The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

Table of contents

Calibration

Before taking any data, we calibrated the camera. The process on this is pretty simple. The camera has a single very narrow focal distance. So, we took a picture of a ruler. Then I fed the image into gimp, rotated it, and used the "measure" tool.

Ruler that we used to calibrate. The smallest marks are spaced by 10 μm (or 5 pixels). So, a pixel is 2μm

For both orientations (vertical and horizontal), there are 50 pixels between the large markings. This means that a pixel corresponds to $1/50 \times 100 = 2\mu m$and makes it easy to convert our data into meaningful units.

From the datasheet, we know that without the microscope objective the *Basler acA1920-40um* has a pixel size of 5.86x5.86 μm. So, combining that with the 3x objective we would have expected 1 pixel to be $\frac{5.86}{3} = 1.953\mu m$. Our calibration was a success!

Back to Top

Data

The data for this lab is very disjointed. Every day, we collected data on a couple of particles. Unfortunately, that means there is not a ton of carryover between the data from March 29th and the data from April 5th. So, it's a little difficult to present things effectively.

Relational databases are probably the most scalable solution to this issue -- however, they are streamlined in a way that does not allow the end-user to easily visualize the information that they contain.

This week, I decided to explore a new method for data presentation. Specifically, I am hosting all fo my .csv files on github. This allows me to create and distribute an interactive jupyter notebook. Static data will always be harder to understand. And the emergence of services like Binder (which will build a docker image from a github repo) has made it incredibly easy to do.

This lab has well over 100 plots. I will not be describing every one in detail because that would take too long and would not be especially instructive. Instead, I will conclude this report with some general high level discussion of the results and encourage the reader to explore the data on her own time.

On a bigger project, it would definitely make sense to build and host a remote database instead of just hosting files. But, for this lab I think that I can get away without really needing a database.

Hopefully the usage of this notebook is fairly intuitive. You can select any date that we were in lab from the drop-down. Then, select one of the measurements that we took and hit "refresh" in order to load all of the relevant plots and captions.

Back to Top

date 3/29/18

pressure
0.839 Torr
284 Torr
3.14 E -5 Torr
620 Torr
94.5 Torr

https://raw.githubusercontent.com/jpribyl/cautious-palm-tree/master/lab4/data/32918/particle1/data_3.14e-5/ (https://raw.githubusercontent.com/jpribyl/cautious-palm-tree/master/lab4/data/32918/particle1/data_3.14e-5/)

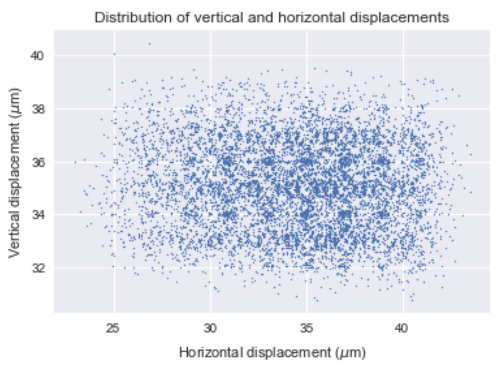
Refresh

Visualizing the Distribution

This is a scatterplot of the .csv generated by track_particle.py. It tracks the particles location over the course of time that data was collected. This plot is visually intriguing but not especially informative.

You might notice that there appear to be some strange lines / patterns in the data. At higher pressures, these are artifacts of the camera and not indicative of anything more interesting. But, at lower pressures you will start to see evidence of harmonic oscillation at a driving frequency. Examining the fourier domain should help illuminate what is actually happening.

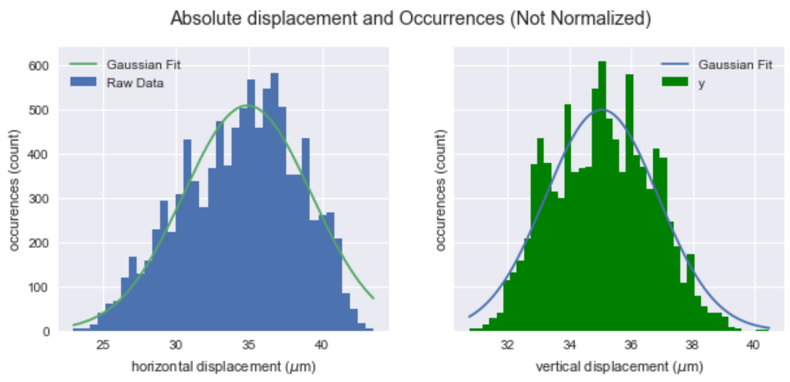
Back to Top



Thermal & Oscillatory Motion

By constructing bins and fitting the counted number of occurrences into the bins, we can start to pick some meaning out of the data. You'll notice that at higher pressures the distribution is essentially gaussian. It is a great visual representation of the random walk of an atom!

Back to Top



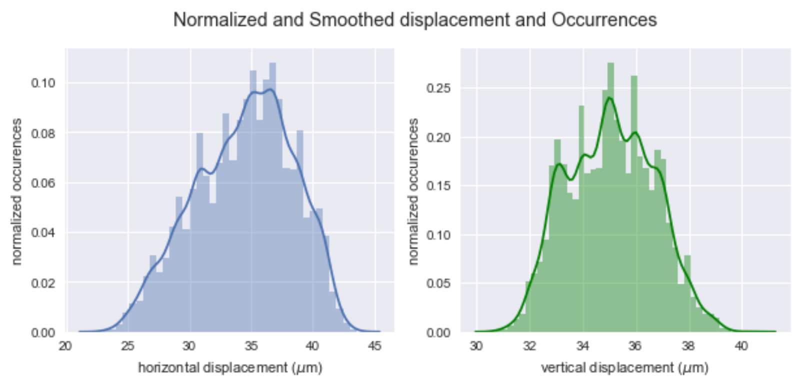
Kernel Density Estimate

Using the Seaborn plotting library allows me to easily take this analysis one step farther. From the docs, the seaborn distplot function:

> combines the matplotlib `hist` function (with automatic calculation of a good default bin size) with the seaborn `kdeplot` and `rugplot` functions. It can also fit `scipy.stats` distributions and plot the estimated PDF over the data.

In other words, it will smooth and normalize the data. This doesn't add any real information; however, it does make the plots more fun to look at.

Back to Top

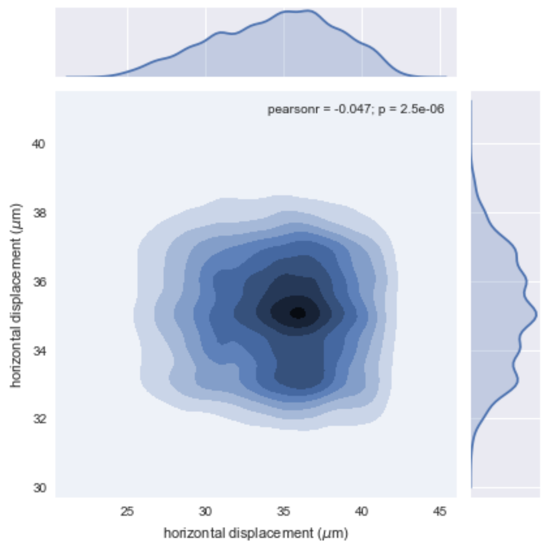


Density Contour

If we combine the kernel density estimates from the previous plot, then we can generate a contour plot. Seaborn's "jointplot" does exactly this. I like this plot because it lets you see the hotspots - IE the places where the particle spends the most time during collection.

Be careful though, artifacts could manifest in here as well. Structure does not necessarily imply an oscillatory or driven particle behavior.

Back to Top



Joint Probability Distribution

This plot shows how a distribution changes from one point to the next. Mathematically, I am plotting

$$P(x_i(t), x_i(t + \Delta t))$$

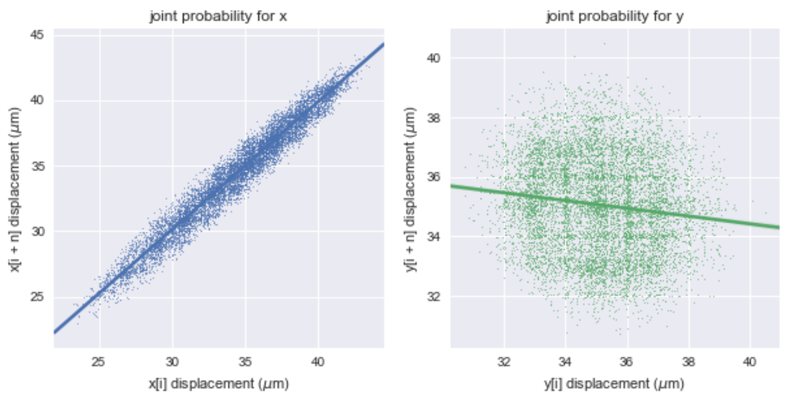
and

$$P(y_i(t), y_i(t + \Delta t))$$

I do this by taking x[i] and plotting it against x[i + n] where "i" represents an element from the array of points gathered and "n" is some integer.

Back to Top

n 1



Fourier Model

The final thing which we are asked to do is plot the distribution in the fourier domain. I made use of the fourierModel class that I wrote for the last lab. It uses SciPy's fft function.

For this lab, almost all of our data consists of 10,000 points spaced by 1/250th of a second. Unfortunately, the transient data that we took was not great. We had to throw out over half of the frames.

This means that there is no good way to resolve the bins on the transient data. We don't have any sort of consistent time between measurements. So, take these graphs with a grain of salt for the transient data. I have only included them for the sake of completeness

Particle Mass

We can use the distribution to reconstruct the mass of the particle. From lecture we know that:

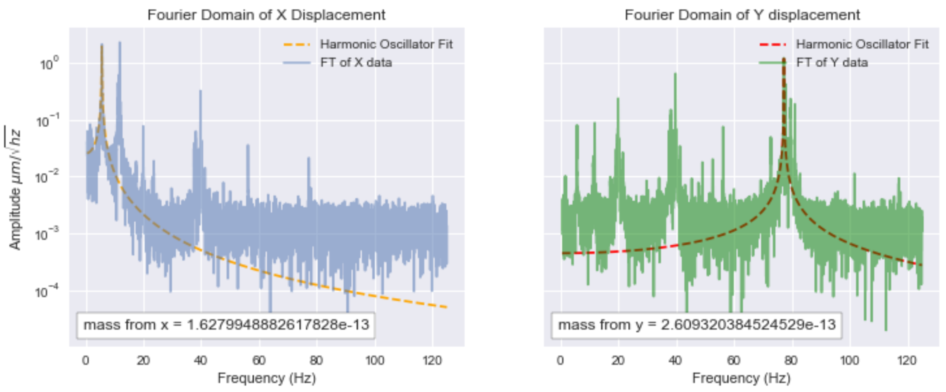
$$P(x) \propto e^{\frac{-m\omega_i^2 x_i^2}{2k_b T}}$$

So, using the standard gaussian distribution and equating the exponents we get:

$$m = \frac{k_b T}{\sigma^2 \omega_i^2}$$

Using scipy's `norm.fit(distribution)` to extract σ and getting ω by picking out the frequency of the largest peak in the fourier domain at high vacuum, we can estimate the mass of our particles.

Back to Top



Questions

High Vacuum Brownian Motion

- Using the oscillation frequencies measured under vacuum and the Brownian motion at or near atmospheric pressure, determine the mass of the particle from each of distributions. Assume that the temperature associated with the Brownian motion is 298 K. Why shouldn't you use the Brownian motion at high vacuum?

You can see my mass estimates for particles by clicking through the data above. The best estimates ought to come from the atmospheric data. However, it is interesting to see how the estimate changes as we lower the pressure.

As for the second part of this question, We discussed it (three times) during lab. At atmospheric pressure the thermal motion is dominant. However, as you increase the vacuum there will be far fewer collisions. Eventually, (typically around 1 Torr) the thermal motion will cease to be dominant. Under high vacuum, the oscillation will be dominant and brownian motion will hardly be noticeable at all.

Back to Top

Diffusion Coefficients

- Determine the diffusion coefficient of a particle at several pressures using the joint probability distribution for both the vertical and axial motion. Assume that the temperature associated with the Brownian motion is 298 K.

We will take Δt to be 4 miliseconds. This corresponds to an offset of one frame given 250 frames per second.

The first thing to do is model a linear regression and subtract off predicted values from actual values. This will give us a cluster of points that are all relatively close to zero. I make use of sklearn's `LinearRegression` class in order to build the model. The results of this may be seen in the right hand column.

Next, we can say from lecture that the diffusion coefficient 'D' is proportional to the distance of these points from zero:

$$D \propto \frac{1}{N} \sum (\Delta x)^2$$

By investigating the paper that Brian put online, we see that in the high sampling rate regime the proportionality constant reduces to $\frac{1}{2\Delta t}$ -- IE the full equation becomes

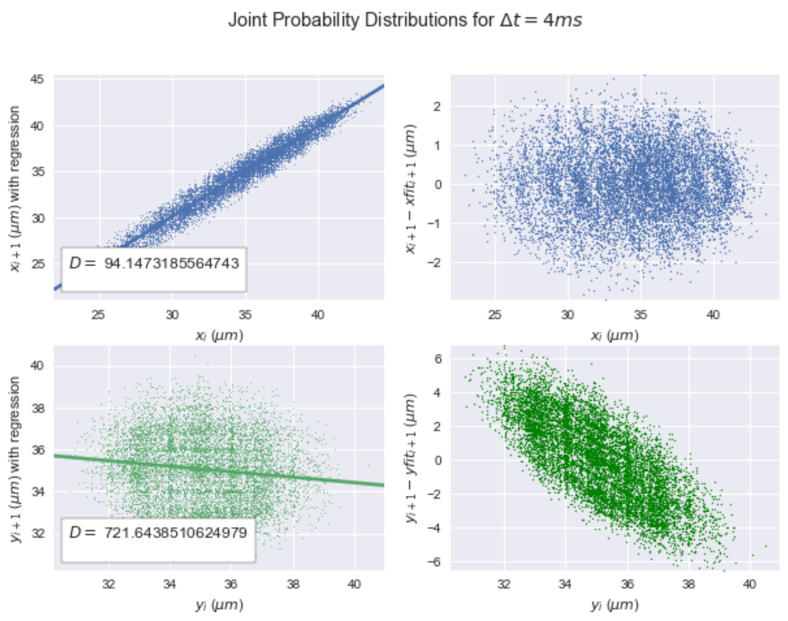
$$D = \frac{1}{2N\Delta t} \sum (\Delta x)^2 = \frac{\sigma^2}{2\Delta t}$$

Fortunately, our sampling rate of 250 frames / second is larger than our particle's natural frequency. So, we would expect this approximation to be reasonably accurate. However, it's worth noting that calculating the diffusion coefficient at low pressures does not make a lot of sense.

You can certainly still do the math and get an estimate for the particle's diffusion (IE the math is still valid). However, physically the most interesting diffusion coefficients are those at atmospheric pressure. As I mentioned before, other forces become dominant at lower pressures. So, especially when driving the particle the diffusion coefficients under high vacuum have very limited physical implications.

I have included the diffusion coefficients at low pressures for curiosity only.

Back to Top



Purpose of Brownian Measurements

- What can you learn about a particular trapped particle from the Brownian motion measurements?

I'm starting to sound a bit like a broken record at this point.. but you can get the diffusion coefficient and mass from atmospheric pressure measurements. However, you can't get a particle's natural frequecy (required for calculating mass and often required for the diffusion coefficient) without examining the particle under high vacuum.

Back to Top

Harmonic Oscillation

- Compare the thermally-driven, coherently-driven, and transient responses in the Fourier domain at all pressures measured. In each case, carefully fit the particle motion to a harmonic oscillator model, and compare the results.

This is done in the fourier data. I was able to pillage my code from last lab and make a couple of changes. According to wikipedia, a driven harmonic oscillator will have the form:

$$\frac{d^2x}{dt^2} + 2\xi\omega_0^2 x = \frac{1}{m} F_0 \sin(\omega t)$$

This has a steady state solution of:

$$x(t) = \frac{F_0}{m Z_m \omega} \sin(\omega t + \phi)$$

with impedance

$$Z_m = \sqrt{(2\omega\xi)^2 + \frac{1}{\omega^2}(\omega_0^2 - \omega^2)^2}$$

We don't need to worry about phase (IE we took the absolute value of the FT), so all that I had to do in order to fit the data was swap out my transfer function, plug in initial guesses on parameters, and let `curve_fit` do it's thing.

Again, in order to see the results of this analysis all you have to do is scroll up to the data section, select a particle / measurement, and hit refresh.

Back to Top

Quality Factor

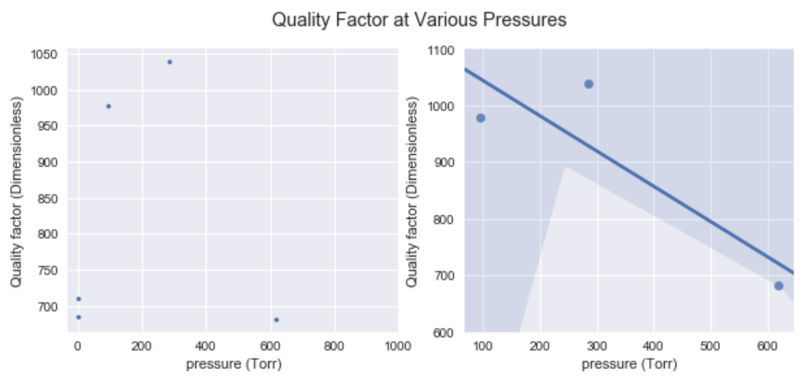
- From your Fourier domain analysis, plot the dependence of the the quality factor (Q) of the particle motion on pressure.

Unfortunately, our data for this is not terribly great. I figured out the quality factors manually by examining the ratio of $\frac{peak\ frequency}{peak\ width}$ from our 3/29 data. Physically, we expect that this data should go like 1 / p. However, it doesn't.

I talked to brian about this and his guess was that lower pressure measurements require a higher sampling rate in order to resolve the peaks well enough for this data to be meaningful.

If I clip the low-pressure data and regress the remaining points then the fit does actually go (roughly) like $\frac{1}{p}$ -- However, there is not enough data for this regression to be very meaningful.

Back to Top


Quality Factor at Various Pressures
Quality factor (Dimensionless)
pressure (Torr)
1050
1000
950
900
850
800
750
700
0
200
400
600
800
1000
Quality factor (Dimensionless)
pressure (Torr)
1100
1000
900
800
700
600
100
200
300
400
500
600