PHY407 Lab05

(Yonatan Eyob Q1), (Kivanc Aykac Q2)
Student Numbers: 1004253309, 1004326222

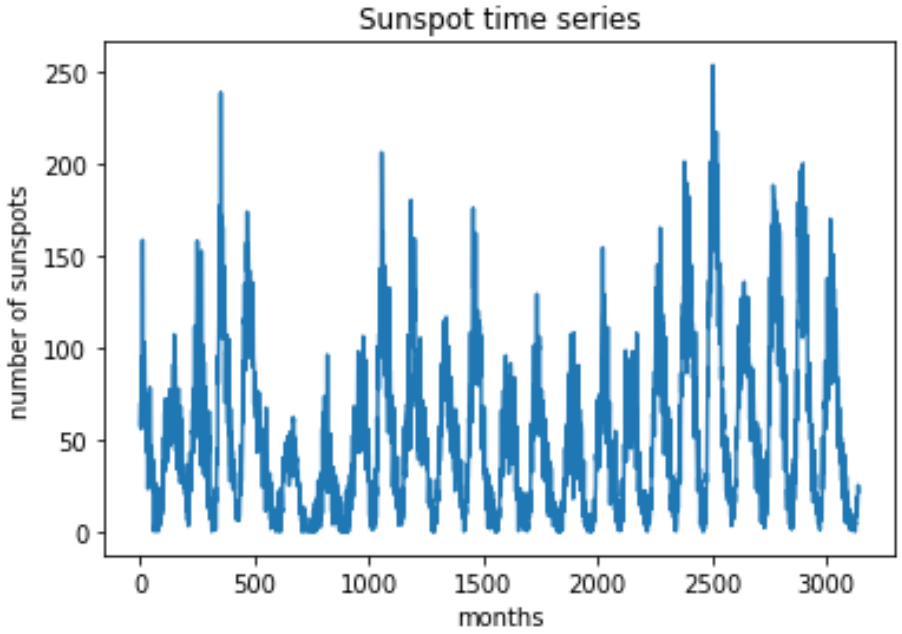
October 16^{th}, 2020

1 Basic applications of the Fourier transform

1.a

In this section we will collect data from the sunspot.txt file and plot it as a function of time, and then we will plot the Fourier coefficients of the sunspot graph as a function of cycles and use this graph to determine the period of the sunspot time series graph.

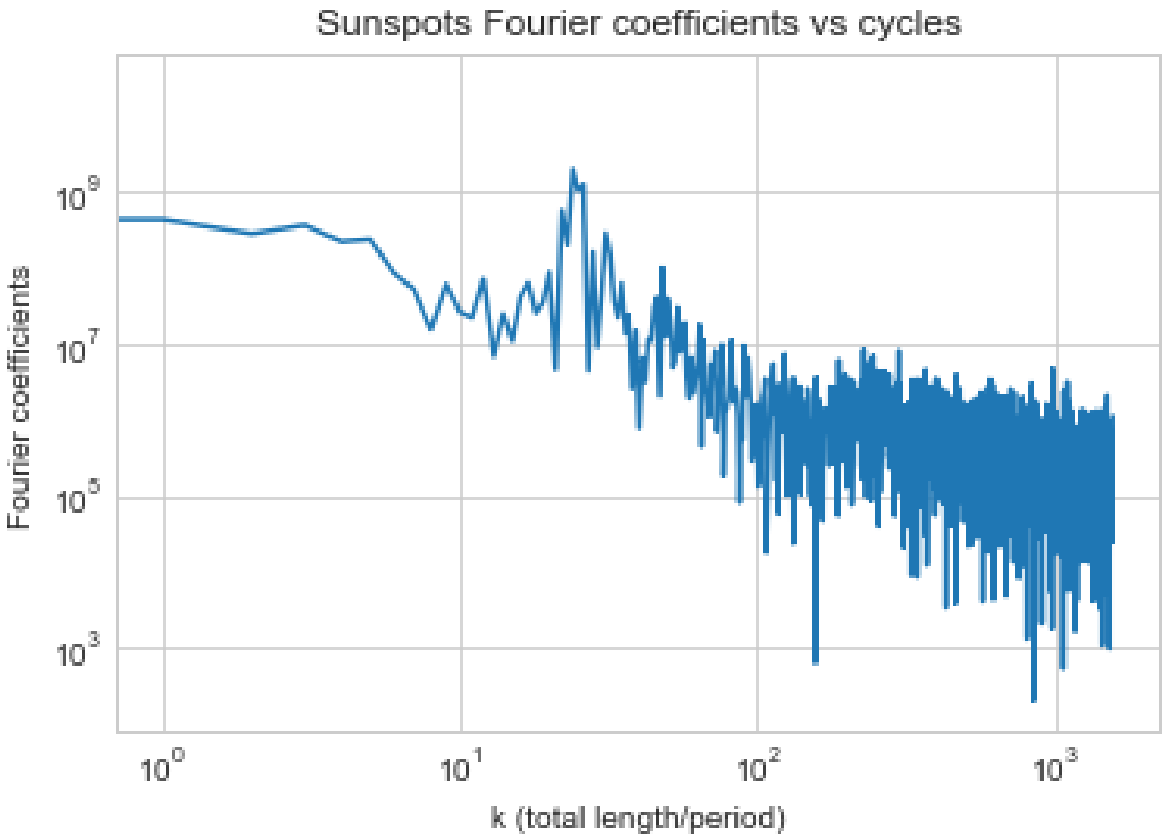
1.a.i



The function goes through about 4 cycles between 1000 and 1500 earth-months

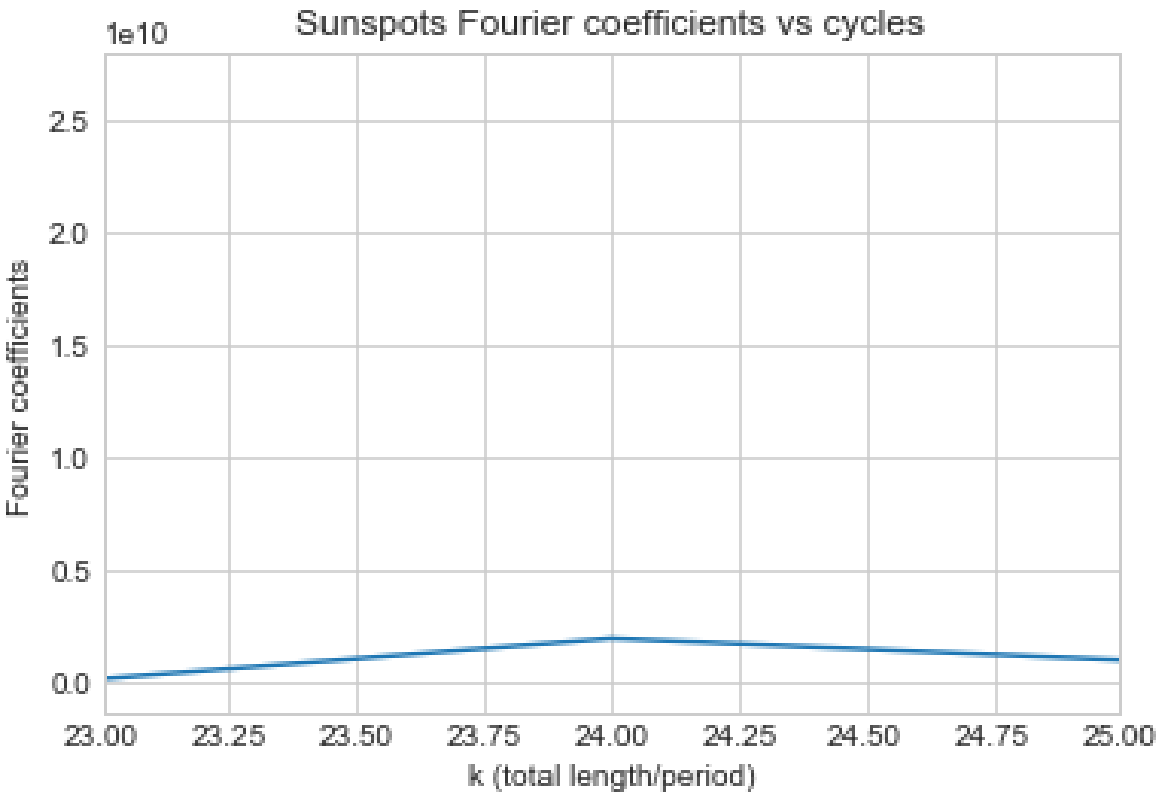
so the period seems to be about 125 earth-months long.

1.a.ii



this is the loglog version of the graph
The Fourier coefficient vs cycle (k) graph above has an obvious large peak which must be corresponding with a large sine-wave pattern we see in the sunspot time series plot. The cycle (k value) at which this peak Fourier coefficient belongs to should be the cycle of the sunspot time series graph.

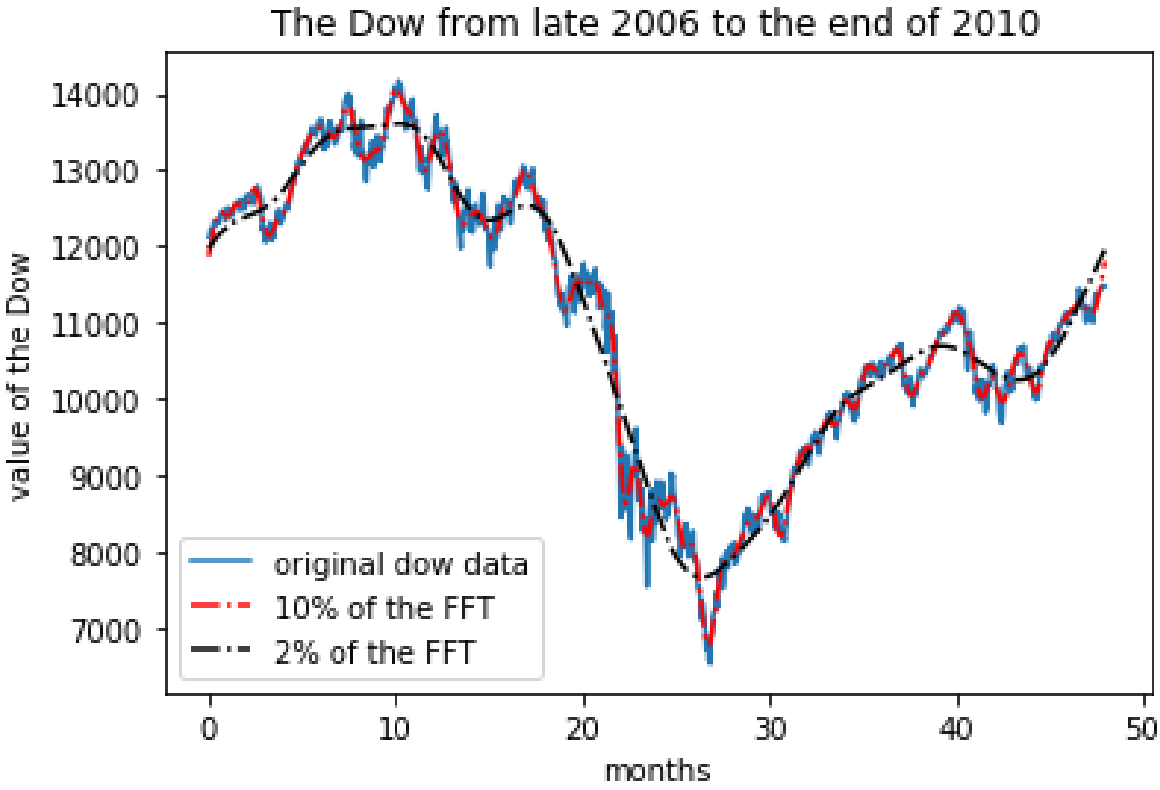
1.a.iii



Zooming into the Fourier coefficients vs k graph above, we can see that the peak occurs at k=24. Since the periodic function is $e^{-i2kx/N}$, we can let the period of this function equal N/k. In this case, N=3143 and k=24 so period = 3143/24 $\approx$ 130.9583 months which is close to the original guess.

1.b

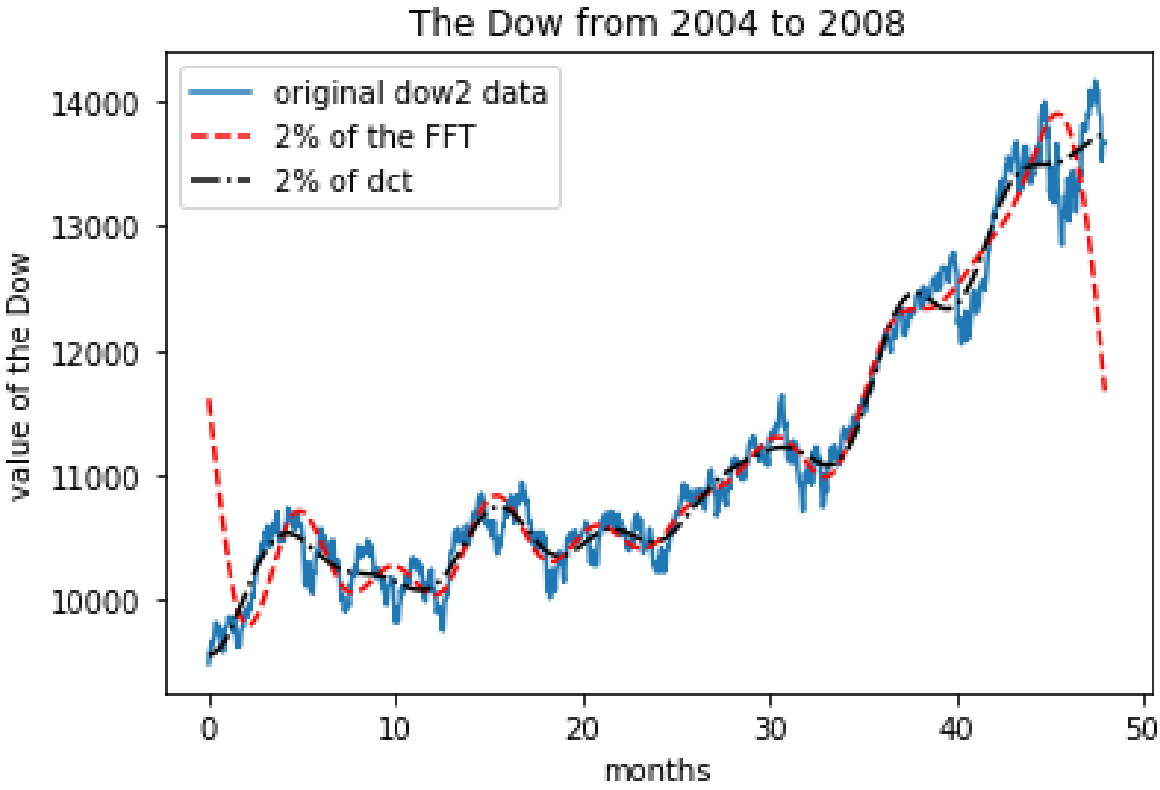
In this section we will collect data from the dow.txt file, take the Fourier transform of the original data using FFT, set the all the Fourier coefficients to zero except for the first 10 percent and 2 percent, then compare the original data with the inverse Fourier transform of the two adjusted Fourier coefficients.



The graph above suggests that lowering the percentage of non-zero Fourier coefficients makes the graph smother but the lower the percentage, the more the resulting plot deviates from the original data. This makes sense because setting Fourier coefficients to zero is removing oscillations frequencies from the original data. If all the Fourier coefficients were set to zero then the graph would be constant.

1.c

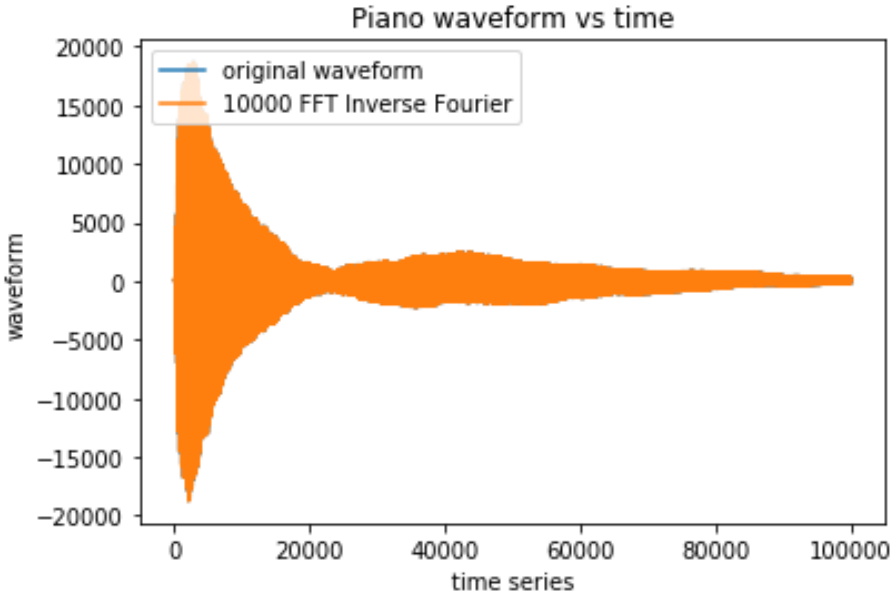
This section is similar to 1c, but instead we will be collecting data from the dow2.txt file, take the Fourier transform using FFT, set all the elements to zero except for the first 2 percent, then compare the inverse Fourier transform to the original data by graphing them, and then do the same thing using the discrete cosine transform (dct) instead of FFT.

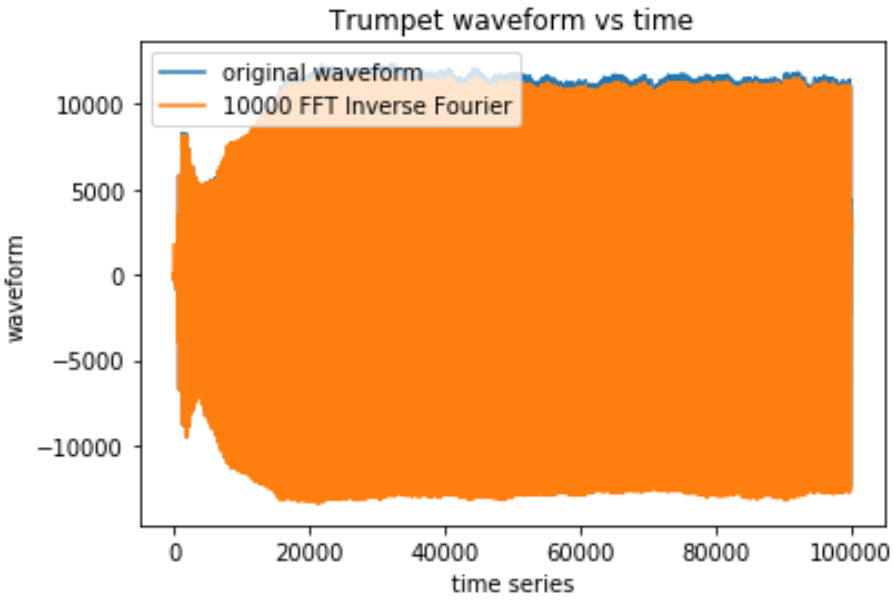


The inverse FFT deviates a lot from the original data at the beginning and end of the graph because it is required to be periodic and therefore must begin and end at the same y-value. The inverse dct doesn't demand this so the inverse dct follows the original data more accurately than the inverse FFT at the beginning and end of the graph.

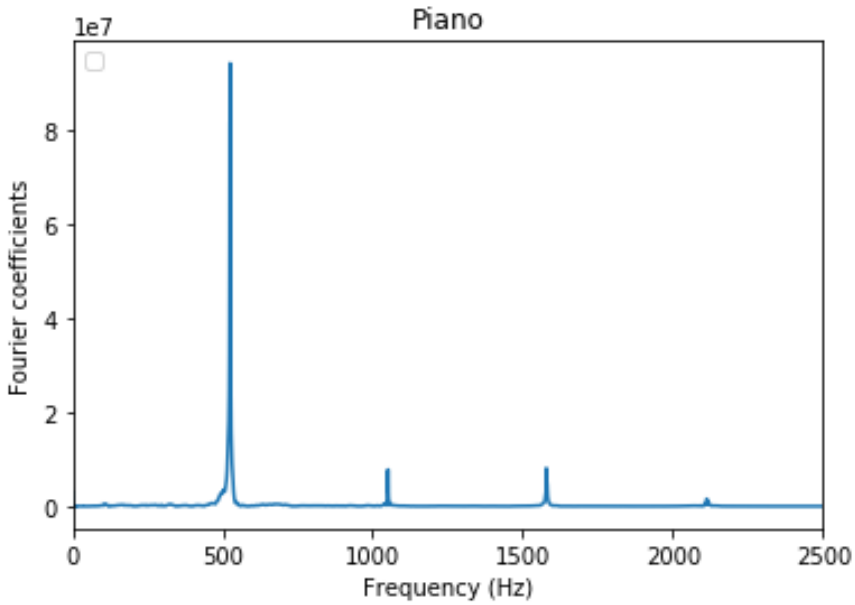
1.d

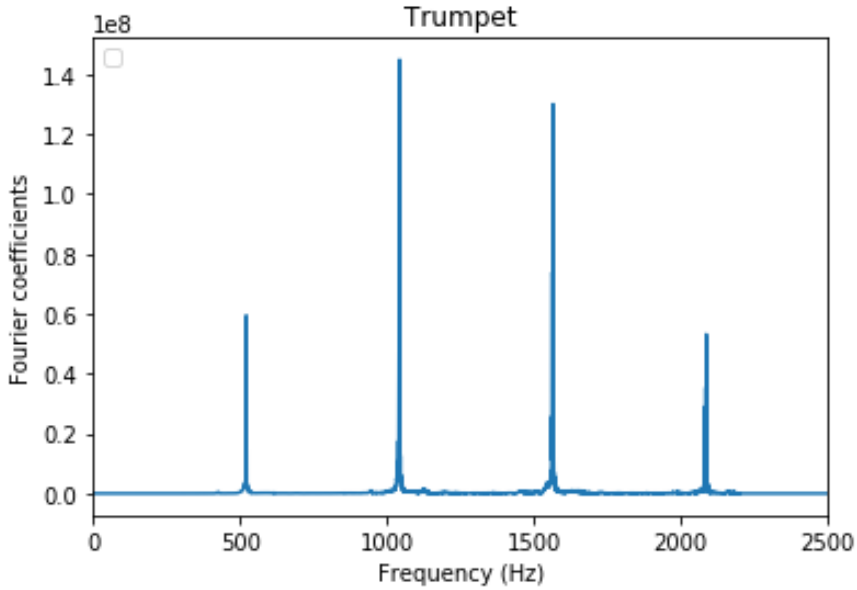
Now we will plot the wave forms of the note played by a trumpet and a piano and attempt to determine which note they are playing by observing their Fourier coefficients vs frequency graphs below:





When we set all the elements of the FFT to zero except for the first 10000 coefficients and plot the inverse, the graph doesn't change much from the original waveform graph as shown in the graphs above. This tells us that the large elements of the FFT are all within the first 10000 elements, because if that was not the case, the inverse Fourier graph would deviate from the original waveform graph a more noticeable amount. Thus the majority of the sound being played by the trumpet and piano are all below a certain frequency threshold.





Looking at the Fourier vs frequency graphs above, assuming these instruments are tuned for A_4 (440Hz), it is clear that both the trumpet and the piano are playing the C note in different octaves. The trumpet's most dominant frequency is approximately 1044Hz, which is about C_6, and the piano's most dominant frequency is approximately 525Hz, which is about C_5.

2 Image Deconvolution

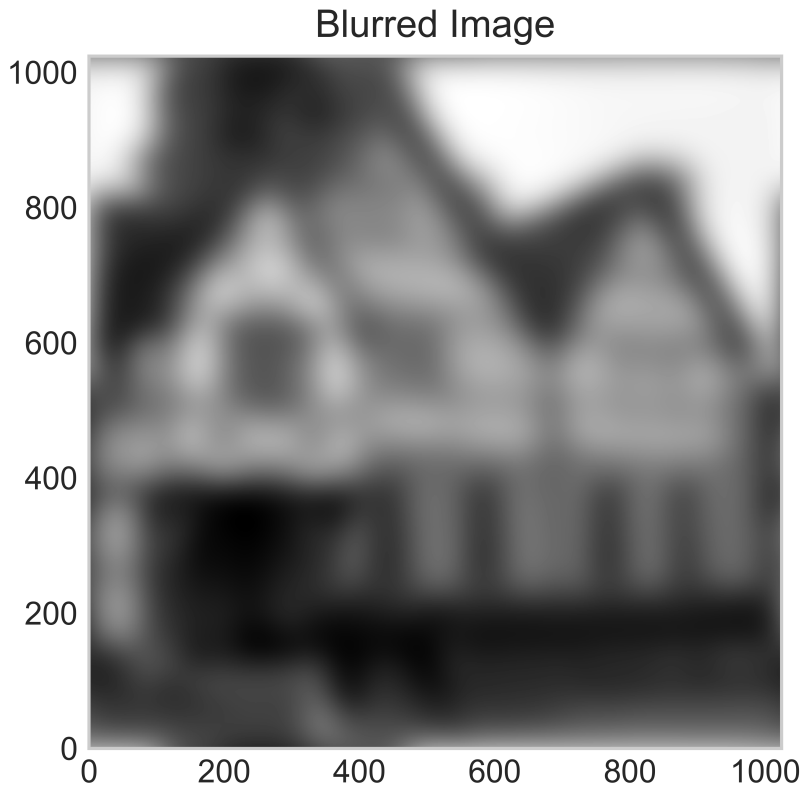
In this section, goal is to use to apply deconvolution on a blurred image using the Gaussian with $\sigma = 25$. Gaussian to be used is:

$$f(x, y) = exp(-\frac{x^2 + y^2}{2\sigma^2}).$$

For simplicity, only a single function that gives brightness values is going to be used, so the images will be black and white.

2.a

In this part, the blurred photo is going to be plotted on black and white color scheme. The data for the brightness values of the image is retrieved from the website.* Below is the image got from directly plotting the data.

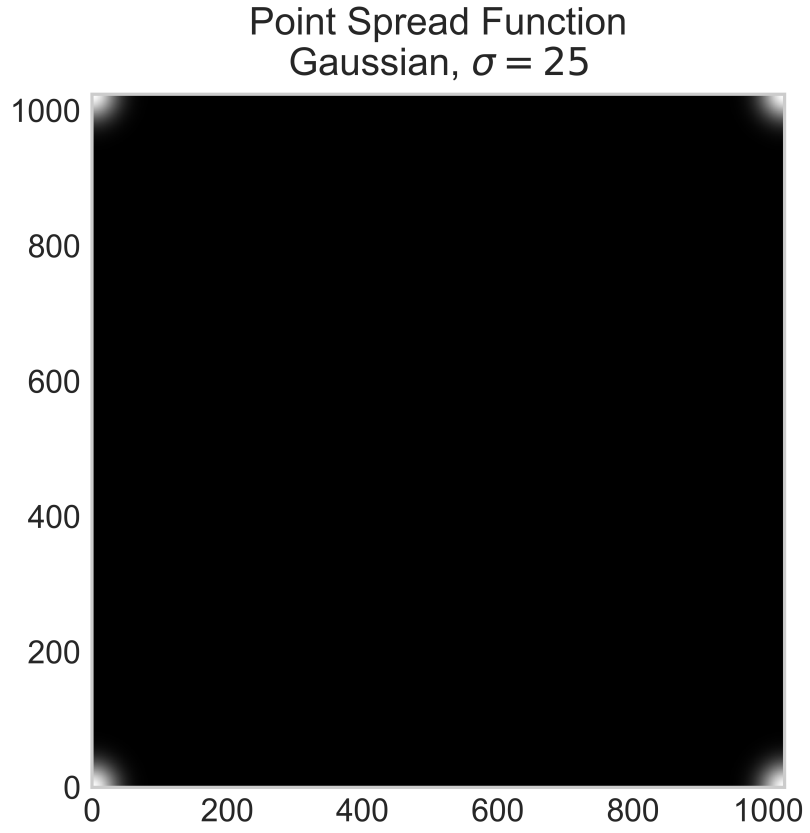


∗. www-personal.umich.edu/ mejn/cp/data/blur.txt

As can be seen, the image is resembling three houses side by side and not much if it is squinted at by the reader. This is the blurred image.

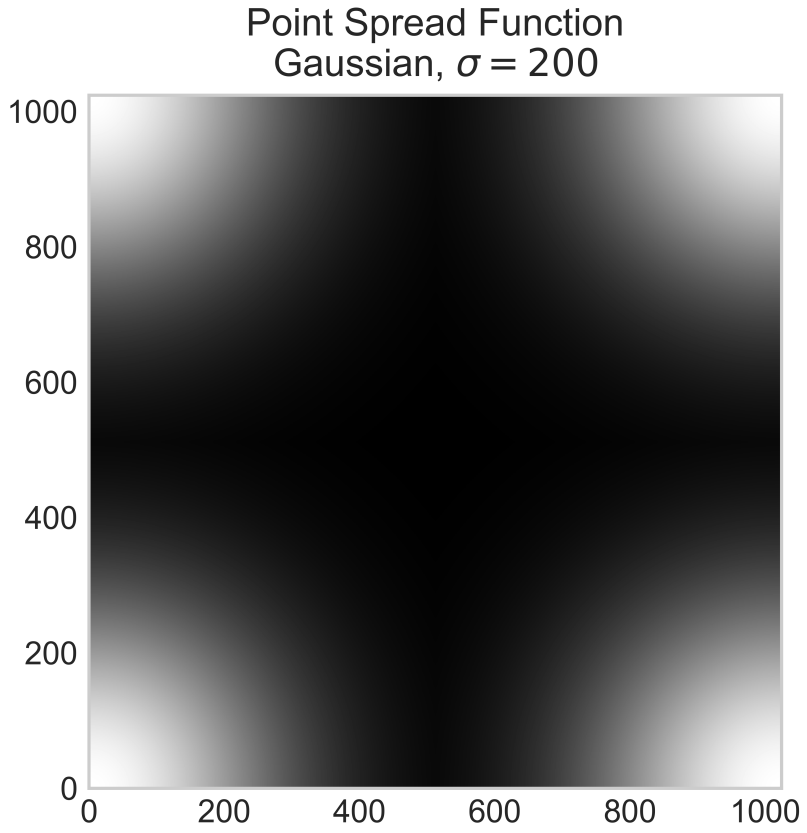
2.b

In this part, the Gaussian function that is going to be used as the point spread function is going to be plotted. From the text, it is understood that the author had a sharp and clear image at their hand and blurred it using Gaussian function with $\sigma = 25$.[†] Therefore, to unblur the image, Gaussian with width $\sigma = 25$ is going be used.
When working with Fourier Transforms with `numpy.fft`, it is important to check to see if the Gaussian matches (coordinate wise) that is going to divide the Fourier transform of the brightness values. The Gaussian is periodic along both axis, so should get bright patches in each of the four corners of the picture of the point spread function. Below is the plot to check this aspect:



†. To learn more about blurs and filters: www.youtube.com/watch?v=C$_z FhWdM4icab_channel = Computerphile$

Since the bright patches on this plot is hardly visible, the width of the Gaussian is going to be exaggerated and set to $\sigma = 200$. Below is the picture of the point spread unction with exaggerated σ:



As can be seen, picture passes the check and the point spread function is correctly adjusted to match the appropriate brightness values in the next part.

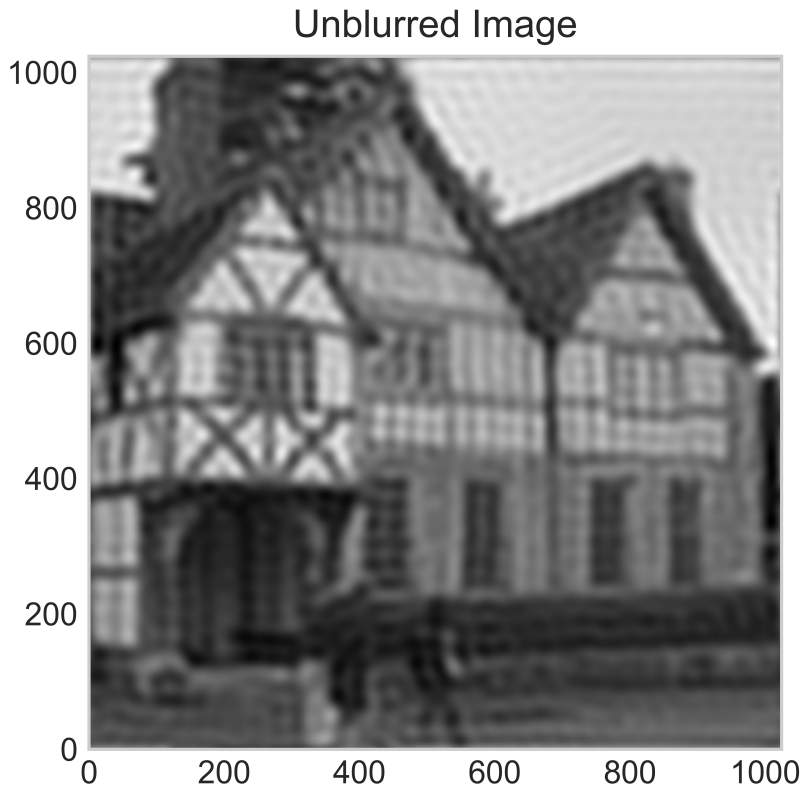
2.c

In this part, the goal is to use the skills gained from the preceding two parts in order to unblur the photo. The formula

$$\frac{\tilde{b}_{kl}}{KL\tilde{f}_{kl}} = \tilde{a}_{kl}$$

Tilde represents the Fourier coefficients (b_{kl} for brightness value at k^{th} row and l^{th} column) and L and K represent the x-axis size and y-axis sizes, respectively. To get Fourier coefficients, `numpy.fft.rfft2` is used (2-dimensional real

fast Fourier transform). Then to avoid division by zero, a simple if statement is used to check if any value of $\tilde{f}_{kl}$ is less than $\epsilon = 10^{-3}$ and equalize $\tilde{a}_{kl}$ to $\tilde{b}_{kl}/KL$ in any such circumstance (this is acceptable since the effect will not be visible clearly on the end result). After finding the values of $\tilde{a}$, only thing to do is to get the inverse Fourier transform of it. To do so `numpy.fft.irfft2` is used (2-dimensional inverse real fast Fourier transform). Below is the final finding that gives the unblurred picture:



As can be seen, when compared to the blurred image above, this is a much sharper image. And the process of manipulating the Fourier transforms before getting the inverse Fourier transform is working. Now the house and the people on the street are much more visible.