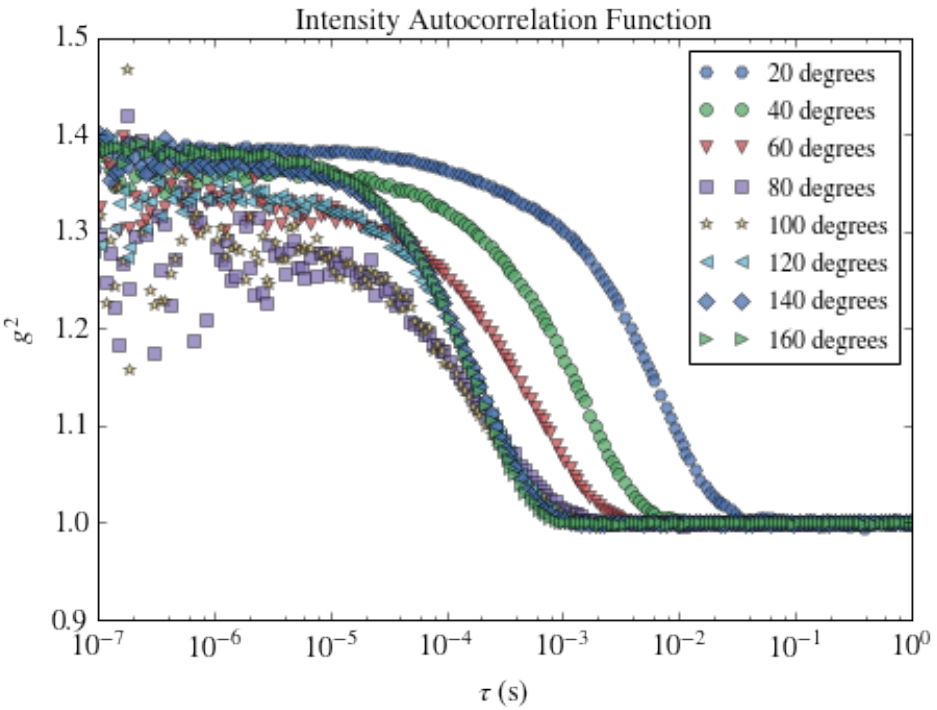


Figure 41: Intensity autocorrelation function of light scattered by a solution of 100 nm gold particles suspended in water, taken across 8 angles.

The fits were calculated using the code in Appendix A.1, with a nonlinear least squares fit to a single exponential decay applied to each autocorrelation function individually. The results of these fits are shown in Figure 42. The residuals are consistently high at the low τ. There is also the repeated feature of high residuals in the center of the decay for small angles; this indicates the possibility of aggregation in the sample, as it indicates that the data are not well described by the theoretical fit for monodisperse samples. Since larger particles scatter more strongly in the forward direction, this would explain why that feature of the residuals falls off as the scattering angle increases. Another feature to note across this data is the amplitude decrease for angles near 90 degrees. This is due to the fact that this data were taken before the laser was realigned and the polarization was corrected. Most likely, the light transmitted by the fiber was horizontally polarized, leading to a loss of signal near 90 degrees. Despite this decrease in signal, the count rate was still high enough for the data to be usable.

To see deviation from the theoretical model of scattering by a completely monodisperse solution, it is helpful to plot the decay coefficient Γ against the value of the scattering vector squared, q^2. If the solution is perfectly monodisperse, this relationship should be linear, with a slope equal to $2D$. The results of this plot can be seen in Figure 43. While there is a general linear trend, there are obvious deviations from the best-fit line. This deviation provides more strong evidence that the solution is not monodisperse, but rather has

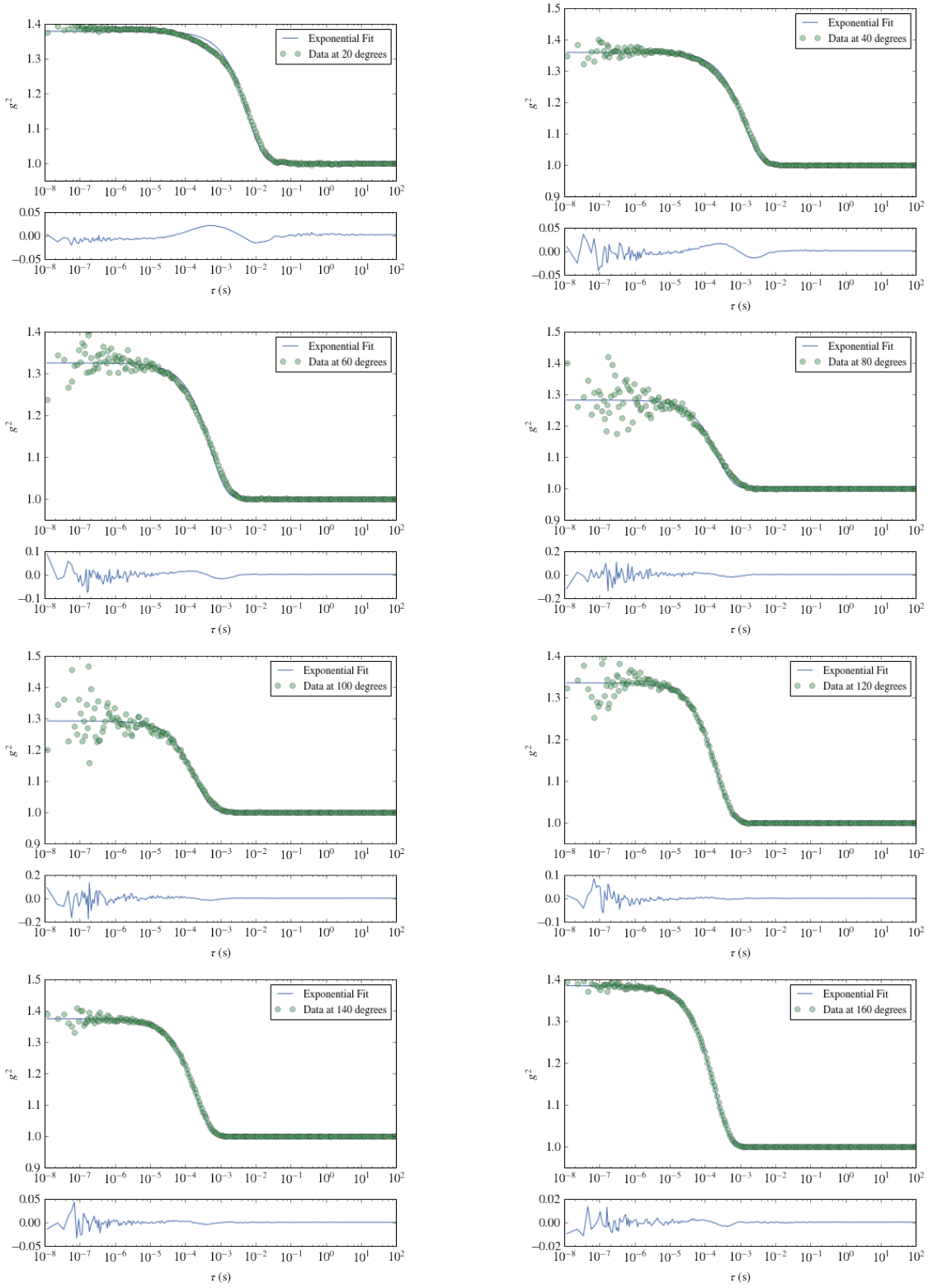


Figure 42: Single exponential fits to each angle of the gold particle data, with residuals graphed below.

some polydispersity. This is not completely unexpected, as gold nanoparticles can be prone to aggregation, and the sample analyzed had a large amount of time to settle before the measurements were taken. Using the value for the diffusion coefficient listed in Figure 43, the radius was calculated to be $49.3 \pm 8.6\,\text{nm}$. The large uncertainty results from the obvious polydispersity of the particles.

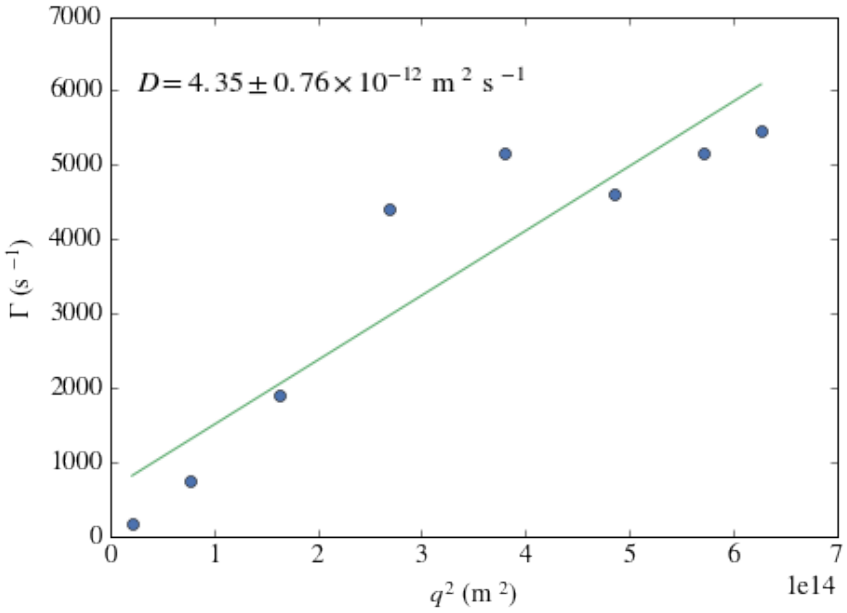


Figure 43: For each fit, the calculated decay coefficient Γ is plotted against the value of q^2. The slope of this relationship gives the diffusion coefficient D. Deviation from the linear relationship indicates that the data is not completely described by the single exponential model, and is not perfectly monodisperse.

Next, the data of 100 nm gold particles were analyzed using the algorithm developed for Bayesian inference. The initial positions of the walkers were determined by the least squares fit analysis. The results are shown in Figure 44; the lack of skew in most of the pair plots indicates that the variables are highly uncorrelated. The posterior probability for each variable is again roughly Gaussian, giving a clear estimation for the uncertainty of the values. This can be seen more clearly in the pair plot just showing the values of the first amplitudes, at 20 degrees, and the diffusion coefficient in Figure 45. Using the most likely value of the diffusion coefficient, $D = 5.1 \pm 0.2 \times 10^{-12}\,\text{m}^2\text{s}^{-1}$, the radius was found to be $42.1 \pm 1.8\,\text{nm}$. These results are compared to the least squares fit in the table below

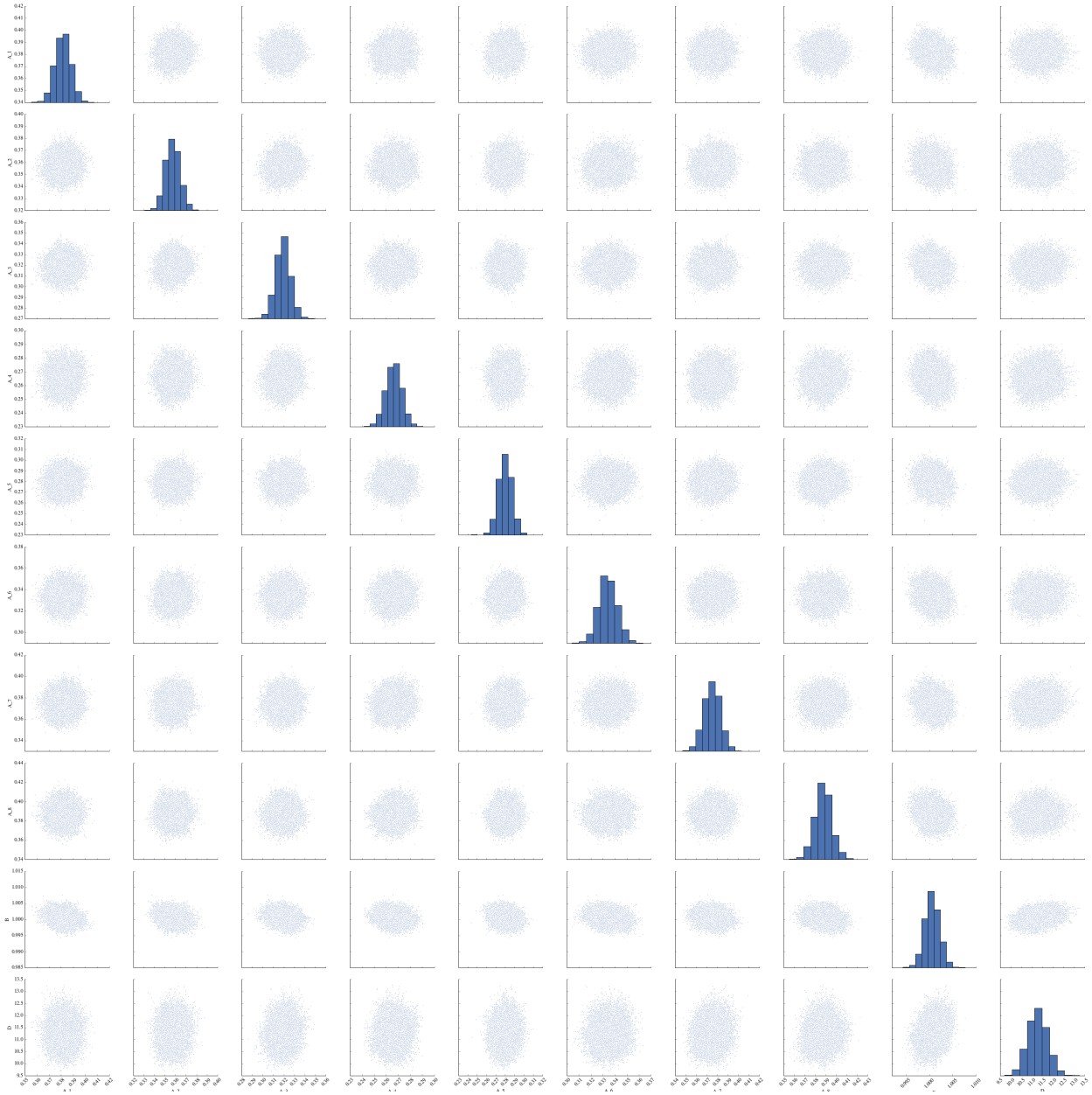


Figure 44: Pair plots for each variable inferred by emcee for the 100 nm gold data. Each plot shows the probability surface of the variables in the given row and column, which gives the correlation between those two variables. Across the diagonal are the marginalized probability functions for each variable, which show the probability density function of only that variable. In general, the inferred variables are largely uncorrelated, and the resulting probability densities for each variable are roughly Gaussian in shape.

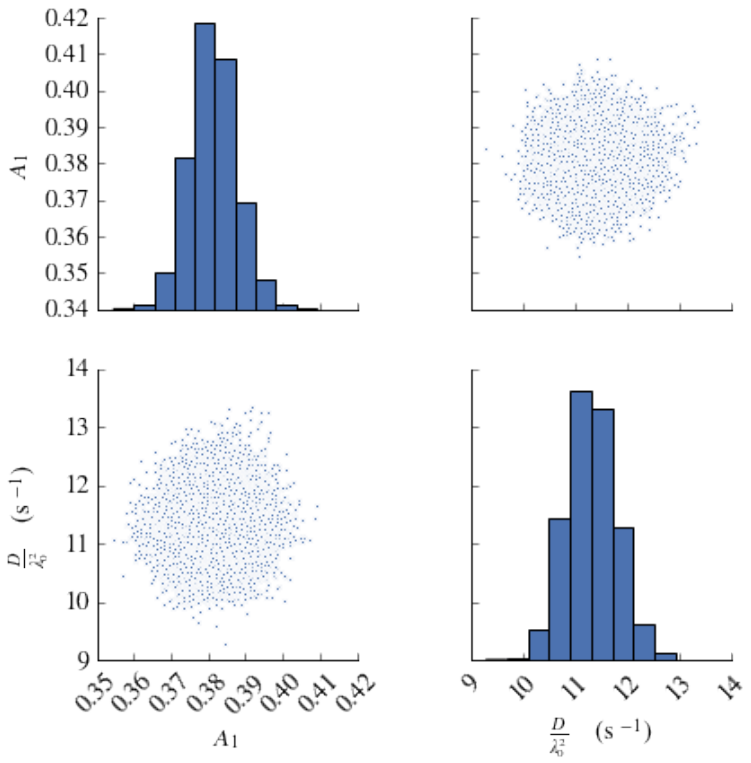


Figure 45: The diffusion coefficient and one amplitude are shown, and marginalized. The variables are highly uncorrelated, and have a strongly Gaussian distribution

Method	Calculated Radius
Least Squares Fit	49.3 ± 8.6 nm
Bayesian Inference	43.8 ± 1.9 nm

The table above lists the values and uncertainties found using the standard method of analysis and the developed model. Though the particles were intended to have a hydrodynamic radius of 50 nm, the data collected did not well match the model of a single exponential decay, most likely due to aggregation within the particles. Although the least squares fit did give the value closest to the anticipated value, it also resulted in a high amount of uncertainty. Including angle dependence in the amplitude of each autocorrelation function did not significantly change the uncertainty of the inferred value, but it did change the inferred radius. Because the data are ill-described by the model, these results are again not conclusive as to accuracy of the Bayesian inference method; the inferred values are reasonable, however.

Finally, this code was expanded to include a second-order cumulant expansion. The code developed is listed in Appendix 6, and applied here to the same set of 100 nm gold particles. As this data was not previously analyzed using the method of cumulants, the results from each angle fit are listed in a table below. The noise in the data presented issues in this analysis; because of the sensitivity of the cumulants expansion,

there was a high amount of variance across the angles. Theoretically, the value fit for the polydispersity index, or $\frac{\mu_2}{\bar{\Gamma}^2}$, should be constant across angles. The variance found in the least squares fit indicates an instability in the data analyzed, as the values for the polydispersity index do not remain constant across angles, but instead range by a considerable amount. The inferred value of μ_2 found by the Bayesian inference was consistent with this large uncertainty. These values are listed in the table below.

Method	$\mu_2\,(\mathrm{s}^{-1})$	$\frac{\mu_2}{\bar{\Gamma}^2}$	Calculated Radius
Least Squares Fit: 20 degrees	7,720	0.6882	$42.5 \pm 0.8\,\mathrm{nm}$
Least Squares Fit: 40 degrees	129,096	0.6087	$35.7 \pm 0.8\,\mathrm{nm}$
Least Squares Fit: 60 degrees	1,191,433	0.7610	$27.9 \pm 1.3\,\mathrm{nm}$
Least Squares Fit: 80 degrees	6,844,043	0.7970	$19.6 \pm 2.0\,\mathrm{nm}$
Least Squares Fit: 100 degrees	7,413,197	0.6916	$24.9 \pm 3.8\,\mathrm{nm}$
Least Squares Fit: 120 degrees	637,558	0.1135	$4.39 \pm 1.9\,\mathrm{nm}$
Least Squares Fit: 140 degrees	994,621	0.1388	$45.7 \pm 0.8\,\mathrm{nm}$
Least Squares Fit: 160 degrees	816,326	0.1038	$47.9 \pm 0.3\,\mathrm{nm}$

The results from the Bayesian inference approach are shown in Figure 46. The amplitude and diffusion remain largely unchanged, but the value for μ_2 showed a huge amount of uncertainty; this is especially noticeable when the value for μ_2 was inferred to be negative, which was not physically possible, as it would indicate a negative width. To account for this, it would be possible to further constrain the prior to only allow for positive values of μ_2. For this inference, however, the large amount of uncertainty across the values was not corrected by a more informed prior. This uncertainty can be seen more clearly in the marginalized plot of μ_2 for the first angle in Figure 47. The posterior probability function of the variable is essentially bimodal, which accounts for the huge amount of uncertainty in the value. Despite this uncertainty, which is also shown in the table of least squares fit values, the inference of the diffusion coefficient remains largely the same, which is an indication of the strength of the inference method.

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	B	D
0.16	0.378672	0.354287	0.312857	0.263125	0.277272	0.335250	0.372566	0.389289	0.998212	13.791591
0.50	0.386127	0.362985	0.320613	0.270844	0.286181	0.344075	0.381966	0.399079	1.000191	16.766440
0.84	0.393740	0.371816	0.330457	0.277746	0.294984	0.353318	0.391580	0.409499	1.002282	17.886795

mu_2_1	mu_2_2	mu_2_3	mu_2_4	mu_2_5	mu_2_6	mu_2_7	mu_2_8
13737.714901	118922.769800	642491.375213	1144933.288658	2380189.558436	-6011824.813046	2600620.128213	-3832588.920131
26177.173032	279480.562668	1510867.835698	4169269.297360	4924886.472546	-699414.445683	3445858.772003	-270424.157773
33923.809184	371898.625761	1910132.575442	5391437.173257	5985747.613988	1031872.740663	4372019.806691	913338.865047

Figure 46: The results from inference using second order cumulant expansion applied to 100 nm gold particles. There is a large amount of uncertainty in the values of μ_2.

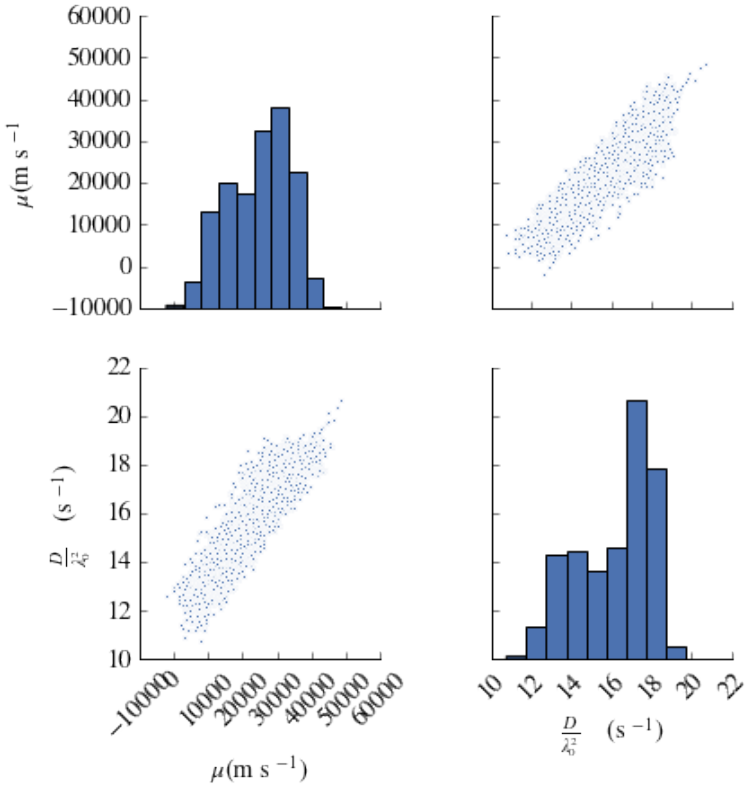


Figure 47: The diffusion coefficient and first cumulant expansion μ_2 for the gold particles are shown, and marginalized. The variables are slightly correlated, but there is a large amount of uncertainty in the value inferred for μ_2.

Method	$\mu_2\,\mathrm{s}^{-1}$	$\frac{\mu_2}{\overline{\Gamma}^2}$
Bayesian Inference: 20 degrees	26,177	2.31
Bayesian Inference: 40 degrees	279,480	0.17
Bayesian Inference: 60 degrees	1,510,867	2.22
Bayesian Inference: 80 degrees	4,169,269	2.26
Bayesian Inference: 100 degrees	4,924,886	1.33
Bayesian Inference: 120 degrees	-699,414	-0.16
Bayesian Inference: 140 degrees	3,445,858	0.41
Bayesian Inference: 160 degrees	-270, 424	-0.03

Finally, the code developed for Bayesian inference using the polydispersity index was applied to the data of the gold particles, in order to confine the uncertainty due to the variance in the value of μ_2. The results of the inference are shown below in Figures 48 and 49. The inferred value of the polydispersity index was found to be $PDI = 0.50 \pm 0.07$, which is high enough that the sample cannot be considered strongly monodisperse. This is not unexpected, given the other indications that the gold particles have aggregated, and can no longer be considered monodisperse.

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	B	D	P
0.16	0.379035	0.354285	0.314785	0.262787	0.276163	0.330079	0.372866	0.382869	0.998145	12.509324	0.340740
0.50	0.385031	0.361283	0.322121	0.269864	0.283253	0.338192	0.380220	0.390711	0.999719	13.457800	0.494837
0.84	0.391564	0.368065	0.329375	0.276607	0.290370	0.345515	0.387653	0.398798	1.001288	14.500907	0.628454

Figure 48: The results from inference using second order cumulant expansion while fitting to the polydispersity index, as applied to 100 nm gold particles. Although there is a large amount of uncertainty in the inferred value of the polydispersity index, it has been confined to a physically meaningful result.

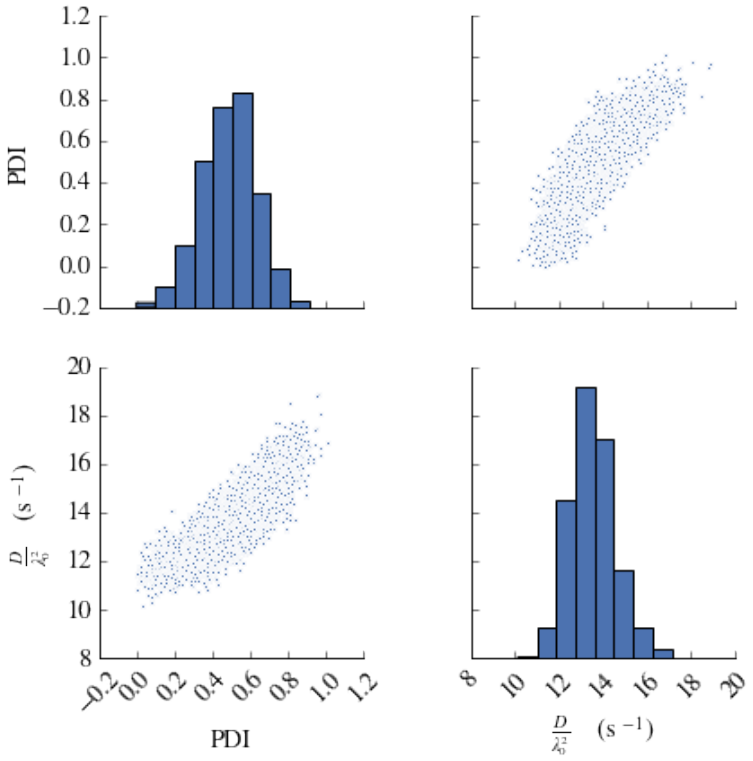


Figure 49: The inferred values for diffusion coefficient and polydispersity index for the gold particles are shown, and marginalized. The variables are highly correlated, but have well defined confidence intervals.

The results of these different methods of analysis are shown in the table below. While there is a large amount of variation in the inferred radii, this does not necessarily indicate that the Bayesian inference method has failed. Because of the strong indications that the sample cannot be considered monodisperse, due the presence of aggregation, it is understandable that the models infer a range of values. The model does not well describe the data; thus, the results produced by the methods below are not necessarily good descriptions of the size distribution.

Method	Calculated Radius
Least Squares Fit	49.3 ± 8.6 nm
Bayesian Inference: Single Exponential	43.8 ± 1.9 nm
Bayesian Inference: Method of Cumulants	42.3 ± 2.0 nm
Bayesian Inference: Inferring PDI	36.8 ± 2.7 nm

In this chapter, the process for developing a Bayesian inference algorithm used emcee was detailed, and that algorithm was applied. Although the analysis cannot conclusively say that the Bayesian inference model is more accurate or more precise, there are several signs of success. First, the developed model produced

comparable results to the least squares fit when applied to the monodisperse solution of 40 nm polystyrene beads. Second, the model produced the same large uncertainties when applied to data that were not well described by the model, for the aggregated 100 nm gold particles. These both indicate the reliability of the results given by the developed Bayesian inference algorithm.

7 Conclusion

The final chapter of this thesis will discuss the computational method developed in the previous chapter, along with the potential sources of uncertainty. Finally, the implications of the model will be discussed, as well as future applications of this project.

7.1 Results

Although this thesis cannot state conclusively whether Bayesian inference offers a more accurate method of determining the size distribution of solutions using DLS data, it does offer a proof of concept. We were able to apply a Bayesian inference method to DLS analysis for single angle and multi-angle data using both a single exponential and method of cumulants model. Because the results from the Bayesian inference have been found to be repeatedly comparable to the results given by the standard least squares fit, we have shown that Bayesian inference can be reliably applied to both single and multi-angle DLS data.

Even though the developed algorithm did not produce an increase in accuracy over the traditional least squares fit method, it did offer additional information about the probability distribution. The marginalized probability functions inferred by emcee provided not only the most likely value for the parameter, but also the width and shape of that probability function, as well as the correlation between the inferred values. This was especially revealing when the data were poorly described by the model. When applying the method of cumulants to the highly polydisperse gold nanoparticles in Chapter 6, the Bayesian inference revealed that the resulting probability distributions for several parameters, including the second order cumulant, were bimodal; thus, the particles could not be well described by the method of cumulants, as there were multiple likely values for μ_2.

While experimental difficulties with laser alignment, sample preparation, and dust contamination limited the amount of data available for analysis in the scope of this thesis, the open source nature of the algorithms developed would easily facilitate further testing with other data sets. With more data, it will become clear whether the Bayesian inference method offers an increase in accuracy or precision when compared to traditional least squares fit methods. A large amount of uncertainty in this thesis is the limited amount of data suitable for the comparison of the data techniques, as one data set was of an unknown particle size and one was not well described by the model. In the future, more calibration data is needed to come to a conclusion about the reliability and accuracy of the developed model.

7.2 Future Work and Applications

In the course of this thesis, single and multi-angle DLS data were analyzed using both a single exponential decays and the method of cumulants. These methods work well when the system is monodisperse, or when there is a well defined spread around a mean. When the suspension is made up of a spread of particles with multiple peaks, however, both methods fail. The standard method for these populations is a constrained regularization algorithm called CONTIN, which discards solutions that it considers unlikely [21]. CONTIN infers the size distribution of a DLS sample by performing an inverse Laplace transform of the measured correlation function. Because of this, the inference of the size distribution from the autocorrelation function is an ill-conditioned problem. In general, there are an infinite number of possible solutions, and the solution produced by CONTIN is highly sensitive to small amounts of noise. Even a small change in data can generate a large change in the result. To combat this, CONTIN constrains the size distribution, either by bounding the possible values of Γ or by making assumptions about the makeup of the sample [34]. The strongest assumption made by CONTIN is that it strongly favors the simplest model, and highly penalizes curvature in the resulting size distribution. By instead applying a Bayesian framework to this analysis, it may be possible constrain the inferred size distribution using prior expectations and decrease this uncertainty when analyzing these polydisperse solutions.

Another logical extension of the developed Bayesian model is the application of inference to a discretized probability distribution. In order to analyze polydisperse solutions, CONTIN first assumes a continuous probability distribution of exponential decays $G(\Gamma)$, similar to the assumptions made in the cumulants expansion in Chapter 2. Instead of assuming that the probabilities are closely clustered around some mean value of Γ, however, CONTIN starts by discretizing the distribution of decay rates $G(\Gamma)$. This can be seen in Figure 50.

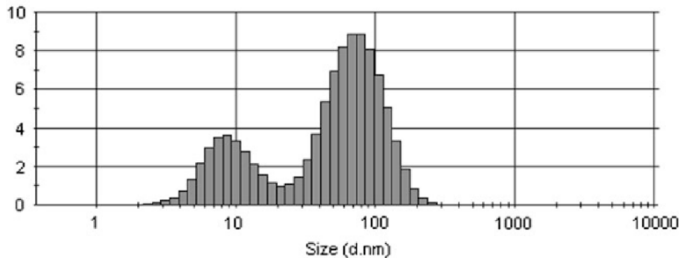


Figure 50: A continuous size distribution of roughly 10 nm and 100 nm silver particles found by CONTIN analysis. Each bar of the histogram represents a discrete probability of that given size range, which can then be used to infer the relative distribution of sizes within the sample [4].

While this discretized probability requires a complex algorithm when analyzed with traditional fitting techniques, the jump from a monodisperse algorithm to a polydisperse is not necessarily as large when

applying Bayesian inference. Each bin shown in Figure 50 could be treated as a parameter to be inferred; this could be easily applied to the code already developed. We have already seen that while the run time of the emcee algorithm does increase with more inferred parameters, it continues to be relatively computationally cheap.

Finally, CONTIN is limited to the analysis of one angle at a time, similar to the least squares method of fitting each autocorrelation function individually, then averaging or inferring a second time based on those results. It would be preferential to develop a model that can analyze data across multiple angles simultaneously; the multi-angle Bayesian algorithm in Appendices A.4 and A.6 could potentially be expanded to polydisperse solutions to allow for this, potentially increasing the accuracy of the inference.

While the algorithms developed in the course of this thesis are currently limited to a narrow set of soft matter systems with a well defined size range, the principles developed could easily be applied to more complex systems and analysis. Further, the algorithms developed are free and open-source, which did not previously exist for this type of analysis. The analysis presented here is initially promising; with future work and expansion of the method, Bayesian inference could offer a robust alternative to traditional fitting methods for DLS analysis.

Appendices

A Python Code

A.1 Standard Approach to Monodisperse Solution- Single Angle Data

```
import matplotlib.pyplot as plt
import numpy as np
import scipy
import scipy.optimize

# Import the data as an array
dls_data = np.loadtxt("CorrelationDLS.asc", skiprows=27)

# First few rows include time, temp, etc.
# data is now in 2 colums: delay time and intensity autocorrelation function

# Unpack imported data
tau = dls_data[:,0]
intensity_ac = dls_data[:,1]

# Fitting data to model g(t) = A*e**(-C*t) + B
def fitmodel(t, C, A, B):
    return A*np.e**(-C*t) + B

popt, pcov = scipy.optimize.curve_fit(fitmodel, tau, intensity_ac)
print ("best-fit parameters: ", popt)
print pcov

popt_uncertainties = np.sqrt(np.diag(pcov))
print popt_uncertainties

frac_uncertainties = (popt_uncertainties / popt)
```

```
print frac_uncertainties

# Use this constant to determine the diffusion constant D
# Data taken at a scattering angle of 90 degrees with a 632.8 nm HeNe laser

lam = 632.8e-9 # wavelength in meters
theta = np.pi # scattering angle in radians
n = 1.33200 # refractive index of water

q = (4*np.pi * n) / (lam) * np.sin(theta/2)

C = popt[0] # exponetial constant from fit
# C = 2*D*q^2
D = (C / (2*q**2)) /0.001 # Diffusion constant

# To find the hydrodynamic radius, use the relationship D = kT/(6*pi*eta*R)

k = 1.38065e-23 # Boltzmann's constant, J/K (joules per kelvin)
T = 297.94999 # Kelvin, from textfile
eta = 8.9e-4 # viscosity of water, Pa s (pascal seconds)

R = k*T / (6*np.pi*eta*D)
R_unc = R * frac_uncertainties[0]
print "The hydrodynamic radius found from the fit is", str(R)
print "The uncertainity of R is +-", str(R_unc)
```

A.2 Bayesian Approach to Monodisperse Solution- Single Angle Data

```
# Import the data as an array
dls_data = np.loadtxt("CorrelationDLS.asc", skiprows=27)
# Unpack imported data
tau = dls_data[:,0]
intensity_ac = dls_data[:,1]
# Assume intensity autocorrelation function has a constant uncertainty of +-5%
sig_y = intensity_ac*0.05

# Now define the log of the liklihood function using uniform prior
def log_prior(theta):
    # returns log of prior probability distribution
    A, C = theta # unpack the model parameters

    # Set a uniform prior, but within boundaries.
    if 0 < A < 10.0 and 0.0 < C < 500.0:
        return 0.0  # Since the probability is 1, this returns 0.
    else:
        return -np.inf # Since the probability is 0

def log_likelihood(theta, x, y, sig_y):
    # returns the log of the likelihood function

    # theta: model parameters (specified as a tuple)
    # x: angles
    # y: measured tau
    # sig_y: uncertainties on measured data, set to be +- 5% of the value

    A, C = theta # unpack the model parameters

    # Using the model A*np.e**(-C*t), define the log of the likelihood function
    # ln (L) = K - 1/2 * Sum [(y- function)^2 / sigma^2]
```

```
    # ln (L) = K - 1/2 Chi^2
    # Based on derivation in Hogg, Bovy, and Lang paper

    residual = (y - A*np.e**(-C*x))**2
    chi_square = np.sum(residual/(sig_y**2))

    # the constant K is determined by the Gaussian function
    constant = np.sum(np.log(1/np.sqrt(2.0*np.pi*sig_y**2)))
    return constant - 0.5*chi_square

def log_posterior(theta, x, y, sig_y):
    # returns log of posterior probability distribution
    A, C = theta

    # Bayes Theorem: Posterior = Prior * likelihood
    # Ln (Posterior) = Ln (Prior ) + Ln (Likelihood)
    return log_prior(theta) + log_likelihood(theta, x, y, sig_y)

# the model has 2 parameters; use 50 walkers and 500 steps each
ndim = 2
nwalkers = 50
nsteps = 500

# set up the walkers in a "Gaussian ball" around the least-squares estimate
# The least squares fit estimate: A = 0.9244, C = 33.155
ls_result = [0.92445056, 33.15515352] # A, C
starting_positions = [ls_result +
                      1e-4*np.random.randn(ndim) for i in range(nwalkers)]

# set up the sampler object using emcee
sampler = emcee.EnsembleSampler(nwalkers, ndim, log_posterior,
                                args=(tau, intensity_ac, sig_y))
```

```
%time sampler.run_mcmc(starting_positions, nsteps)

fig, (ax_A, ax_C) = plt.subplots(2)
ax_A.set(ylabel='A')
ax_C.set(ylabel='C')

for i in range(10):
    sns.tsplot(sampler.chain[i,:,0], ax=ax_A)
    sns.tsplot(sampler.chain[i,:,1], ax=ax_C)
# It takes about 100 steps for the walkers to settle
# Trim the data to include only steps after 100
samples = sampler.chain[:,100:,:]

# reshape the samples into a 1D array where the colums are A, C
traces = samples.reshape(-1, ndim).T
parameter_samples = pd.DataFrame({'A': traces[0], 'C': traces[1]})
```

A.3 Standard Approach to Monodisperse Solution- Multiangle Data

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import scipy
import scipy.stats
import scipy.optimize

# Applying simple exponential fit to multiangle data
dls_data = np.loadtxt("100nm.dat", skiprows=1)
# Data is in columns based on angle, first row labels these columns

tau = dls_data[:,0]
intensity_20 = dls_data[:,1]
intensity_40 = dls_data[:,2]
intensity_60 = dls_data[:,3]
intensity_80 = dls_data[:,4]
intensity_100 = dls_data[:,5]
intensity_120 = dls_data[:,6]
intensity_140 = dls_data[:,7]
intensity_160 = dls_data[:,8]

def fitmodel(t, C, A, B):
    return A*np.e**(-C*t) + B

popt1, pcov1 = scipy.optimize.curve_fit(fitmodel, tau, intensity_20)
popt2, pcov2 = scipy.optimize.curve_fit(fitmodel, tau, intensity_40)
popt3, pcov3 = scipy.optimize.curve_fit(fitmodel, tau, intensity_60)
popt4, pcov4 = scipy.optimize.curve_fit(fitmodel, tau, intensity_80)
popt5, pcov5 = scipy.optimize.curve_fit(fitmodel, tau, intensity_100)
popt6, pcov6 = scipy.optimize.curve_fit(fitmodel, tau, intensity_120)
popt7, pcov7 = scipy.optimize.curve_fit(fitmodel, tau, intensity_140)
```

```
popt8, pcov8 = scipy.optimize.curve_fit(fitmodel, tau, intensity_160)

# Plot linear relationship between q^2 and fit constant Gamma to find D

gamma = np.array([popt1[0], popt2[0], popt3[0], popt4[0], popt5[0],
                  popt6[0], popt7[0], popt8[0]])

#Load in the angles
dls_data = np.loadtxt("100nm.dat", usecols=(1,2,3,4,5,6,7,8))
theta = dls_data[0,:]
theta = theta*(np.pi/180) # scattering angle in radians

lam = 658e-9 # wavelength in meters
n = 1.333 # refractive index

q = ((4*np.pi * n) /(lam)) * np.sin(theta/2)

def straight_line_model(x, A, B):
    '''
    Model function for a straight-line fit with y-intercept A and slope B.
    '''
    return A + B * x

popt, pcov = scipy.optimize.curve_fit(straight_line_model, q**2, gamma)
popt_uncertainties = np.sqrt(np.diag(pcov))
print popt_uncertainties

frac_uncertainties = (popt_uncertainties / popt)
print frac_uncertainties

# The slope gives the value for diffusion coefficient 2*D
```

```
# To find the radius, use the relationship D = kT/(6*pi*eta*R)

k = 1.38065e-23 # Boltzmann's constant, J/K
T = 293.08 # Kelvin, roomtemp
eta = 10.016e-4 # viscosity of water, Pa s at temperature of 293.08 Kelvin

# popt[1] = 2*D
D = popt[1] /2

R = k*T / (6*np.pi*eta*D)
R_unc_1 = R * frac_uncertainties[1]

print "The calculated hydrodynamic radius: ", R
print "The uncertainity of R is +-", str(R_unc_1)
```

A.4 Bayesian Approach to Monodisperse Solution- Multiangle Data

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import scipy
import scipy.stats
import scipy.optimize
%matplotlib inline
import emcee
import seaborn as sns
import pandas as pd

#Load in data
dls_data = np.loadtxt("100nm.dat", skiprows=1)

# Unpack the data
tau = dls_data[:,0]
intensity_20 = dls_data[:,1]
intensity_40 = dls_data[:,2]
intensity_60 = dls_data[:,3]
intensity_80 = dls_data[:,4]
intensity_100 = dls_data[:,5]
intensity_120 = dls_data[:,6]
intensity_140 = dls_data[:,7]
intensity_160 = dls_data[:,8]

#Load in the angles
dls_data = np.loadtxt("100nm.dat", usecols=(1,2,3,4,5,6,7,8))
phi = dls_data[0,:]
phi = phi*(np.pi/180) # scattering angle in radians

# Set a uniform uncertainity on the data as 5%
```

```
sig_g_20 = intensity_20*0.05
sig_g_40 = intensity_40*0.05
sig_g_60 = intensity_60*0.05
sig_g_80 = intensity_80*0.05
sig_g_100 = intensity_100*0.05
sig_g_120 = intensity_120*0.05
sig_g_140 = intensity_140*0.05
sig_g_160 = intensity_160*0.05

#Construct empty arrays to fill with all the data
all_angles= []
all_intensity =[]
all_sig = []
all_tau = []

# Construct an array that has each angle repeated for each data point
# Ex: [ 20, 20, 20, ....., 40, 40, 40, ....., 60, 60, 60, ....]
for j in range(0, 8):
    for i in range(0, len(intensity_20)):
        all_angles = np.append(all_angles, phi[j])

# Construct an array of all the intensity autocorrelation data
list_of_data = [intensity_20, intensity_40, intensity_60, intensity_80,
                intensity_100, intensity_120, intensity_140, intensity_160]
for item in list_of_data:
    for i in range(0, len(item)):
        all_intensity = np.append(all_intensity, item[i])

# Construct an array of all the errors for all the ac data
list_of_error = [sig_g_20, sig_g_40, sig_g_60, sig_g_80, sig_g_100,
                 sig_g_120, sig_g_140, sig_g_160]
for item in list_of_error:
```

```
    for i in range(0, len(item)):
        all_sig = np.append(all_sig, item[i])

# Construct an array of the delay time tau that repeats for each data set
# Ex: (0.01, 0.1, 1, .... 0.01, 0.1, 1, .... )
for j in range(0, 8):
    for i in range(0, len(tau)):
        all_tau = np.append(all_tau, tau[i])

# Now define the log of the liklihood function
# Assume that the prior is equal probablity
def log_prior(theta):
    # returns log of prior probability distribution
    # First unpack the model parameters, where B is baseline, D is the decay
    # constant, and each A is the amplitude associated with 1 angle measurement
    B, D, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8 = theta

    # Set a uniform prior, but within boundaries. Both D and B must be positive
    if 0.0 < D and 0.0 < B < 10.0:
        return 0.0  # Since the probability is 1, returns 0.
    else:
        return -np.inf # Since the probability is 0, returns - infinity.


def log_likelihood(theta, tau, phi, g, sig_g):
    # returns the log of the likelihood function

    # tau: delay time
    # g: measurements (autocorrelation function)
    # sig_g: uncertainties on measured data, set to be +- 5% of the value

    # Unpack the model parameters
```

```
B, D, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8 = theta

# Using the model A_i*np.e**(-2*D* (4 pi n / lam * sin(phi/2) )**2 *tau) + B
# define the log of the likelihood function as
# ln (L) = K - 1/2 * Sum [(y- function)^2 / sigma^2]
# ln (L) = K - 1/2 Chi^2

n = 1.333 # refractive index

# To account for the different amplitudes, construct an array of all values
# Each repeats for the length of the associated data set
A = np.array([])

for i in range(0, len(intensity_20)):
    A = np.append(A, A_1)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_2)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_3)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_4)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_5)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_6)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_7)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_8)

# Define the angle dependent portion
m = (4*np.pi * n * np.sin(phi/2))**2
```

```
    residual = (g - A*np.e**(-2*D*m*tau) - B)**2
    chi_square = np.sum(residual/(sig_g**2))

    # the constant K is determined by the Gaussian function
    constant = np.sum(np.log(1/np.sqrt(2.0*np.pi*sig_g**2)))

    return constant - 0.5*chi_square


def log_posterior(theta, tau, phi, g, sig_g):
    # returns log of posterior probability distribution

    # Unpack the model parameters
    B, D, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8 = theta

    # Bayes Theorem: Posterior = Prior * likelihood
    # Ln (Posterior) = Ln (Prior ) + Ln (Likelihood)
    return log_prior(theta) + log_likelihood(theta, tau, phi, g, sig_g)

# the model has 10 parameters; we'll use 50 walkers and 500 steps each with emcee
ndim = 10
nwalkers = 50
nsteps = 500

# set up the walkers in a "Gaussian ball" around the least-squares estimate
# The least squares fit estimate said that the amplitude is about 0.37
# C is about 18.35, and the baseline is usually set as 1.
ls_result = [1, 10.04, 0.37, 0.37, 0.37, 0.37, 0.37, 0.37, 0.37, 0.37]
            # B, D, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8
starting_positions = [ls_result + 1e-4*np.random.randn(ndim) for i in
                      range(nwalkers)]
```

```
# set up the sampler object using emcee
sampler = emcee.EnsembleSampler(nwalkers, ndim, log_posterior,
                                args=(all_tau, all_angles, all_intensity, all_sig))

# run the sampler and use iPython's %time directive to tell us how long it took
%time sampler.run_mcmc(starting_positions, nsteps)
print('Done')

# Plot the walkers to see them converge on a value
fig, (ax_B, ax_D, ax_A_1, ax_A_2, ax_A_3, ax_A_4,
      ax_A_5, ax_A_6, ax_A_7, ax_A_8) = plt.subplots(10)
ax_B.set(ylabel='B')
ax_D.set(ylabel='D')
ax_A_1.set(ylabel='A_1')
ax_A_2.set(ylabel='A_2')
ax_A_3.set(ylabel='A_3')
ax_A_4.set(ylabel='A_4')
ax_A_5.set(ylabel='A_5')
ax_A_6.set(ylabel='A_6')
ax_A_7.set(ylabel='A_7')
ax_A_8.set(ylabel='A_8')
for i in range(10):
    sns.tsplot(sampler.chain[i,:,0], ax=ax_B)
    sns.tsplot(sampler.chain[i,:,1], ax=ax_D)
    sns.tsplot(sampler.chain[i,:,2], ax=ax_A_1)
    sns.tsplot(sampler.chain[i,:,3], ax=ax_A_2)
    sns.tsplot(sampler.chain[i,:,4], ax=ax_A_3)
    sns.tsplot(sampler.chain[i,:,5], ax=ax_A_4)
    sns.tsplot(sampler.chain[i,:,6], ax=ax_A_5)
    sns.tsplot(sampler.chain[i,:,7], ax=ax_A_6)
    sns.tsplot(sampler.chain[i,:,8], ax=ax_A_7)
```

```
    sns.tsplot(sampler.chain[i,:,8], ax=ax_A_8)

# Because it take a lot of steps before the walkers settle into the most likely
# points, cut the samples to include only values after 300 stepes
samples = sampler.chain[:,300:,:]

# Reshape the samples into a 1D array
traces = samples.reshape(-1, ndim).T

# create a pandas DataFrame with labels to allow for calculations and graphing
parameter_samples = pd.DataFrame({'B': traces[0], 'D': traces[1], 'A_1': traces[2], 'A_2':
                                  'A_4': traces[5], 'A_5': traces[6], 'A_6': traces[7], 'A_

# calculate the most likely value and the uncertainties using pandas
q = parameter_samples.quantile([0.16,0.50,0.84], axis=0)

# Finally, solve for the hydrodynamic radius

k = 1.38065e-23 # Boltzmann's constant, J/K
T = 293.08 # Kelvin, roomtemp
eta = 10.016e-4 # viscosity of water, Pa s at temperature of 293.08 Kelvin

# Solve for true diffusion coeffient, since we fit to D/lam^2
# D = D / lam^2

lam = 658e-9 # wavelength of laser in meters
D = 20.407227*lam**2

R = k*T / (6*np.pi*eta*D)
print "The hydrodynamic radius: ", R
```

A.5 Method of Cumulants

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import scipy
import scipy.optimize
import scipy.misc
import scipy.stats

# Import the data as an array
dls_data = np.loadtxt("CorrelationDLS.asc", skiprows=27)
tau = dls_data[:,0]
intensity_ac = dls_data[:,1]

# Using the expansion
# g^2 = B + beta* e^{-2*Gamma\tau}*(1 + mu_2/2! *tau^2 - mu_3/3!*tau^3 )

#2nd Order Cumulant Fit
def fitmodel_2nd_order(t, B, beta, Gamma, mu_2):
    return B + beta*np.e**(-2*Gamma*t)* (1 + (mu_2/2 *t**2)**2)

popt1, pcov1 = scipy.optimize.curve_fit(fitmodel_2nd_order, tau, intensity_ac)
popt_uncertainties1 = np.sqrt(np.diag(pcov1))
print popt_uncertainties1

frac_uncertainties = (popt_uncertainties / popt)
print frac_uncertainties

# 3rd Order Cumulant Fit
def fitmodel_3rd_order(t, B, beta, Gamma, mu_2, mu_3):
    return B + beta*np.e**(-2*Gamma*t)* (1 +(mu_2/2 *t**2)-(mu_3/6 * t**3)**2)
```

```
popt2, pcov2 = scipy.optimize.curve_fit(fitmodel_3rd_order, tau, intensity_ac)

popt_uncertainties2 = np.sqrt(np.diag(pcov2))
print popt_uncertainties2

frac_uncertainties2 = (popt_uncertainties / popt)
print frac_uncertainties2

# Use these fits to solve for the hydrodynamic radius
# Data taken at a scattering angle of 90 degrees with a 632.8 nm HeNe laser.

k = 1.38065e-23 # Boltzmann's constant, J/K (joules per kelvin)
T = 293.08 # Kelvin, roomtemp
eta = 10.016e-4 # viscosity of water, Pa s (pascal seconds)
lam = 632.8e-9 # wavelength in meters
theta = np.pi # scattering angle in radians
n = 1.33200 # refractive index

q = (4*np.pi * n) / (lam) * np.sin(theta/2)

Gamma1 = popt1[2] # exponetial constant from 2nd order fit
D1 = (Gamma1 / q**2) /0.001 # Diffusion constant with conversion from ms to s

Gamma2 = popt2[2] # exponetial constant from 3rd order fit
D2 = (Gamma2 / q**2) /0.001



R1 = k*T / (6*np.pi*eta*D1)
R_unc_1 = R1 * frac_uncertainties1[0]
print "The hydrodynamic radius from the 2nd order fit: ", R1
print "The uncertainity of R is +-", str(R_unc_1)
```

```
R2 = k*T / (6*np.pi*eta*D2)
R_unc_2 = R2 * frac_uncertainties2[0]
print "The hydrodynamic radius from the 3rd order fit: ", R2
print "The uncertainity of R is +-", str(R_unc_2)
```

A.6 Bayesian Approach to Method of Cumulants

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import scipy
import scipy.stats
import scipy.optimize
%matplotlib inline
import emcee
import seaborn as sns
import pandas as pd

#Load in data
dls_data = np.loadtxt("100nm.dat", skiprows=1)

# Unpack the data
tau = dls_data[:,0]
intensity_20 = dls_data[:,1]
intensity_40 = dls_data[:,2]
intensity_60 = dls_data[:,3]
intensity_80 = dls_data[:,4]
intensity_100 = dls_data[:,5]
intensity_120 = dls_data[:,6]
intensity_140 = dls_data[:,7]
intensity_160 = dls_data[:,8]

#Load in the angles
dls_data = np.loadtxt("100nm.dat", usecols=(1,2,3,4,5,6,7,8))
phi = dls_data[0,:]
phi = phi*(np.pi/180) # scattering angle in radians

# Set a uniform uncertainity on the data as 5%
```

```
sig_g_20 = intensity_20*0.05
sig_g_40 = intensity_40*0.05
sig_g_60 = intensity_60*0.05
sig_g_80 = intensity_80*0.05
sig_g_100 = intensity_100*0.05
sig_g_120 = intensity_120*0.05
sig_g_140 = intensity_140*0.05
sig_g_160 = intensity_160*0.05

#Construct empty arrays to fill with all the data
all_angles= []
all_intensity =[]
all_sig = []
all_tau = []

# Construct an array that has each angle repeated for each data point
# Ex: [ 20, 20, 20, ....., 40, 40, 40, ....., 60, 60, 60, ....]
for j in range(0, 8):
    for i in range(0, len(intensity_20)):
        all_angles = np.append(all_angles, phi[j])

# Construct an array of all the intensity autocorrelation data
list_of_data = [intensity_20, intensity_40, intensity_60, intensity_80,
                intensity_100, intensity_120, intensity_140, intensity_160]
for item in list_of_data:
    for i in range(0, len(item)):
        all_intensity = np.append(all_intensity, item[i])

# Construct an array of all the errors for all the ac data
list_of_error = [sig_g_20, sig_g_40, sig_g_60, sig_g_80, sig_g_100,
                 sig_g_120, sig_g_140, sig_g_160]
for item in list_of_error:
```

```
    for i in range(0, len(item)):
        all_sig = np.append(all_sig, item[i])

# Construct an array of the delay time tau that repeats for each data set
# Ex: (0.01, 0.1, 1, .... 0.01, 0.1, 1, .... )
for j in range(0, 8):
    for i in range(0, len(tau)):
        all_tau = np.append(all_tau, tau[i])

# Now define the log of the liklihood function
# Assume that the prior is equal probablity
def log_prior(theta):
    # returns log of prior probability distribution
    # First unpack the model parameters, where B is baseline, D is the decay
    # constant, and each A is the amplitude associated with 1 angle measurement
    # mu_2 is the second order cumulant
    B, D, mu_2_1, mu_2_2, mu_2_3, mu_2_4, mu_2_5, mu_2_6, mu_2_7, mu_2_8, A_1,
    A_2, A_3, A_4, A_5, A_6, A_7, A_8 = theta # unpack the model parameters

    # Set a uniform prior, but within boundaries. Both D and B must be positive
    if 0.0 < D and 0.0 < B < 10.0:
        return 0.0  # Since the probability is 1, returns 0.
    else:
        return -np.inf # Since the probability is 0, returns - infinity.


def log_likelihood(theta, tau, phi, g, sig_g):
    # returns the log of the likelihood function

    # tau: delay time
    # g: measurements (autocorrelation function)
    # sig_g: uncertainties on measured data, set to be +- 5% of the value
```

```
# Unpack the model parameters
B, D, mu_2_1, mu_2_2, mu_2_3, mu_2_4, mu_2_5, mu_2_6, mu_2_7, mu_2_8, A_1,
A_2, A_3, A_4, A_5, A_6, A_7, A_8 = theta # unpack the model parameters



# Using the model g^{(2)} = B + A e^{-2 Gamma tau}*(1 + mu_2/2! tau^2 )^2
# define the log of the likelihood function as
# ln (L) = K - 1/2 * Sum [(y- function)^2 / sigma^2]
# ln (L) = K - 1/2 Chi^2

n = 1.333 # refractive index

# To account for the different amplitudes, construct an array of all values
# Each repeats for the length of the associated data set
A = np.array([])

for i in range(0, len(intensity_20)):
    A = np.append(A, A_1)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_2)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_3)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_4)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_5)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_6)
for i in range(0, len(intensity_20)):
    A = np.append(A, A_7)
for i in range(0, len(intensity_20)):
```

```
    A = np.append(A, A_8)

# To account for the angle dependence in mu_2, construct an array of
# all values. Each repeats for the length of the associated data set
mu_2 = np.array([])

for i in range(0, len(intensity_20)):
    mu_2 = np.append(mu_2, mu_2_1)
for i in range(0, len(intensity_20)):
    mu_2 = np.append(mu_2, mu_2_2)
for i in range(0, len(intensity_20)):
    mu_2 = np.append(mu_2, mu_2_3)
for i in range(0, len(intensity_20)):
    mu_2 = np.append(mu_2, mu_2_4)
for i in range(0, len(intensity_20)):
    mu_2 = np.append(mu_2, mu_2_5)
for i in range(0, len(intensity_20)):
    mu_2 = np.append(mu_2, mu_2_6)
for i in range(0, len(intensity_20)):
    mu_2 = np.append(mu_2, mu_2_7)
for i in range(0, len(intensity_20)):
    mu_2 = np.append(mu_2, mu_2_8)

# Define the angle dependent portion
m = (4*np.pi * n * np.sin(phi/2))**2
# g^{(2)} = B + A e^{-2 Gamma tau}*(1 + mu_2/2! tau^2 )^2
residual = (g - A*np.e**(-2*D*m*tau)*(1+ mu_2/2 * tau**2)**2 - B)**2
chi_square = np.sum(residual/(sig_g**2))

# the constant K is determined by the Gaussian function
constant = np.sum(np.log(1/np.sqrt(2.0*np.pi*sig_g**2)))
```

```
    return constant - 0.5*chi_square


def log_posterior(theta, tau, phi, g, sig_g):
    # returns log of posterior probability distribution

    # Unpack the model parameters
    B, D, mu_2_1, mu_2_2, mu_2_3, mu_2_4, mu_2_5, mu_2_6, mu_2_7, mu_2_8, A_1,
    A_2, A_3, A_4, A_5, A_6, A_7, A_8 = theta # unpack the model parameters

    # Bayes Theorem: Posterior = Prior * likelihood
    # Ln (Posterior) = Ln (Prior ) + Ln (Likelihood)
    return log_prior(theta) + log_likelihood(theta, tau, phi, g, sig_g)

# the model has 11 parameters; we'll use 50 walkers and 1500 steps with emcee
# The number of steps is increased to fully explore the parameter space
ndim = 11
nwalkers = 50
nsteps = 2000

# set up the walkers in a "Gaussian ball" around the least-squares estimate
# The least squares fit estimate said that the amplitude is about 0.37
# C is about 18.35, and the baseline is usually set as 1.

ls_result = [1.0, 18.35, 7720, 129096, 1191433, 6844043, 7413197, 637558,
             994621, 816326, 0.37, 0.35, 0.32, 0.28, 0.29, 0.33, 0.37, 0.38]
             #B, D, mu_2_1, mu_2_2..., A_1, A_2...

starting_positions = [ls_result + 1e-4*np.random.randn(ndim) for i in
                      range(nwalkers)]

# set up the sampler object using emcee
```

```
sampler = emcee.EnsembleSampler(nwalkers, ndim, log_posterior,
                                args=(all_tau, all_angles, all_intensity, all_sig))

# run the sampler and use iPython's %time directive to tell us how long it took
%time sampler.run_mcmc(starting_positions, nsteps)
print('Done')

# Plot the walkers to see them converge on a value
fig, (ax_B, ax_D, ax_mu_2_1, ax_mu_2_2, ax_mu_2_3, ax_mu_2_4, ax_mu_2_5,
      ax_mu_2_6, ax_mu_2_7, ax_mu_2_8, ax_A_1, ax_A_2, ax_A_3, ax_A_4,
      ax_A_5, ax_A_6, ax_A_7, ax_A_8) = plt.subplots(18)
ax_B.set(ylabel='B')
ax_D.set(ylabel='D')
ax_mu_2_1.set(ylabel='mu_2_1')
ax_mu_2_2.set(ylabel='mu_2_2')
ax_mu_2_3.set(ylabel='mu_2_3')
ax_mu_2_4.set(ylabel='mu_2_4')
ax_mu_2_5.set(ylabel='mu_2_5')
ax_mu_2_6.set(ylabel='mu_2_6')
ax_mu_2_7.set(ylabel='mu_2_7')
ax_mu_2_8.set(ylabel='mu_2_8')
ax_A_1.set(ylabel='A_1')
ax_A_2.set(ylabel='A_2')
ax_A_3.set(ylabel='A_3')
ax_A_4.set(ylabel='A_4')
ax_A_5.set(ylabel='A_5')
ax_A_6.set(ylabel='A_6')
ax_A_7.set(ylabel='A_7')
ax_A_8.set(ylabel='A_8')
for i in range(17):
    sns.tsplot(sampler.chain[i,:,0], ax=ax_B)
    sns.tsplot(sampler.chain[i,:,1], ax=ax_D)
```

```
    sns.tsplot(sampler.chain[i,:,2], ax=ax_mu_2_1)
    sns.tsplot(sampler.chain[i,:,3], ax=ax_mu_2_2)
    sns.tsplot(sampler.chain[i,:,4], ax=ax_mu_2_3)
    sns.tsplot(sampler.chain[i,:,5], ax=ax_mu_2_4)
    sns.tsplot(sampler.chain[i,:,6], ax=ax_mu_2_5)
    sns.tsplot(sampler.chain[i,:,7], ax=ax_mu_2_6)
    sns.tsplot(sampler.chain[i,:,8], ax=ax_mu_2_7)
    sns.tsplot(sampler.chain[i,:,9], ax=ax_mu_2_8)
    sns.tsplot(sampler.chain[i,:,10], ax=ax_A_1)
    sns.tsplot(sampler.chain[i,:,11], ax=ax_A_2)
    sns.tsplot(sampler.chain[i,:,12], ax=ax_A_3)
    sns.tsplot(sampler.chain[i,:,13], ax=ax_A_4)
    sns.tsplot(sampler.chain[i,:,14], ax=ax_A_5)
    sns.tsplot(sampler.chain[i,:,15], ax=ax_A_6)
    sns.tsplot(sampler.chain[i,:,16], ax=ax_A_7)
    sns.tsplot(sampler.chain[i,:,17], ax=ax_A_8)

# Because it take a lot of steps before the walkers settle into the most likely
# points, cut the samples to include only values after 300 steps, or where ever
# it settles
samples = sampler.chain[:,900:,:]

# Reshape the samples into a 1D array
traces = samples.reshape(-1, ndim).T

# create a pandas DataFrame with labels to allow for calculations and graphing
parameter_samples = pd.DataFrame({'B': traces[0], 'D': traces[1],
                'mu_2_1': traces[2], 'mu_2_2': traces[2], 'mu_2_3': traces[4],
                'mu_2_4': traces[5], 'mu_2_5': traces[6], 'mu_2_6': traces[7],
                'mu_2_7': traces[8], 'mu_2_8': traces[9], 'A_1': traces[10],
                'A_2': traces[11], 'A_3': traces[12], 'A_4': traces[13],
                'A_5': traces[14], 'A_6': traces[15],'A_7': traces[16],
```

```
                'A_8': traces[17]})

# calculate the most likely value and the uncertainties using pandas
q = parameter_samples.quantile([0.16,0.50,0.84], axis=0)

k = 1.38065e-23 # Boltzmann's constant, J/K (joules per kelvin)
T = 293.08 # Kelvin, roomtemp
eta = 10.016e-4 # viscosity of water, Pa s (pascal seconds) at temperature of 293.08 Kelvi

# D = D / lam^2
uncertainty = ((11.701408-11.169676) + (12.269040-11.701408)) / 2
frac_uncert = uncertainty /11.701408



lam = 658e-9 # wavelength in meters
D = 11.701408*lam**2
print D

R = k*T / (6*np.pi*eta*D)
print "Radius: ", R

R_unc = R * frac_uncert
print R_unc

D_unc = frac_uncert*D
print D_unc

n = 1.333
q = (4*np.pi * n * np.sin(phi/2))

Gamma = D*q**2
```

```
mu = np.array([])
sigma = Gamma**2 / mu

print "The values for the polydispersity index are: ", sigma
```

A.6.1 Inferring the Polydispersity Index

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import scipy
import scipy.stats
import scipy.optimize
%matplotlib inline
import emcee
import seaborn as sns
import pandas as pd

#Load in data
dls_data = np.loadtxt("100nm.dat", skiprows=1)

# Unpack the data
tau = dls_data[:,0]
intensity_20 = dls_data[:,1]
intensity_40 = dls_data[:,2]
intensity_60 = dls_data[:,3]
intensity_80 = dls_data[:,4]
intensity_100 = dls_data[:,5]
intensity_120 = dls_data[:,6]
intensity_140 = dls_data[:,7]
intensity_160 = dls_data[:,8]

#Load in the angles
dls_data = np.loadtxt("100nm.dat", usecols=(1,2,3,4,5,6,7,8))
```

```
phi = dls_data[0,:]
phi = phi*(np.pi/180) # scattering angle in radians

# Set a uniform uncertainity on the data as 5%
sig_g_20 = intensity_20*0.05
sig_g_40 = intensity_40*0.05
sig_g_60 = intensity_60*0.05
sig_g_80 = intensity_80*0.05
sig_g_100 = intensity_100*0.05
sig_g_120 = intensity_120*0.05
sig_g_140 = intensity_140*0.05
sig_g_160 = intensity_160*0.05

#Construct empty arrays to fill with all the data
all_angles= []
all_intensity =[]
all_sig = []
all_tau = []

# Construct an array that has each angle repeated for each data point
# Ex: [ 20, 20, 20, ....., 40, 40, 40, ....., 60, 60, 60, ....]
for j in range(0, 8):
    for i in range(0, len(intensity_20)):
        all_angles = np.append(all_angles, phi[j])

# Construct an array of all the intensity autocorrelation data
list_of_data = [intensity_20, intensity_40, intensity_60, intensity_80,
                intensity_100, intensity_120, intensity_140, intensity_160]
for item in list_of_data:
    for i in range(0, len(item)):
        all_intensity = np.append(all_intensity, item[i])
```

```
# Construct an array of all the errors for all the ac data
list_of_error = [sig_g_20, sig_g_40, sig_g_60, sig_g_80, sig_g_100,
                 sig_g_120, sig_g_140, sig_g_160]
for item in list_of_error:
    for i in range(0, len(item)):
        all_sig = np.append(all_sig, item[i])

# Construct an array of the delay time tau that repeats for each data set
# Ex: (0.01, 0.1, 1, .... 0.01, 0.1, 1, .... )
for j in range(0, 8):
    for i in range(0, len(tau)):
        all_tau = np.append(all_tau, tau[i])

# Now define the log of the liklihood function
# Assume that the prior is equal probablity
def log_prior(theta):
    # returns log of prior probability distribution
    # First unpack the model parameters, where B is baseline, D is the decay
    # constant, and each A is the amplitude associated with 1 angle measurement
    # mu_2, the second order cumulant, has been factored out
    # the constant P, the polydispersity index, is constant across angles
    # P = mu_2 / Gamma^2
    B, D, P, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8 = theta
    # unpack the model parameters

    # Set a uniform prior, but within boundaries. Both D and B must be positive
    if 0.0 < D and 0.0 < B < 10.0 and 0.0 < P:
        return 0.0  # Since the probability is 1, returns 0.
    else:
        return -np.inf # Since the probability is 0, returns - infinity.
```

```
def log_likelihood(theta, tau, phi, g, sig_g):
    # returns the log of the likelihood function

    # tau: delay time
    # g: measurements (autocorrelation function)
    # sig_g: uncertainties on measured data, set to be +- 5% of the value

    # Unpack the model parameters
    B, D, P, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8  = theta



    # Using the model g^{(2)} = B + A e^{-2 Gamma tau}*(1 + mu_2/2! tau^2 )^2
    # define the log of the likelihood function as
    # ln (L) = K - 1/2 * Sum [(y- function)^2 / sigma^2]
    # ln (L) = K - 1/2 Chi^2

    n = 1.333 # refractive index

    # To account for the different amplitudes, construct an array of all values
    # Each repeats for the length of the associated data set
    A = np.array([])

    for i in range(0, len(intensity_20)):
        A = np.append(A, A_1)
    for i in range(0, len(intensity_20)):
        A = np.append(A, A_2)
    for i in range(0, len(intensity_20)):
        A = np.append(A, A_3)
    for i in range(0, len(intensity_20)):
        A = np.append(A, A_4)
    for i in range(0, len(intensity_20)):
        A = np.append(A, A_5)
```

```
    for i in range(0, len(intensity_20)):
        A = np.append(A, A_6)
    for i in range(0, len(intensity_20)):
        A = np.append(A, A_7)
    for i in range(0, len(intensity_20)):
        A = np.append(A, A_8)

    # Define the angle dependent portion
    m = (4*np.pi * n * np.sin(phi/2))**2
    # g^{(2)} = B + A e^{-2 Gamma tau}*(1 +
    #                     (mu_2/Gamma^2)* (Gamma^2/mu_2) *mu_2/2! tau^2 )^2
    # P - mu_2 / Gamma^2
    # g^{(2)} = B + A e^{-2 Gamma tau}*(1 + P* Gamma^2/2! tau^2 )^2
    residual = (g - A*np.e**(-2*D*m*tau)*(1 + P*(D*m)**2/2*tau**2)**2 - B)**2
    chi_square = np.sum(residual/(sig_g**2))

    # the constant K is determined by the Gaussian function
    constant = np.sum(np.log(1/np.sqrt(2.0*np.pi*sig_g**2)))

    return constant - 0.5*chi_square


def log_posterior(theta, tau, phi, g, sig_g):
    # returns log of posterior probability distribution

    # Unpack the model parameters
    B, D, P, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8 = theta

    # Bayes Theorem: Posterior = Prior * likelihood
    # Ln (Posterior) = Ln (Prior ) + Ln (Likelihood)
    return log_prior(theta) + log_likelihood(theta, tau, phi, g, sig_g)
```

```
# the model has 11 parameters; we'll use 50 walkers and 1500 steps with emcee
# The number of steps is increased to fully explore the parameter space
ndim = 11
nwalkers = 50
nsteps = 2000

# set up the walkers in a "Gaussian ball" around the least-squares estimate
# The least squares fit estimate said that the amplitude is about 0.37
# C is about 18.35, and the baseline is usually set as 1.

ls_result = [1.0, 18.35, 0.5, 0.37, 0.35, 0.32, 0.28, 0.29, 0.33, 0.37, 0.38]
            #B, D, P, A_1, A_2...

starting_positions = [ls_result + 1e-4*np.random.randn(ndim) for i in
                      range(nwalkers)]

# set up the sampler object using emcee
sampler = emcee.EnsembleSampler(nwalkers, ndim, log_posterior,
                                args=(all_tau, all_angles, all_intensity, all_sig))

# run the sampler and use iPython's %time directive to tell us how long it took
%time sampler.run_mcmc(starting_positions, nsteps)
print('Done')

# Plot the walkers to see them converge on a value
fig, (ax_B, ax_D, ax_P, ax_A_1, ax_A_2, ax_A_3, ax_A_4,
      ax_A_5, ax_A_6, ax_A_7, ax_A_8) = plt.subplots(11)
ax_B.set(ylabel='B')
ax_D.set(ylabel='D')
ax_P.set(ylabel='P')
ax_A_1.set(ylabel='A_1')
ax_A_2.set(ylabel='A_2')
```

```
ax_A_3.set(ylabel='A_3')
ax_A_4.set(ylabel='A_4')
ax_A_5.set(ylabel='A_5')
ax_A_6.set(ylabel='A_6')
ax_A_7.set(ylabel='A_7')
ax_A_8.set(ylabel='A_8')
for i in range(10):
    sns.tsplot(sampler.chain[i,:,0], ax=ax_B)
    sns.tsplot(sampler.chain[i,:,1], ax=ax_D)
    sns.tsplot(sampler.chain[i,:,2], ax=axP)
    sns.tsplot(sampler.chain[i,:,3], ax=ax_A_1)
    sns.tsplot(sampler.chain[i,:,4], ax=ax_A_2)
    sns.tsplot(sampler.chain[i,:,5], ax=ax_A_3)
    sns.tsplot(sampler.chain[i,:,6], ax=ax_A_4)
    sns.tsplot(sampler.chain[i,:,7], ax=ax_A_5)
    sns.tsplot(sampler.chain[i,:,8], ax=ax_A_6)
    sns.tsplot(sampler.chain[i,:,9], ax=ax_A_7)
    sns.tsplot(sampler.chain[i,:,10], ax=ax_A_8)

# Because it take a lot of steps before the walkers settle into the most likely
# points, cut the samples to include only values after 500 steps, or where ever
# it settles
samples = sampler.chain[:,500:,:]

# Reshape the samples into a 1D array
traces = samples.reshape(-1, ndim).T

# create a pandas DataFrame with labels to allow for calculations and graphing
parameter_samples = pd.DataFrame({'B': traces[0], 'D': traces[1],
                'P': traces[2], 'A_1': traces[3],
                'A_2': traces[4], 'A_3': traces[5], 'A_4': traces[6],
                'A_5': traces[7], 'A_6': traces[8],'A_7': traces[9],
```

```
                'A_8': traces[10]})

# calculate the most likely value and the uncertainties using pandas
q = parameter_samples.quantile([0.16,0.50,0.84], axis=0)

k = 1.38065e-23 # Boltzmann's constant, J/K (joules per kelvin)
T = 293.08 # Kelvin, roomtemp
eta = 10.016e-4 # viscosity of water, Pa s (pascal seconds) at temperature of 293.08 Kelvi

# D = D / lam^2
uncertainty = ((11.701408-11.169676) + (12.269040-11.701408)) / 2
frac_uncert = uncertainty /11.701408



lam = 658e-9 # wavelength in meters
D = 11.701408*lam**2
print D

R = k*T / (6*np.pi*eta*D)
print "Radius: ", R

R_unc = R * frac_uncert
print R_unc

D_unc = frac_uncert*D
print D_unc
```

References

[1] J. Stetefeld, S. A. McKenna, and T. R. Patel, "Dynamic light scattering: a practical guide and applications in biomedical sciences," *Biophysical Reviews*, vol. 8, pp. 409–427, Dec. 2016.

[2] "Laser to Fiber Source Couplers Application Notes," Feb. 2015.

[3] D. Sivia and J. Skilling, *Data Analysis: A Bayesian Tutorial*. OUP Oxford, June 2006. Google-Books-ID: Kxx8CwAAQBAJ.

[4] G. Dzido and A. Jarzebski, "Fabrication of silver nanoparticles in a continuous flow, low temperature microwave-assisted polyol process," *Journal of Nanoparticle Research*, vol. 13, pp. 2533–2541, June 2011.

[5] T. G. Dimiduk and V. N. Manoharan, "Bayesian approach to analyzing holograms of colloidal particles," *Opt. Express*, vol. 24, pp. 24045–24060, Oct. 2016.

[6] P. A. Hassan, S. Rana, and G. Verma, "Making Sense of Brownian Motion: Colloid Characterization by Dynamic Light Scattering," *Langmuir*, vol. 31, pp. 3–12, Jan. 2015.

[7] B. J. Berne and R. Pecora, *Dynamic Light Scattering: With Applications to Chemistry, Biology, and Physics*. Courier Corporation, 2000. Google-Books-ID: vBB54ABhmuEC.

[8] C. E. Dorman, P. Guhathakurta, M. A. Fardal, D. Lang, M. C. Geha, K. M. Howley, J. S. Kalirai, J. S. Bullock, J.-C. Cuillandre, J. J. Dalcanton, K. M. Gilbert, A. C. Seth, E. J. Tollerud, B. F. Williams, and B. Yniguez, "The SPLASH Survey: Kinematics of Andromeda's Inner Spheroid," *The Astrophysical Journal*, vol. 752, p. 147, June 2012. arXiv: 1204.4455.

[9] J. N. Rouder, P. L. Speckman, D. Sun, R. D. Morey, and G. Iverson, "Bayesian t tests for accepting and rejecting the null hypothesis," *Psychonomic Bulletin & Review*, vol. 16, pp. 225–237, Apr. 2009.

[10] Altman Douglas G., "Statistics in medical journals: some recent trends," *Statistics in Medicine*, vol. 19, pp. 3275–3289, Nov. 2000.

[11] Andrews Mark and Baguley Thom, "Prior approval: The growth of Bayesian methods in psychology," *British Journal of Mathematical and Statistical Psychology*, vol. 66, pp. 1–7, Jan. 2013.

[12] P. E. Rossi and G. M. Allenby, "Bayesian Statistics and Marketing," *Marketing Science*, vol. 22, no. 3, pp. 304–328, 2003.

[13] M. Naiim, A. Boualem, C. Ferre, M. Jabloun, A. Jalocha, and P. Ravier, "Multiangle dynamic light scattering for the improvement of multimodal particle size distribution measurements," *Soft Matter*, vol. 11, no. 1, pp. 28–32, 2015.

[14] A. Boualem, M. Jabloun, P. Ravier, M. Naiim, and A. Jalocha, "A reversible jump MCMC algorithm for Particle Size inversion in Multiangle Dynamic Light Scattering," in *2014 22nd European Signal Processing Conference (EUSIPCO)*, pp. 1327–1331, Sept. 2014.

[15] L. A. Clementi, J. R. Vega, L. M. Gugliotta, and H. R. B. Orlande, "A Bayesian inversion method for estimating the particle size distribution of latexes from multiangle dynamic light scattering measurements," *Chemometrics and Intelligent Laboratory Systems*, vol. 107, pp. 165–173, May 2011.

[16] F. Reif, *Fundamentals of Statistical and Thermal Physics*. Waveland Press, Jan. 2009. Google-Books-ID: ObsbAAAAQBAJ.

[17] A. Siegert, "On the fluctuations in signals returned by many independently moving scatterers," *Radiation Laboratory, Massachusetts Institute of Technology*, 1943.

[18] B. J. Frisken, "Revisiting the method of cumulants for the analysis of dynamic light-scattering data," *Applied Optics*, vol. 40, pp. 4087–4091, Aug. 2001.

[19] A. G. Mailer, P. S. Clegg, and P. N. Pusey, "Particle sizing by dynamic light scattering: non-linear cumulant analysis," *Journal of Physics: Condensed Matter*, vol. 27, p. 145102, Apr. 2015. arXiv: 1504.06502.

[20] T. Kodger, "Dynamic Light Scattering: Techniques and Tutorials."

[21] S. W. Provencher, "CONTIN: A general purpose constrained regularization program for inverting noisy linear algebraic and integral equations," *Computer Physics Communications*, vol. 27, pp. 229–242, Sept. 1982.

[22] S. I. Inc, "ST100 Variable Angle Light Scattering Instrument," Mar. 2005.

[23] V. Degiorgio, M. Corti, and C. Minero, "Laser light scattering in micellar solutions," *Nuovo cimento della Societa italiana di fisica.*, vol. 3, pp. 44–61, Jan. 1984.

[24] "Sulfate Latex Beads, 8% w/v, 0.04 m - Thermo Fisher Scientific."

[25] Scipy, "scipy.optimize.curve_fit — SciPy v1.0.0 Reference Guide," Oct. 2017.

[26] "Example: Fitting a Model to Data — emcee 2.2.1 documentation."

[27] P. J. Green, "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination," *Biometrika*, vol. 82, no. 4, pp. 711–732, 1995.

[28] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, "emcee: The MCMC Hammer," *Publications of the Astronomical Society of the Pacific*, vol. 125, pp. 306–312, Mar. 2013. arXiv: 1202.3665.

[29] A. Kirichenko, A. Danilenko, P. Shternin, Y. Shibanov, E. Ryspaeva, D. Zyuzin, M. Durant, O. Kargaltsev, G. Pavlov, and A. Cabrera-Lavers, "Optical observations of PSR J2021+3651 in the Dragonfly Nebula with the GTC," *The Astrophysical Journal*, vol. 802, p. 17, Mar. 2015. arXiv: 1501.04594.

[30] L. M. Pérez, A. Isella, J. M. Carpenter, and C. J. Chandler, "Large-Scale Asymmetries in the Transitional Disks of SAO 206462 and SR 21," *The Astrophysical Journal*, vol. 783, p. L13, Feb. 2014. arXiv: 1402.0832.

[31] D. Narbutis, D. Semionov, R. Stonkutė, P. de Meulenaer, T. Mineikis, A. Bridžius, and V. Vansevičius, "Deriving structural parameters of semi-resolved star clusters. FitClust: a program for crowded fields," *Astronomy & Astrophysics*, vol. 569, p. A30, Sept. 2014. arXiv: 1410.2514.

[32] D. W. Hogg, J. Bovy, and D. Lang, "Data analysis recipes: Fitting a model to data," *arXiv:1008.4686 [astro-ph, physics:physics]*, Aug. 2010. arXiv: 1008.4686.

[33] T. Pella, "Unconjugated Gold and Silver colloids."

[34] A. Scotti, W. Liu, J. S. Hyatt, E. S. Herman, H. S. Choi, J. W. Kim, L. A. Lyon, U. Gasser, and A. Fernandez-Nieves, "The CONTIN algorithm and its application to determine the size distribution of microgel suspensions," *The Journal of Chemical Physics*, vol. 142, p. 234905, June 2015.

[35] S. R. Eddy, "What is Bayesian statistics?," *Nature Biotechnology*, vol. 22, p. 1177, Sept. 2004.

[36] P. Gregory, *Bayesian Logical Data Analysis for the Physical Sciences: A Comparative Approach with Mathematica® Support*. Cambridge University Press, Apr. 2005. Google-Books-ID: idkLAQAAQBAJ.

[37] B. Shaver, "A Zero-Math Introduction to Markov Chain Monte Carlo Methods," Dec. 2017.