Computational Physics Homework 3

Stanley "Alex" Breitweiser"

October 28, 2015

1 Question 1

1.1 Average and Standard Deviation

Taking the average and standard deviation is pretty straightforward using built in functions; we see that Year 1 and Year 2 had similar values for temperature and humiditiy.

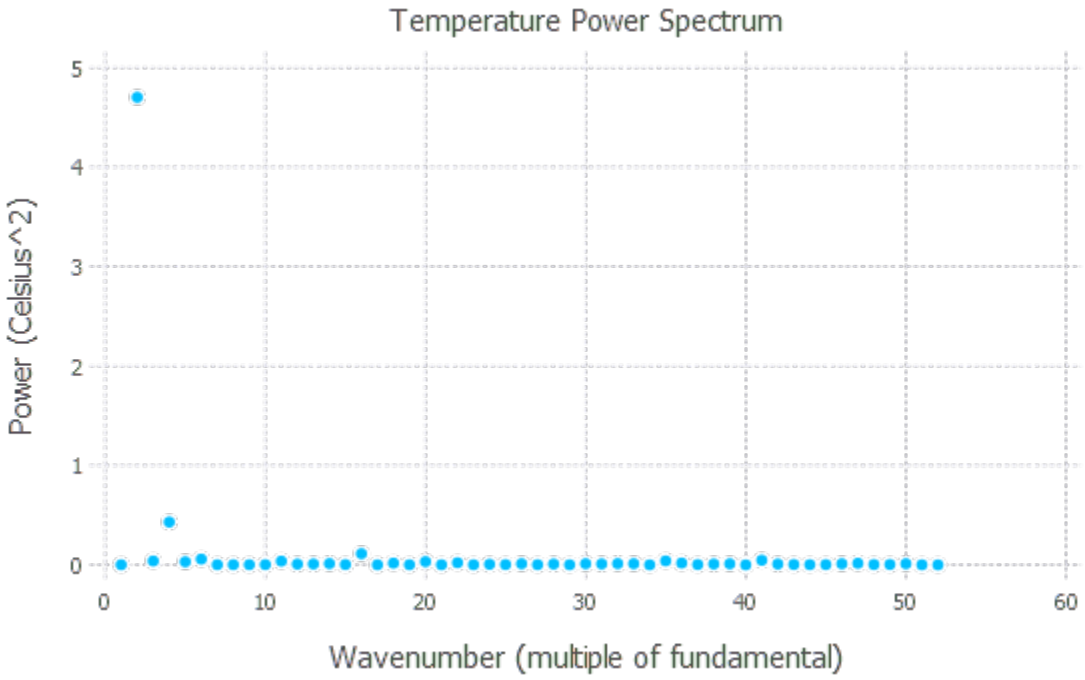
Time	Avg Temp (C)	Std Temp (C)	Avg Hum (%)	Std Hum (%)
Year 1	23.31	3.51	41.58	2.82
Year 2	23.42	3.37	40.62	3.00
Total	23.36	3.42	41.10	2.94

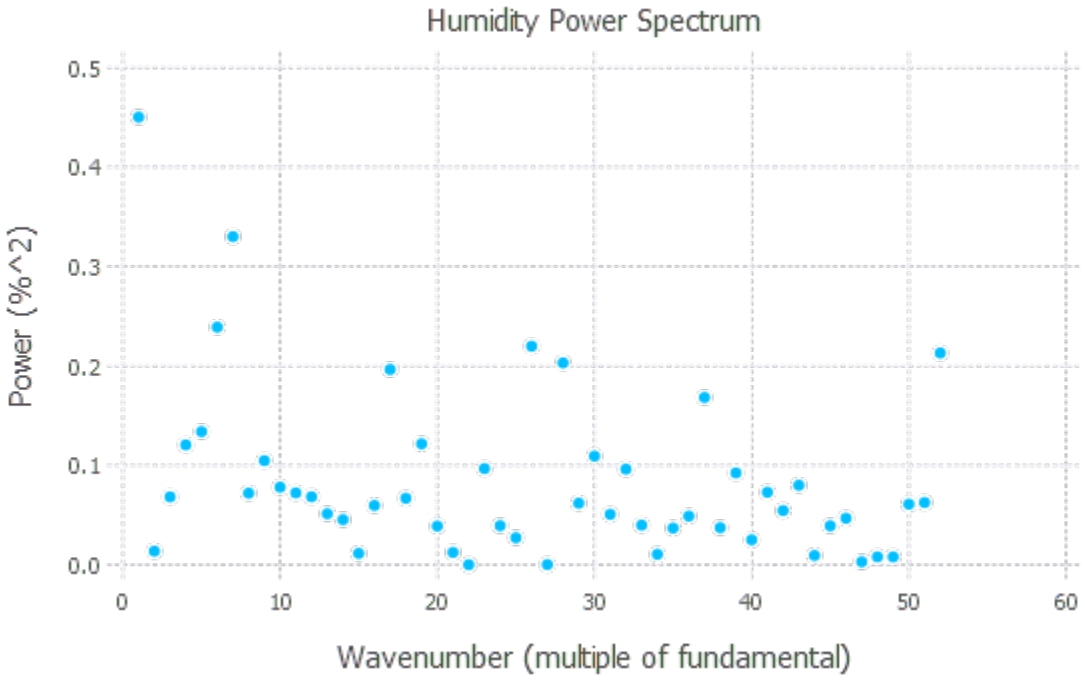
1.2 FFT and Power Spectrum

Taking the fft is done using built in functions. In 1 dimension, the power spectrum is the elementwise square of the absolute value of the fourier transform, normalized by the total number of samples.

$$P_i = \frac{|\hat{f}_i|^2}{N}$$

We plot them for temperature and humidity below.



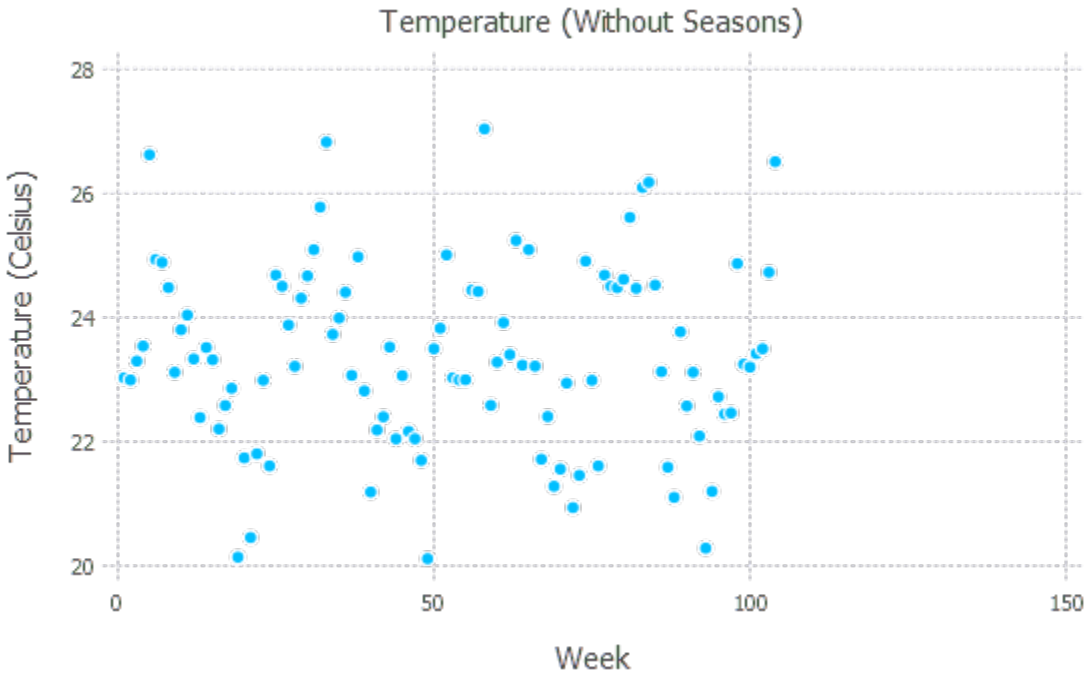


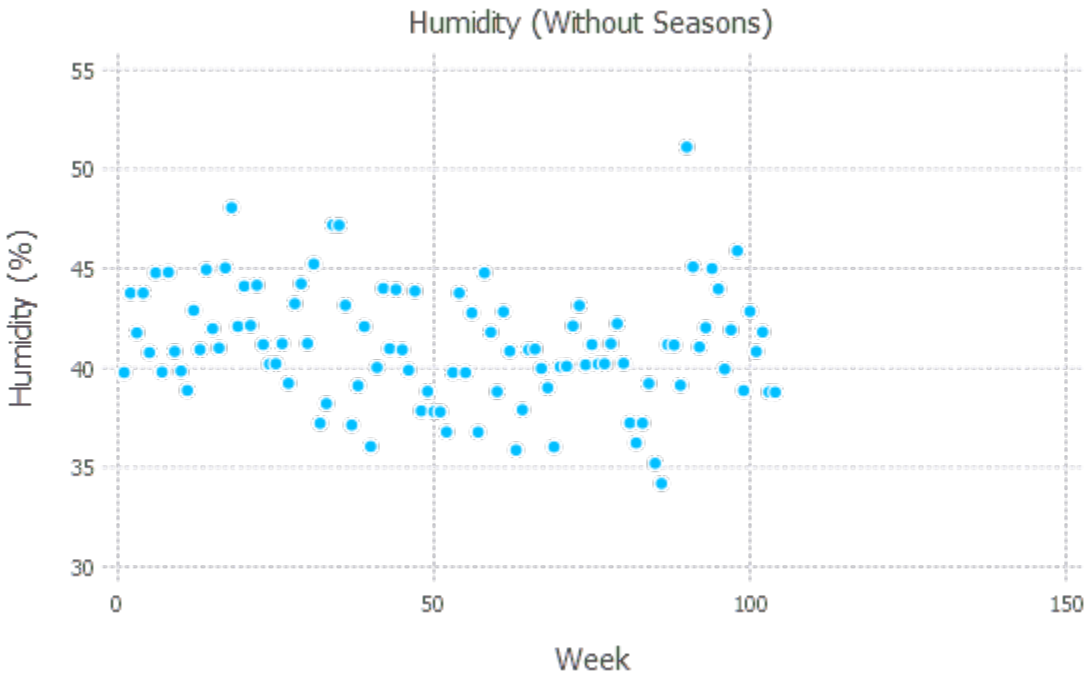
Note that we have hidden the first value of each power spectrum, which would dominate as it is the average of the sampling. Also note that this is just the first half of the power spectrum; however, since our data is real the second half of the fourier transform (corresponding to negative wave numbers) is the same

up to conjugacy, and the second half of the power spectrum would just be the mirror of the first.

1.3 A World Without Seasons

We immediately notice that the largest temperature fluctions comes from the second wavenumber, which corresponds to a frequency with two periods in our window, or a year. Since seasons have yearly period, this is just the seasonal variation. The humidity also has large seasonal variation, although not as drastic compared to other modes. To remove these seasonal variations, we set the term to zero in the fourier transform and calculate the new inverse FFT. This gives us the following:





Now that we've taken out the largest component of the power, both are starting to look more random, bouncing around in a smaller range. Looking at the standard deviation divided by the mean of our new datasets (a standard way to de-dimensionalize), we see that humidity has slightly higher variations.

Variable	Standard Deviation / Mean
Temperature	0.064
Humidity	0.071

However, this is a hard thing to pin down; we could have just as easily, and perhaps more rigorously, looked at the temperature in Kelvin, which is a true unit (goes through absolute 0 at 0), which would have increased the mean but left the standard deviation unaffected. Then, we would have said that temperature had even smaller variations; but this seems unreasonable given that very low temperatures are not something we would consider to be in the realm of the problem (conditions in a room at some given time). We also could have used the range of values sampled as the normalization, but this is susceptible to outliers and closely tied to the standard deviation, so probably is not a good choice. Without a better way to assign dimensionless real numbers to these variances for comparison, we'll have to settle with this answer.

2 Question 2

2.1 FFT

After reading in the data points and reshaping them into a grid, it is easy to take the FFT using built in functions.

2.2 Power Spectrum

To obtain the (linear) power spectrum in 2D requires more care than in 1D. Since the FFT gives us modes with 2 wavenumbers, we want to somehow linearize them according to their "total" wavenumber, which is

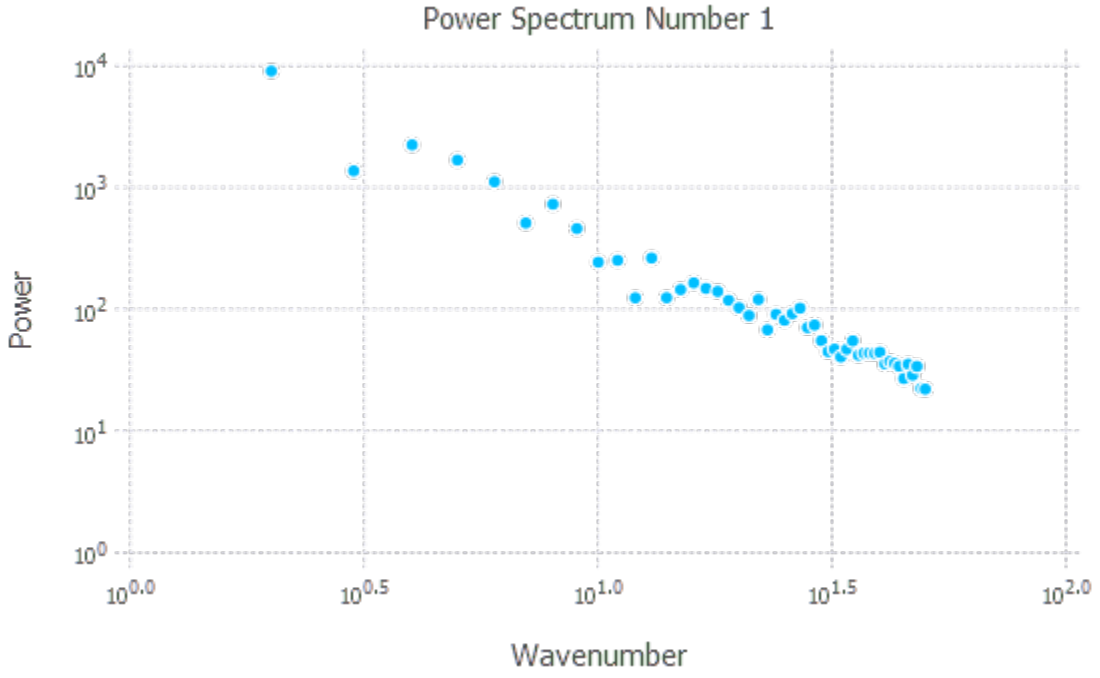
the magnitude of the 2D wavenumber.

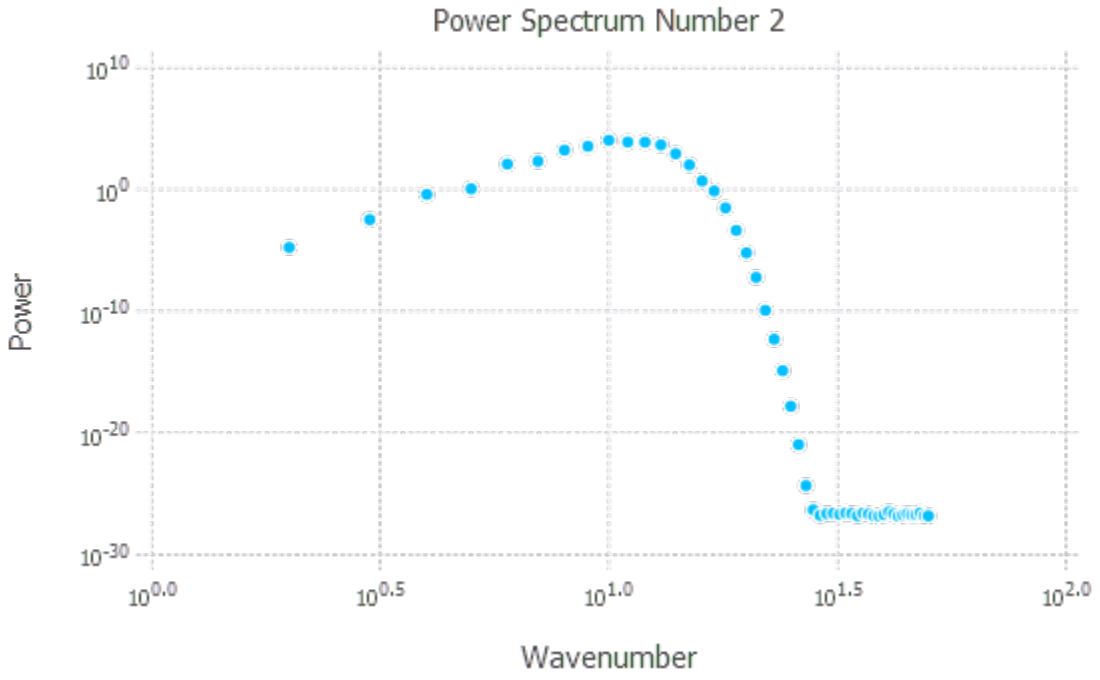
$$k = \sqrt{k_1^2 + k_2^2}$$

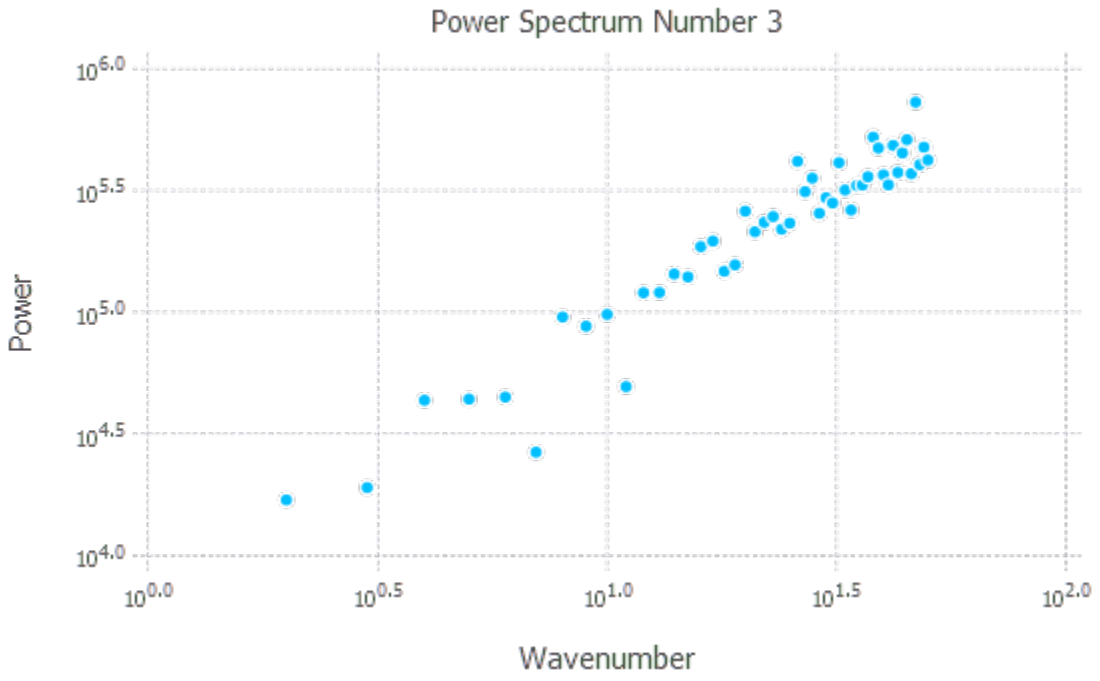
However, taking this gives us some non-integer values, so we want to put them into bins. In particular, we put each one into the bin corresponding to its integer component.

We can also see this by taking integers k from 1 to N_{nyq} (the nyquist frequency), and drawing shells of inner radius k (inclusive) and outer radius $k+1$ (exclusive) from the origin. Taking all the points in those shells and assigning them to the k bin yields the same result. Since the bins will have more and more points as k increases, we want to normalize them to get the average power per mode, so we divide by the number of points in each bin. We could also use the continous approximation, and normalize by $2\pi k$, corresponding to the infinitesimal shell volume $2\pi k dr$ as dr goes to 1; however, this could lead to finite artifacts. Note that shells beyond the Nyquist frequncy go outside of our domain, and therefore we simply drop them (we could still perform the calculation, but it would be much more susceptible to noise and artifacts). Also note that negative wavenumbers in the second dimension are packed in the second half of the FFT (remember only the first dimension is halved), and need to be iterated over as well.

The three power spectra are included below:




Power Spectrum Number 2
Power
Wavenumber
10^{10}
10^{0}
10^{-10}
10^{-20}
10^{-30}
$10^{0.0}$
$10^{0.5}$
$10^{1.0}$
$10^{1.5}$
$10^{2.0}$

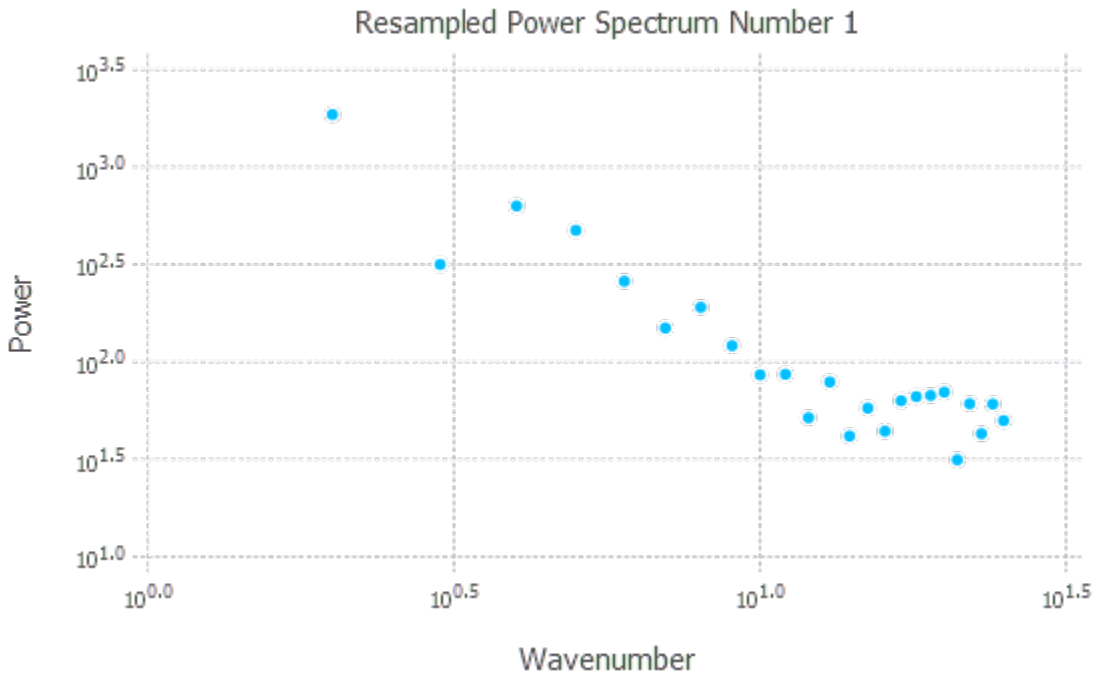

Power Spectrum Number 3
Power
Wavenumber
$10^{6.0}$
$10^{5.5}$
$10^{5.0}$
$10^{4.5}$
$10^{4.0}$
$10^{0.0}$
$10^{0.5}$
$10^{1.0}$
$10^{1.5}$
$10^{2.0}$

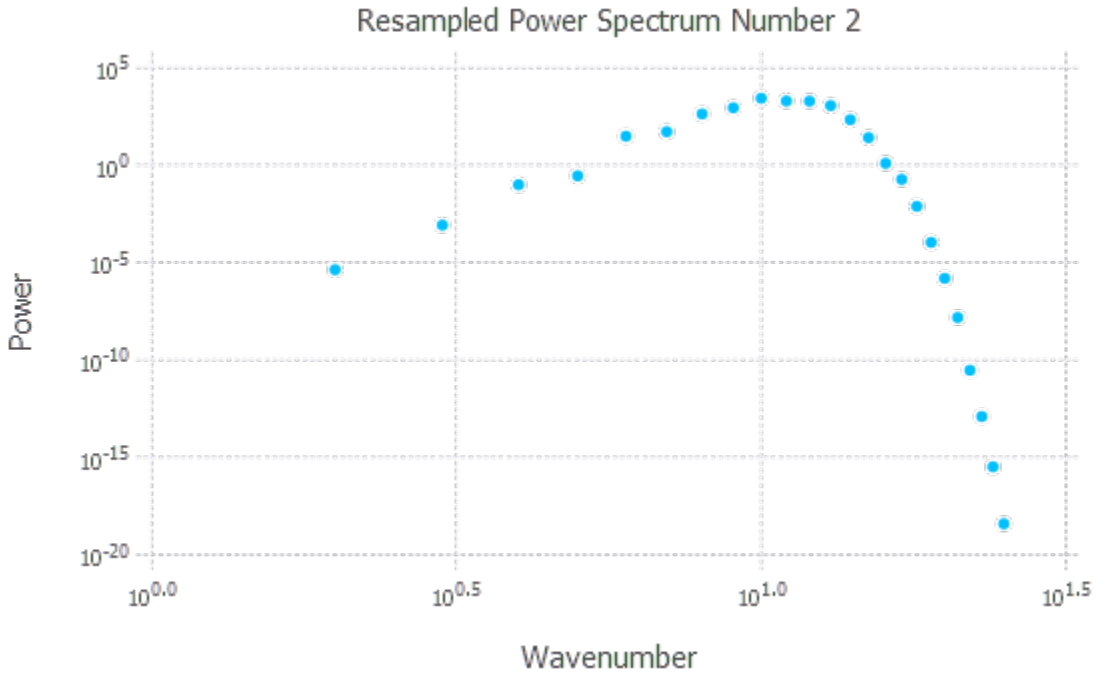
2.3 Explanation

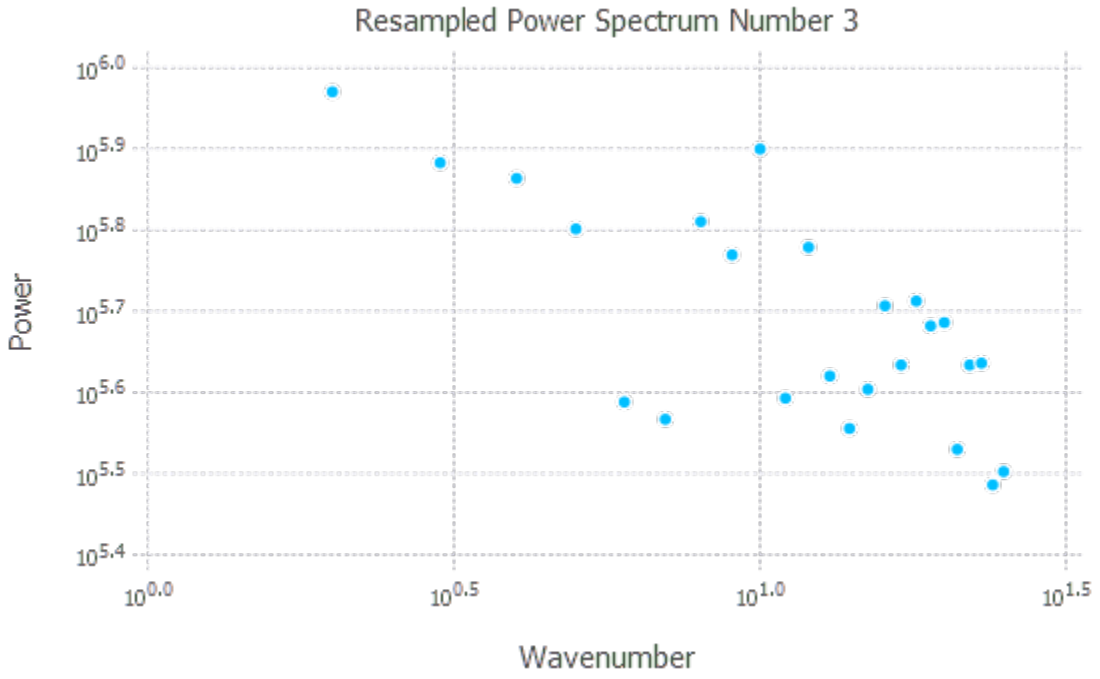
It looks like the first one was generated to be pink noise, decreasing in power as a function of frequency. The second looks to be a gaussian in power as a function of frequency, meaning the original sample was gaussian (the fourier transform of a gaussian signal is itself gaussian). The third looks to be violet or blue noise, increasing in power as a function of frequency.

2.4 Resampling

To re-sample, we simply take every other point on the grid in both directions and put them into a new grid. The three power spectra, resampled at half the grid width, are included below:







Resampling seems to more or less keep the shape of the first two power spectra, but the third one becomes nearly constant. This distortion comes from the resampling; the high end of the spectrum is cut off, and aliasing occurs, shifting the higher wavenumber modes down. This is because the Fourier Transform cannot

distinguish high wavenumber modes in the coarser grid, and mistakes them for low wavenumber modes. This happens when the Nyquist condition,

$$f_{sample} > 2f_{max,signal}$$

is broken, violating the sampling theorem. In particular, the third one suffered from this more than the other two because it had much more power in the high frequency region.

This distortion can be attenuated by first filtering out high modes (using a low-pass filter) before down-sampling. It can also be combated by first smoothing the signal, although this is more common in large, noisy datasets that are expected to be smooth, such as photography.

3 Question 3

3.1 Rayleigh-Levy Walk

We want the complementary cumulative distribution ($1 - C(r)$, where C(r) is the cumulative distribution) to satisfy

$$1 - C(r) = (\frac{r_0}{r})^{\alpha} \qquad r > r_0$$

Cumulative distributions are easy to pick from; every number between 0 and 1 will correspond to exactly one argument in the cumulative distribution. In fact, picking exactly that argument from a uniform random variable between 0 and 1 will lead to the original distribution; this is because the probability density corresponds the the density of random numbers crossed, which is the derivative of the cumulative distribution and therefore the original distribution itself.

Given a random, uniformly distributed number ρ (from Julia's built in Mersenne Twister), we calculate the radius of our step

$$r = r_0(1 - \rho)^{-\frac{1}{\alpha}}$$

We also calculate the direction using two more random numbers, θ and ϕ (both from 0 to 2π), and use those to make a unit direction vector in 3 dimensions in the standard way

$$\hat{r} = [cos(\theta)sin(\phi), sin(\theta)sin(\phi), cos(\phi)]$$

Multiplying this unit vector by our scalar step length gives us our step, which we then add on to our previous point. Staring from a random seed point and iterating in this way, we generate the 500 walks of 100 steps each, taking care to loop around points that wander outside the box (due to periodic boundary conditions).

3.2 CIC Density

After some geometric reasoning, we see that for any particle at (x, y, z), we simply add to each of the eight grid cells surrounding it the particle density given by

$$p = \frac{(d - |a - x|)(d - |b - y|)(d - |c - z|)}{d^3}$$

Where d is the grid step, and the grid point's position is give by (a, b, c). We can de-dimensionalize this by dividing one of the denominator's d's into each of the factors, giving

$$p = (1 - |\hat{a} - \hat{x}|)(1 - |\hat{b} - \hat{y}|)(1 - |\hat{c} - \hat{z}|)$$

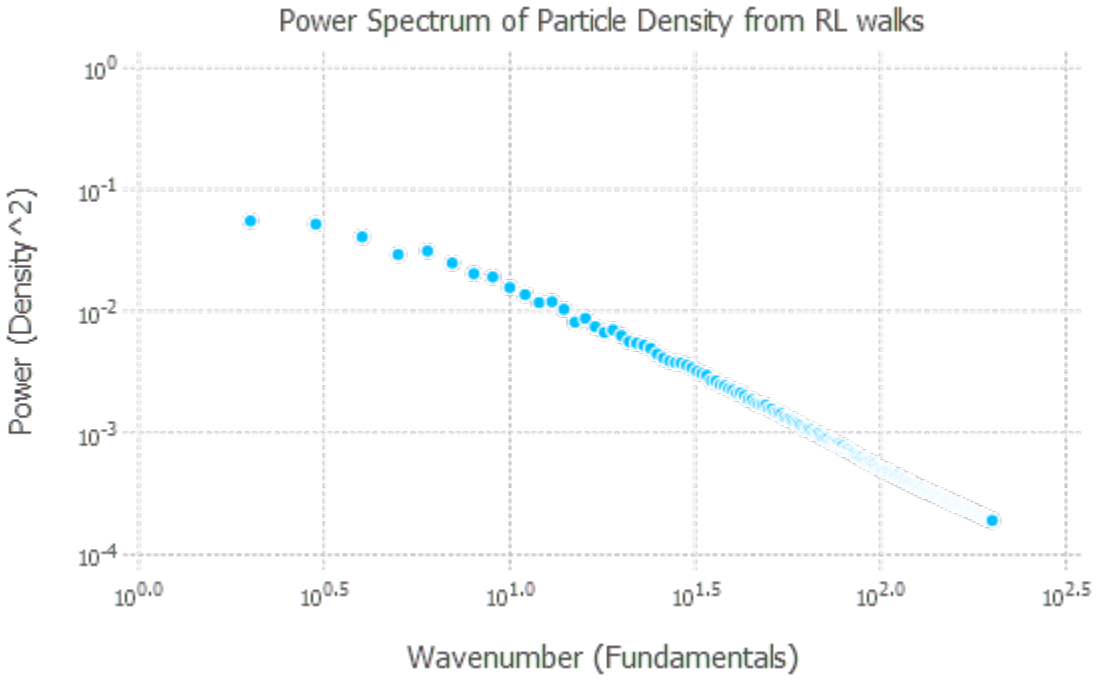
which is much easier and more natural to compute on our discrete grid. To complete this, we simply iteratre through each particle, algorithmically finding out the eight nearest points (keeping the periodic boundary conditions in mind, since the some of them may cross the boundary), and then assigning each one this fraction. To check, we sum over our grid and find the we do, indeed, get exactly 50,000 particles back.

3.3 FFT

Again, the FFT is trivial to compute using Julia's built in library. Note that Julia uses FFTW underneath!

3.4 Power Spectrum, CIC Correction

We can extend our 2D power spectrum algorithm to 3D by considering 3D wavenumbers instead. This produces the following power spectrum from our random walk particle density



This looks like Brownian Motion noise (steady fall off), which is a good sign, but not quite what we're looking for. We still need to correct for the CIC interpolation window, namely its distortion in our spherical sampling method. In 1D, we know we can correct this using

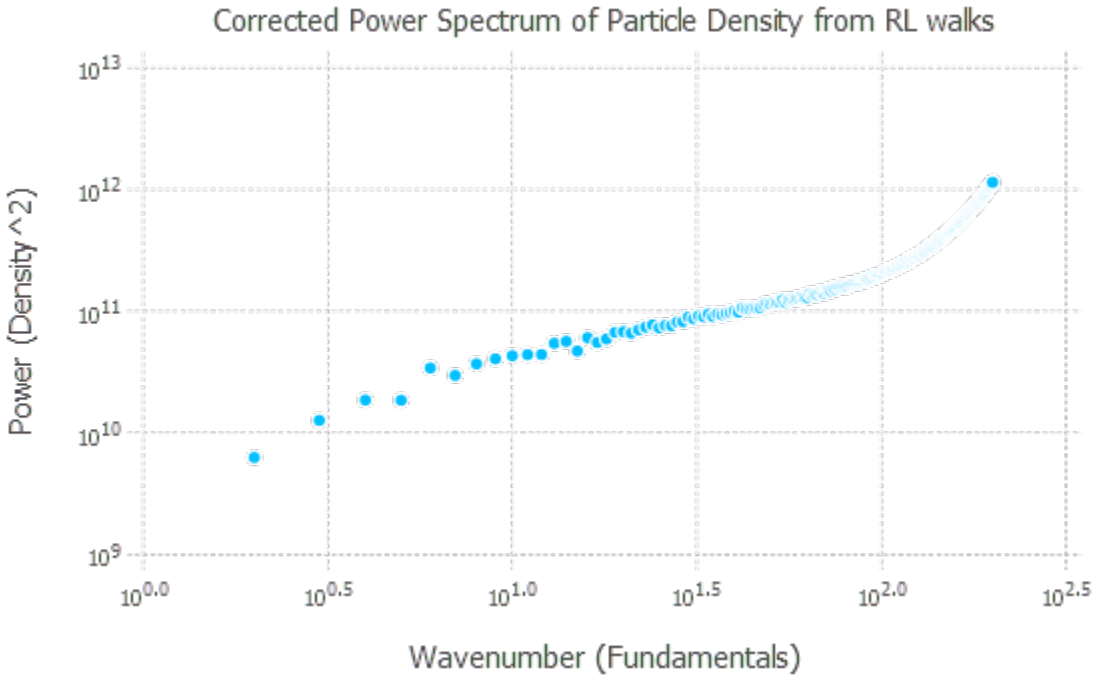
$$P_{true}(k) = \frac{P_{calc}(k)}{|\hat{w}(k)|^2} \qquad \hat{w}(k) = |\frac{sin(k/2)}{k/2}|^2$$

This extends straightforwardly to 3D, with

$$\hat{w}(k) = \hat{w}(k_x)\hat{w}(k_y)\hat{w}(k_z)$$

Note, however, that we now need to apply this at every point of our FFT before we include it in the power spectrum, since different points are subject to different windows. Also note that we de-dimensionalize this in the computer so we can work with wavenumbers,

$$\hat{w}(\hat{k}) = |\frac{sin(\pi\hat{k}/L)}{\pi\hat{k}/L}|^2 \qquad \hat{k} = \frac{k}{k_{fund}}$$



3.5 Explanation

Rather than the brownian noise we saw before, this better shows the features of our walk. In the low wavenumbers the features look nearly random, like constant power white noise; and, indeed, they should be, since the large scale structure is just the different clusters of walks, which were randomly seeded.

However, as the wavenumber increases, corresponding to shorter wavelengths, we start to see our particle generation walks, which give rise to high power features of high wavenumber near the end of the graph. In particular, since our $r0$ was on the order of our grid size, we see these features at the very end of the spectrum. The grid size was less than half of $r0$ so the full feature is still chopped off a little, but we see the sharp peak that comes from the intra-walk features rise 10dB.