Assignment 5: Computing Fourier Transforms

Question 1

a) Using FFT to determine period

Below is a plot of the number of sunspots observed each month since January 1949. Qualitatively speaking, the period appears to be 125 months. Figure 2 shows a plot of the magnitude squared of the Fourier coefficients. Noticeable peaks occur at k = 0 and k = 24. Given that there are 3000 data points, this gives us a period of 3000/24 = 125, as expected.

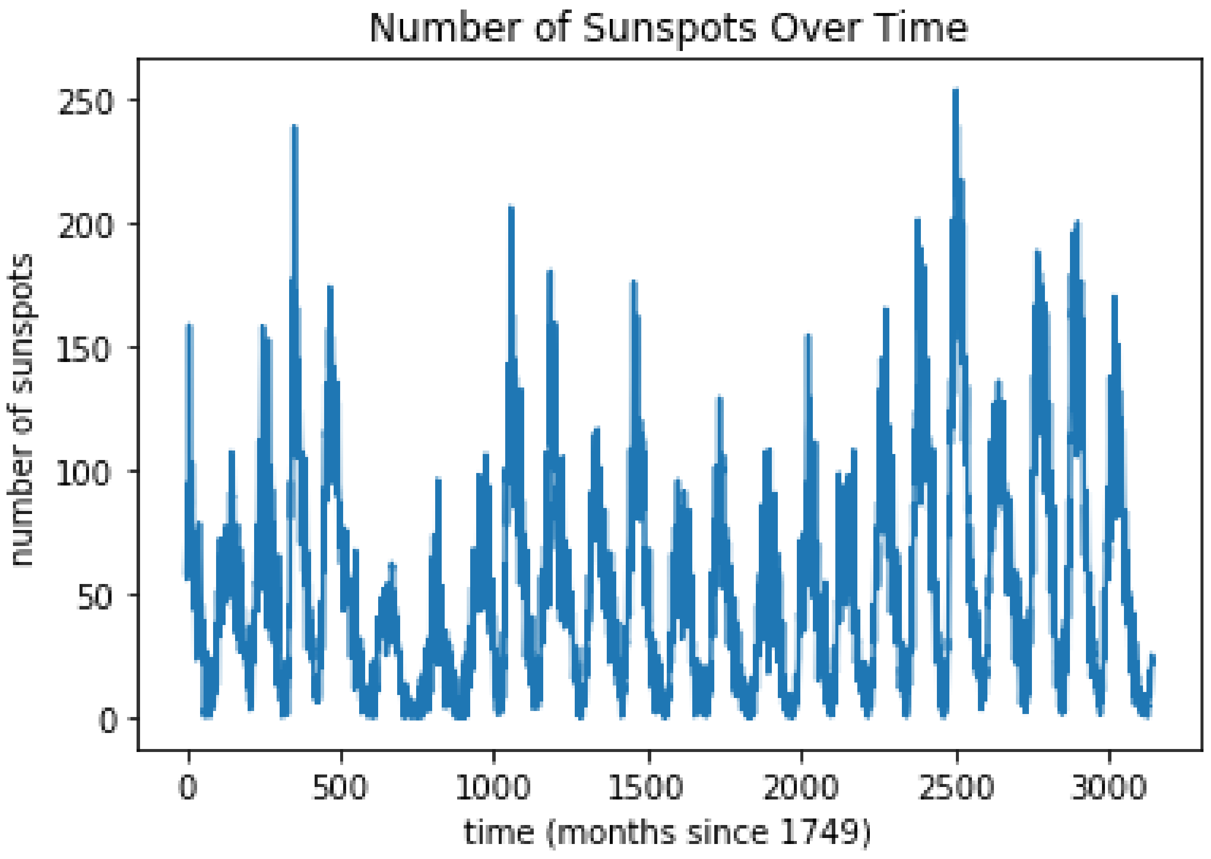

Figure 1. There seems to be 4 periods every 500 months, or a period every 125 months.

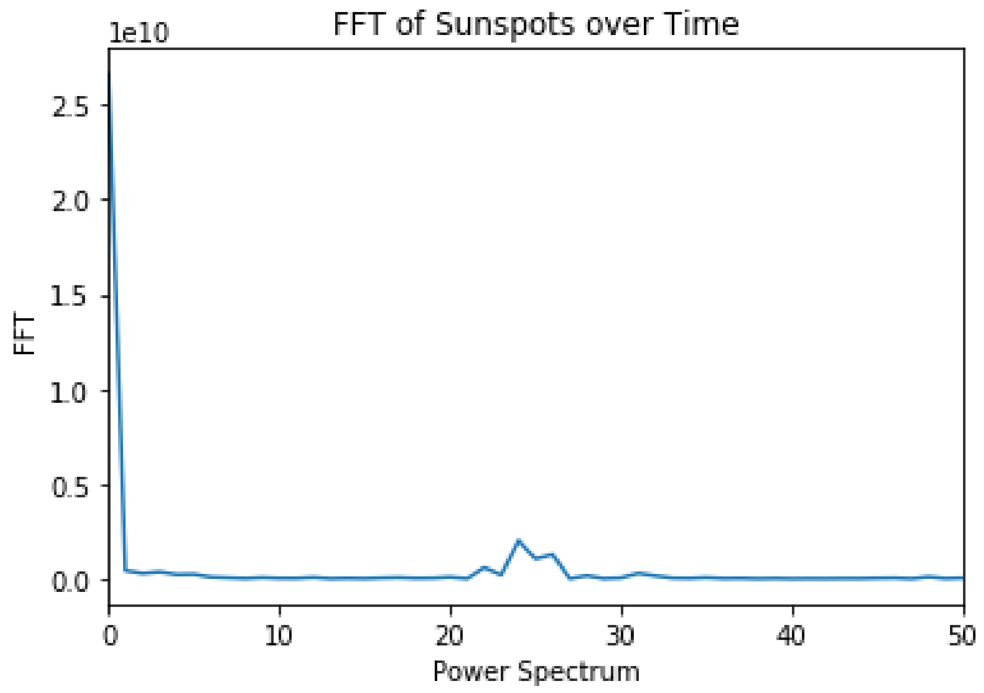

Figure 2. Zoomed in plot of FFT. Ignoring the peak at k=0, another peak occurs at k = 24. Since we have 3000 data points, 3000/24 = 125, which exactly corresponds to the period we previously calculated.

b) Filtering and smoothing plots with FFT

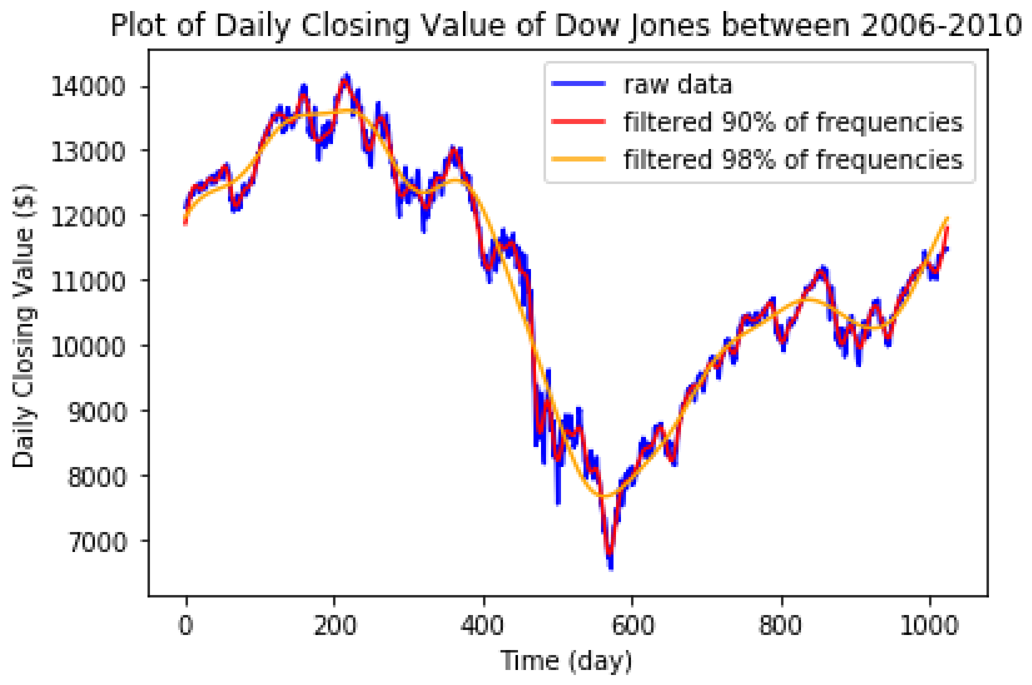

Figure 3. Plot of the daily closing value of Dow Jones over time. The plot contains three curves – in blue is the raw data; in red is the inverted FFT for data in which the last 90% of the FFT has been set to zero; in orange is the inverted FFT for data in which the last 98% of the FFT has been set to zero.

By setting the Fourier coefficients to zero for higher-order frequencies, we are ignoring the contribution of the Fourier eignestates corresponding to those frequencies. This filters out the [high-frequency] noise seen in the plotted raw data, resulting in a smoothened curve. Setting the last 90% of the data to zero (p = 10), the plot has undergone some noise reduction, but still follows much of the original plot's trends. In contrast, by setting the last 98% of the data to zero (p = 2), we are left with only the most general trends of the original curve.

c) Filtering and smoothing plots with FFT continued

As before the closing value of Dow Jones as plotted, this time between 2004 – 2008. The last 98% of the FFT was set to zero before the inverse FFT was taken. This plot is slightly less periodic than before in 1c) as there is significant change from t=0 to t=1000. Smoothing using inverse FFT doesn't appear to work well as before, particulalry near t=0 and t=1000, as seen in Figure 4. Better results appear to be obtained when inverse cosine FFT is used, however this method only loosely captures the original data from t=0 to t=1000.

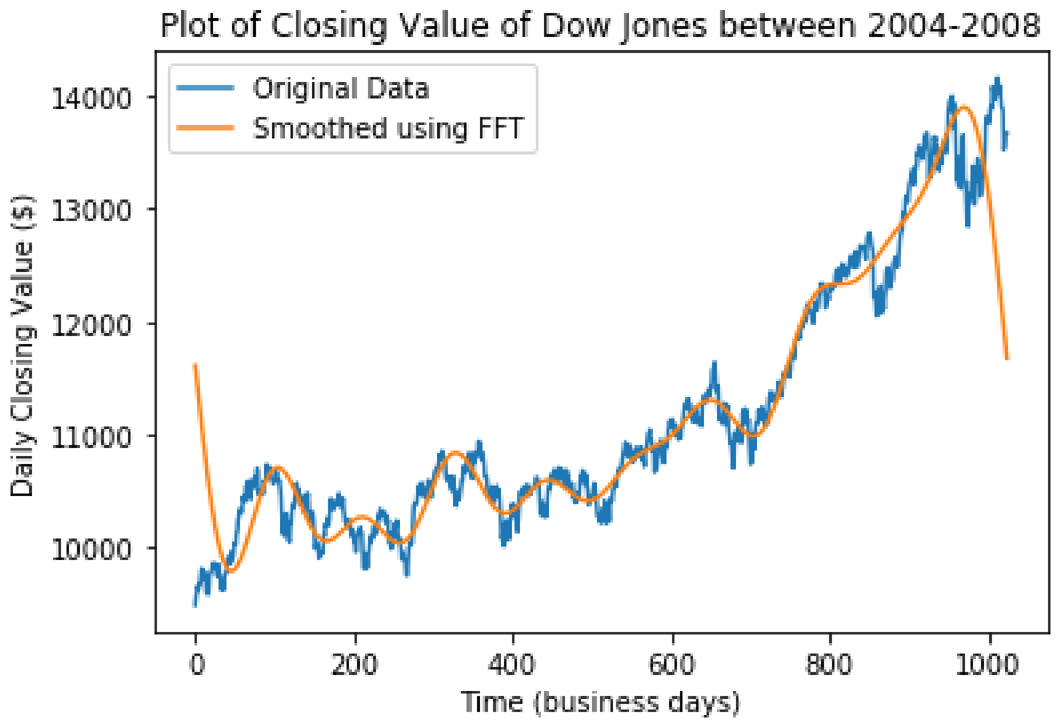

Figure 4. Smoothing using inverse FFT is not as accurate as in 1b), t = 0 and t = 1000 do not correspond well to the original data.

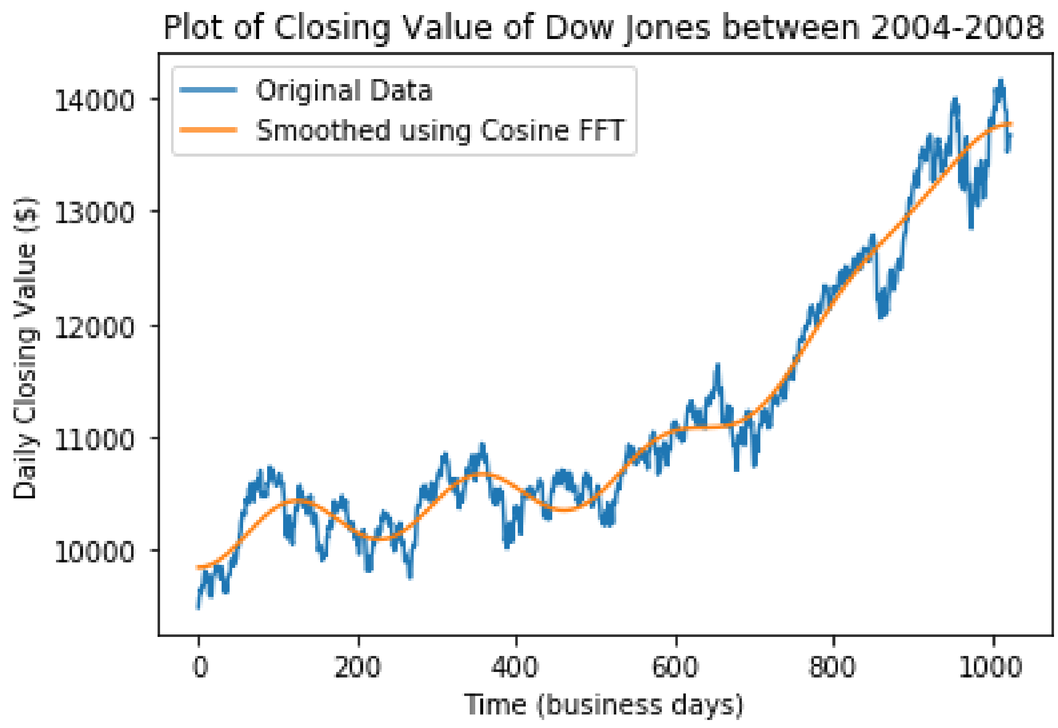

Figure 5. Smoothing using inverse cosine FFT.

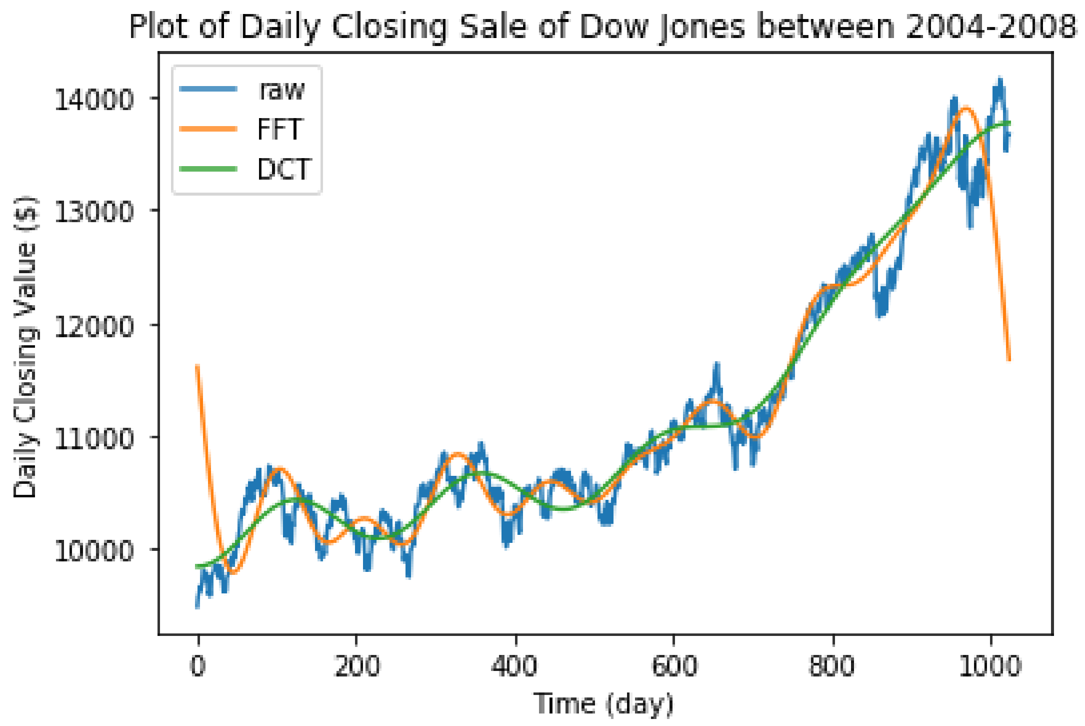

Figure 6. Plot of inverse DCT and inverse FFT over the original data.

Denisa Bani 1001330981; completed Q1 a) and c); collaborated on Q2
Brian Xuan Viet Nghiem 1002551589; completed Q1 b) and d); collaborated on Q2

d) Spectral analysis of musical instruments

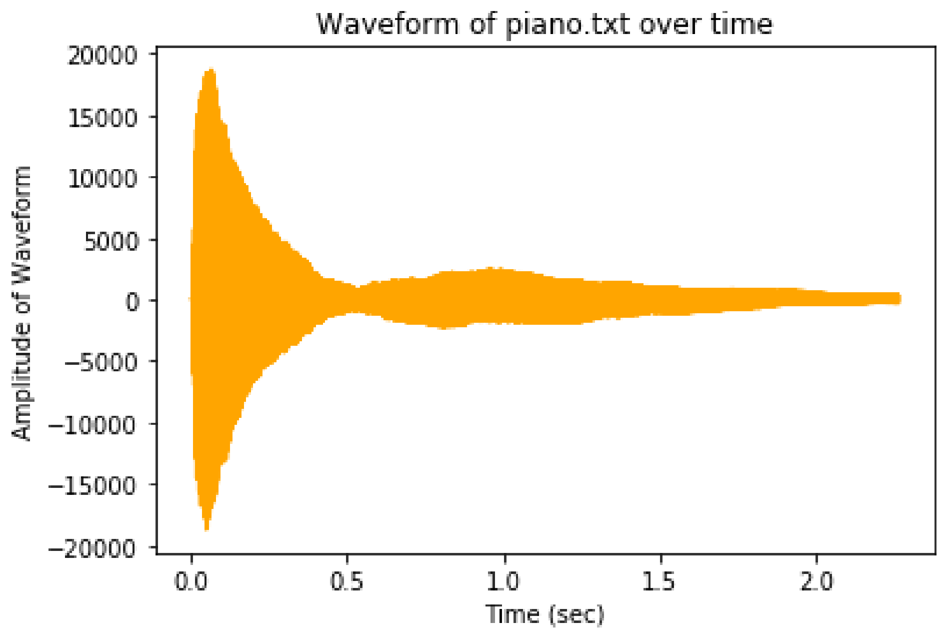

Figure 7. Plot of the waveform of the piano data over time.

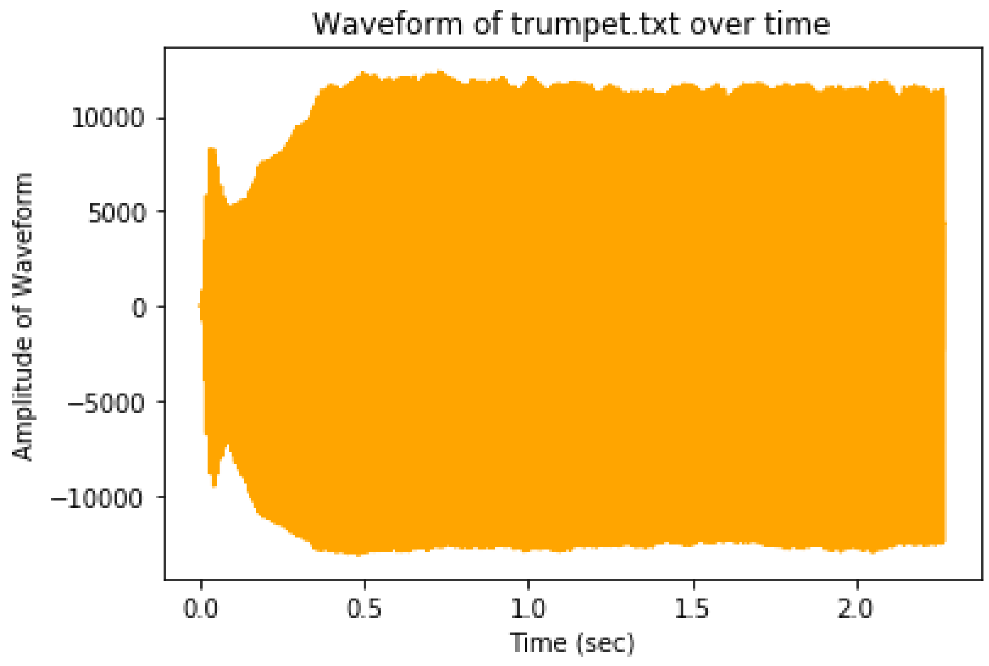

Figure 8. Plot of the waveform of the trumpet over time.

Comparing the waveforms of the piano and the trumpet, they are clearly different. The shape of the piano waveform shows that it experiences a significant amplitude decrease over time, corresponding to the sound decay of the instrument that occurs after you strike a piano key. In contrast, the amplitude of the trumpet waveforms are relatively steady after ~0.5 seconds, which is due to the fact that the note being played is sustained as long as the player steadily blows air into the instrument.

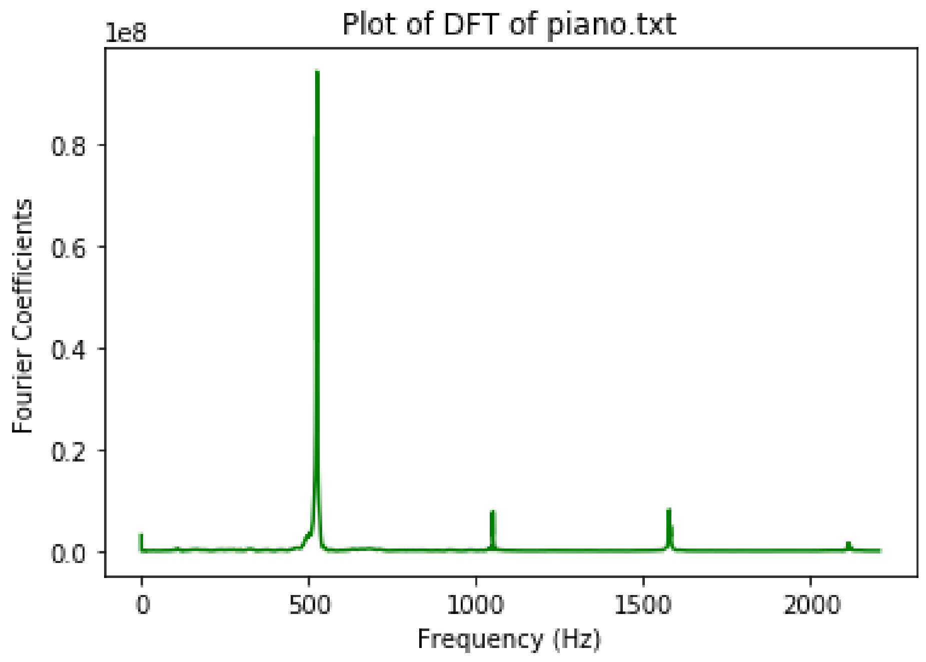

Figure 9. Plot of the FFT of the piano waveforms. Maximum peak occurs at 524.79 Hz.

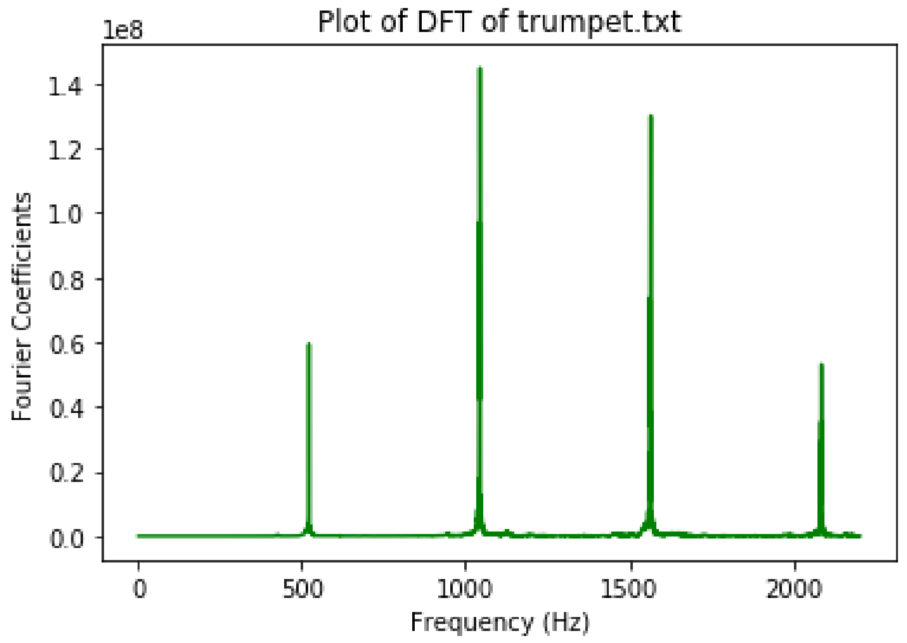

Figure 10. Plot of the FFT of the trumpet waveforms. Maximum peak occurs at 1043.85 Hz.

For the piano data, the maximum peak occurs at 524.79 Hz; for the trumpet data, the maximum peak occurs at 1043.85 Hz. Given that middle C corresponds to a frequency of 261 Hz and moving between one octave corresponds to a doubling of the current frequency, the note being played on the piano is the C one octave above middle C, while the note being played on the trumpet is the C two octaves above middle C.

Question 2. Image Deconvolution

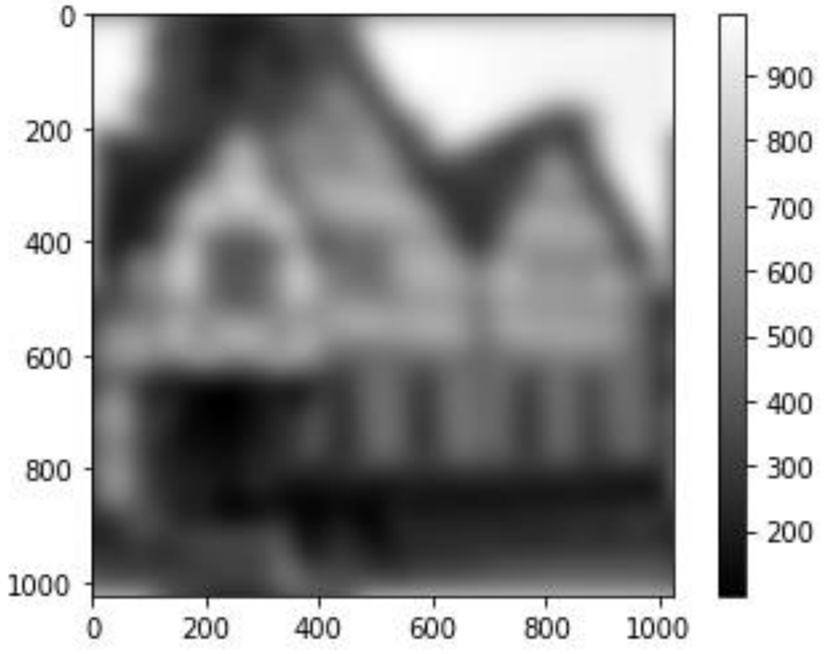

Figure 11. The original image prior to deconvolution.

In Figure 11, we have the plot of an image that has been blurred using a Gaussian point spread function (sigma = 25). Since in this case we know the exact form of the spread function, we can use concepts from Fourier transforms (namely equation 4 from the lab manual) to retrieve the original, unblurred image.

Equation 4 tells us that b_{kl} =KL* a_{kl} * f_{kl}, where b denotes an element of the FT of the blurred image, K and L are the lengths and widths of the image respectively, a is an element of the FT of the original image, and f is the element of the FT of the Gaussian spread function.

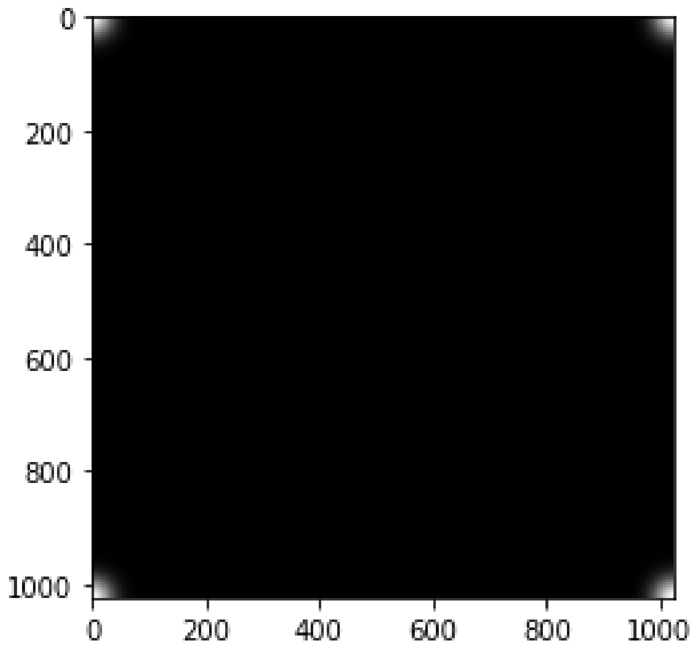

Figure 12. The Gaussian point spread function.

Figure 12 shows the plot of the point spread function that will be used to deconvolve the blurred image; it is a Gaussian point spread function with sigma = 25.

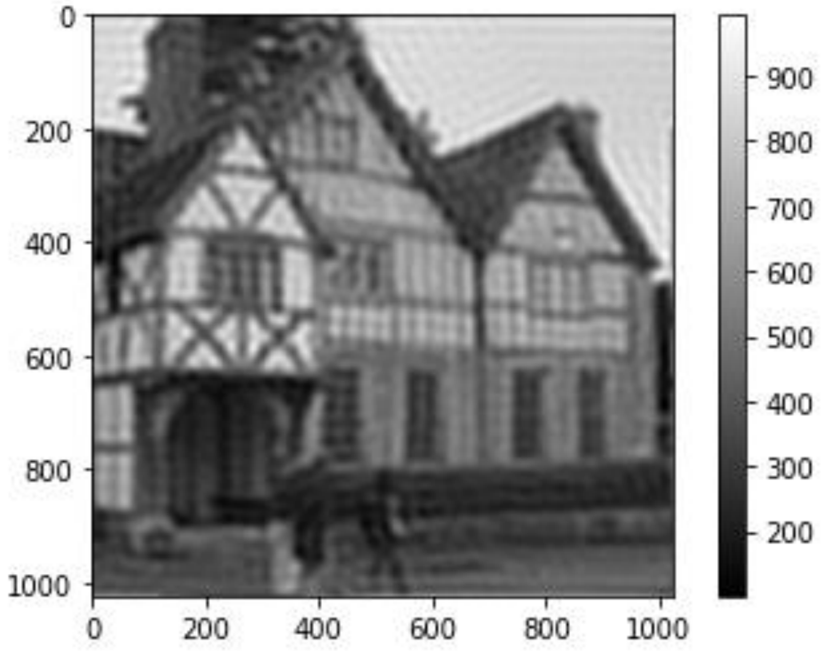

Figure 13. The deconvoluted image.

Finally, in Figure 13 we have the deconvolved image. Deconvolution was carried out by rearranging for the original image (a_{kl}) in equation 4. In our Python implementation of this algorithm, we had to account cases where the value of the Gaussian spread function (f_{kl}) was very small (qualified as being less than e-3), in which case the f_k term is ignored in equation 4.