PHY407 (Lab 5: Computing Fourier Transforms)

Utkarsh Mali[1] and Aslesha Pokhrel[1,2]

[1]Department of Physics, University of Toronto
[2]Department of Computer Science, University of Toronto

October 16, 2020

Contents

Work Allocation:
Aslesha: Question 1, Question 2
Utkarsh: Question 1

Question 1

(a) **Here we are asked to write code that reads the sunspot data and then apply a Fourier transform to it in order to find a non-zero peak in Fourier coefficients $|c_k|^2$.** We executed the Fourier transform as required and then saves the plot as a function of k. We have truncated the first few and last values since we are only highlighting the very sharp peak at a specific sin frequency corresponding to the period of the graph.

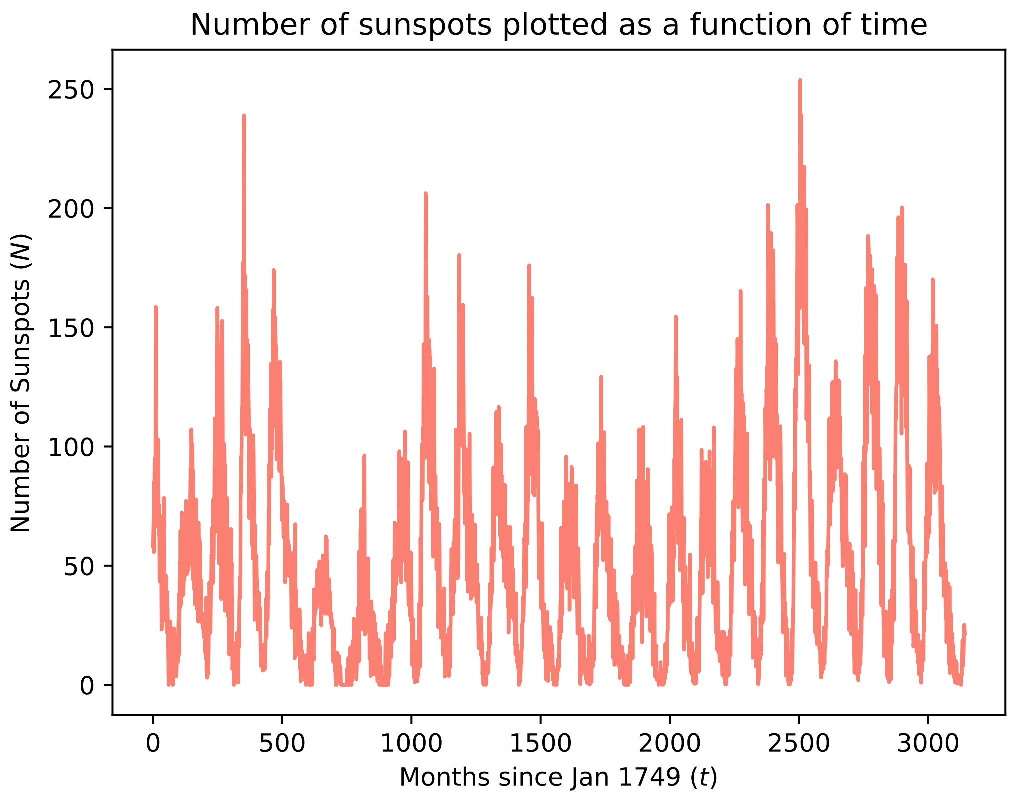

Figure 1: The plot showing sunspot count as a function of time. A period like nature can be observed which should correspond to the sine frequency.

```
CODE OUTPUT:
The maximum value of |ck|^2 corresponds to k: 24
The period corresponding to the length of the cycle we found: 131
```

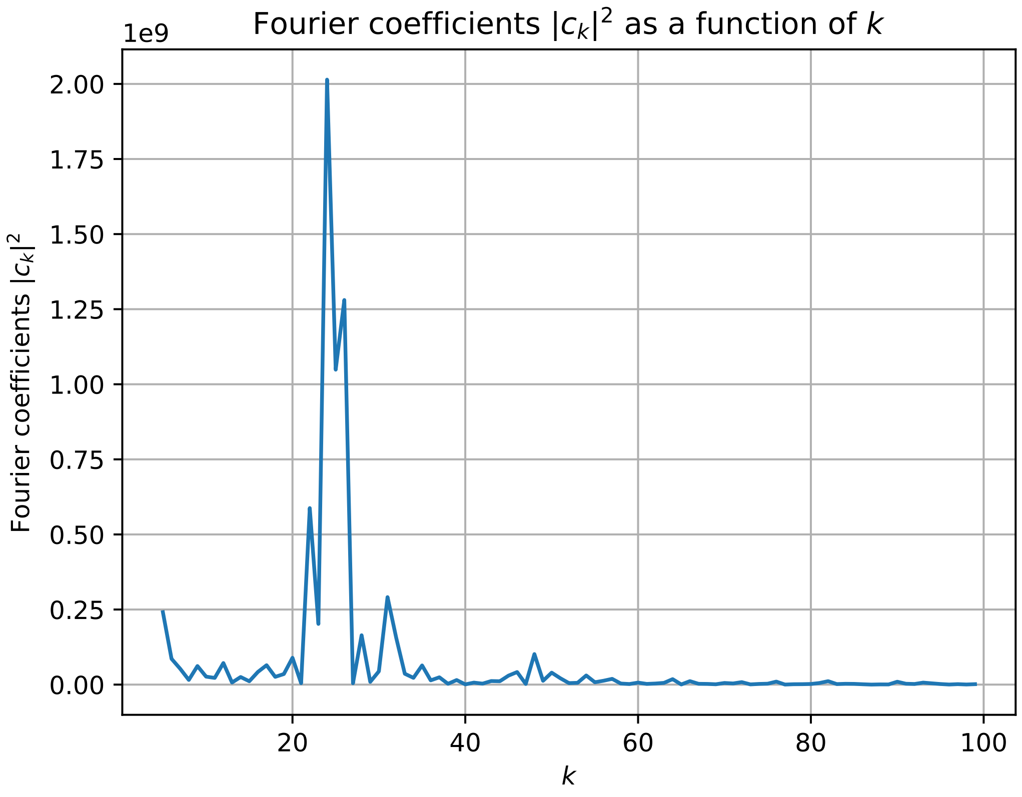

Figure 2: The plot showing the Fourier transform of sunspot count as a function of time. The values of k correspond to specific values of the period.

The approximate value of k found using `np.argmax()` was k=24 which corresponded to a period cycle of 131 months. The period was calculated using $T = N/k$ where N is `len(sunspots)` This period corresponds to the period of the oscillatory wave observed in the sunspots graph. Which was empirically read to be around 125 months, this corresponds to our prediction. Other peaks in the $|c_k|^2$ graph correspond to other oscillatory patterns and noise.

(b) **In this question we are asked to read the data from `dow.txt` and plot the initial results. We are then asked to calculate the coefficients of the Fourier transform, truncating the array to just the few few percent of values. Using this we are then asked to apply an inverse Fourier transform and plot the results.** We loaded the data and plotted the graph using typical numpy features. Following that we used `np.fft.rfft` since it was the fastest method for real valued data, which we know the DOW index is. After truncating the data we applied the inverse Fourier transform and plotted our results. We found **the stronger the clipping of the values the more smoothing was applied on the index**. This was because clipping the coefficients got rid of some terms in the data. This in turn made the graph more smooth.

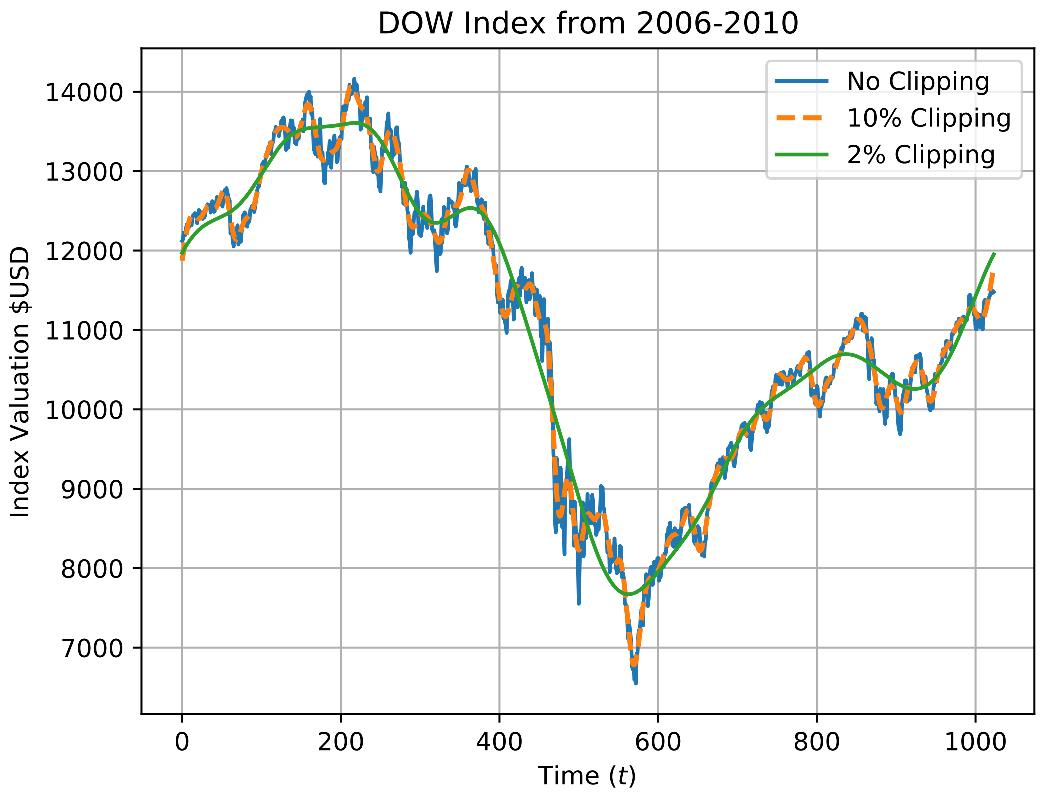

Figure 3: This figure demonstrates the different strengths of smoothing corresponding to different amounts of data clipped from the Fourier transform, in the DOW index from 2006 to 2010. The larger the amount of the data that was clipped the more the smoothing for the graph appears to be.

(c) **This is an extension of part b and asks us to plot the DOW from 2004-2008 but this time we are to compare DFT (Discrete Fourier Transform) vs DCT (Discrete Cosine Transform).** We uploaded the file `dow2.txt` and then plotted it as a function of time. We then wrote a similar program to that in part b in which we apply the Fourier transform. We first calculated the smoothing using the DFT method and plotted the results, we then used the DCT method on the same data set. While the DFT enforces periodicity values at the ends to stay the same, it does fit the data more strongly. The DCT method does not complete the data fitting in a strong manner, but it does not require the values at the endpoint to be the same, therefore it can be applied on continuous growth/decline data sets and is slightly more versatile. The extra artifacts at the edge of the DFT plot are known as the *Gibbs Phenomenon*. The DCT fits less strong since discarding 98% of the DCT information discards alot more than if we discarded the same amount of the DFT information.

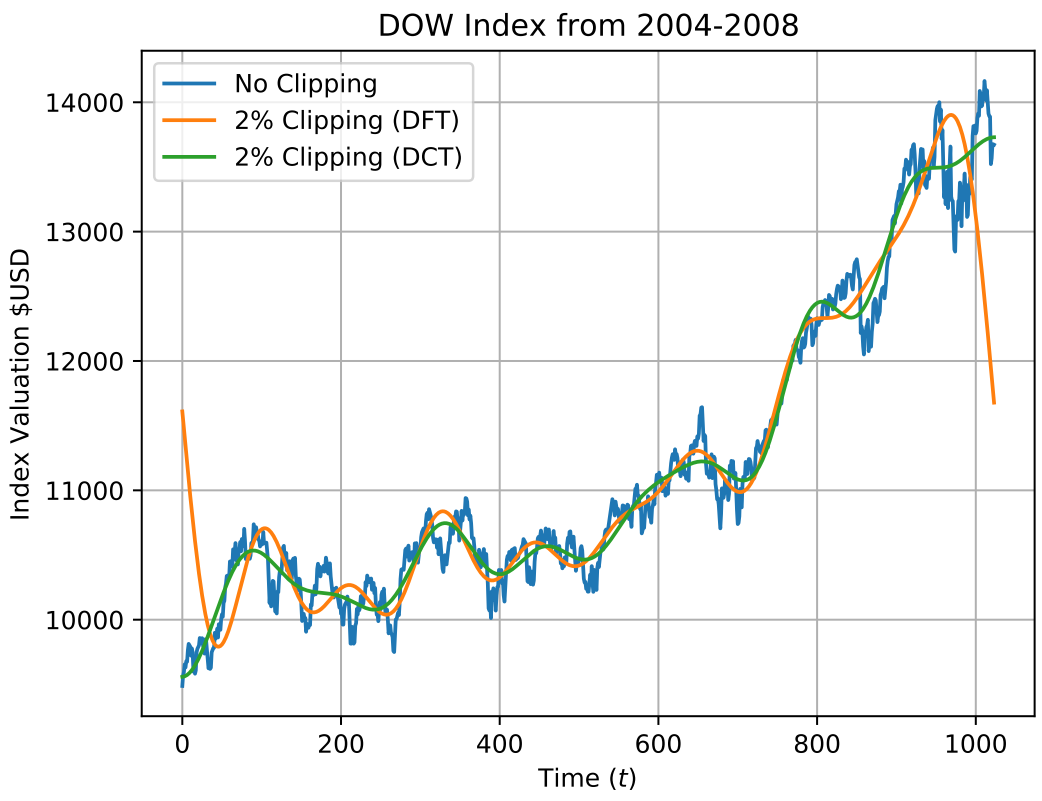

Figure 4: This figure demonstrates the different strengths of smoothing when using the discrete Fourier transform (DFT) and the discrete cosine transform (DCT), one which requires the values at the end of periodicity to stay the same while the other not enforcing the values of periodicity to be the same. We notice that while the cosine transform does not require the edge values to bt the same it does however not fit the data as strongly as the DFT method.

(d) **In this part, we apply the Fourier transform to the given piano and trumpet waveform of a single note and calculate the note for each of the instrument.**

(a) **First, the waveform was loaded from the given files `piano.txt` and `trumpet.txt`.** Each of these waveforms are visualized below:

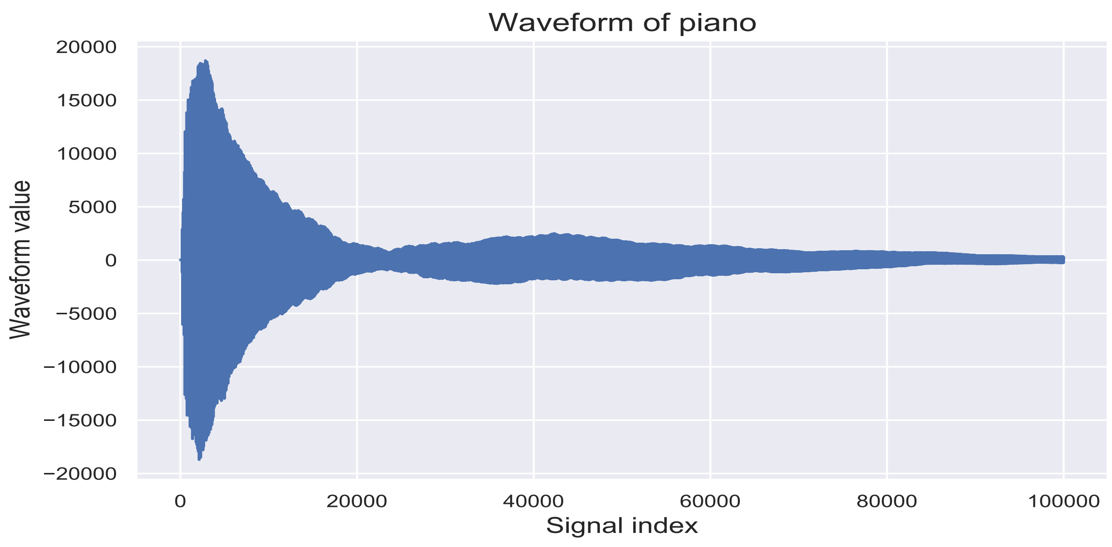

Figure 5: The plot showing the waveform for piano given in `piano.txt`.

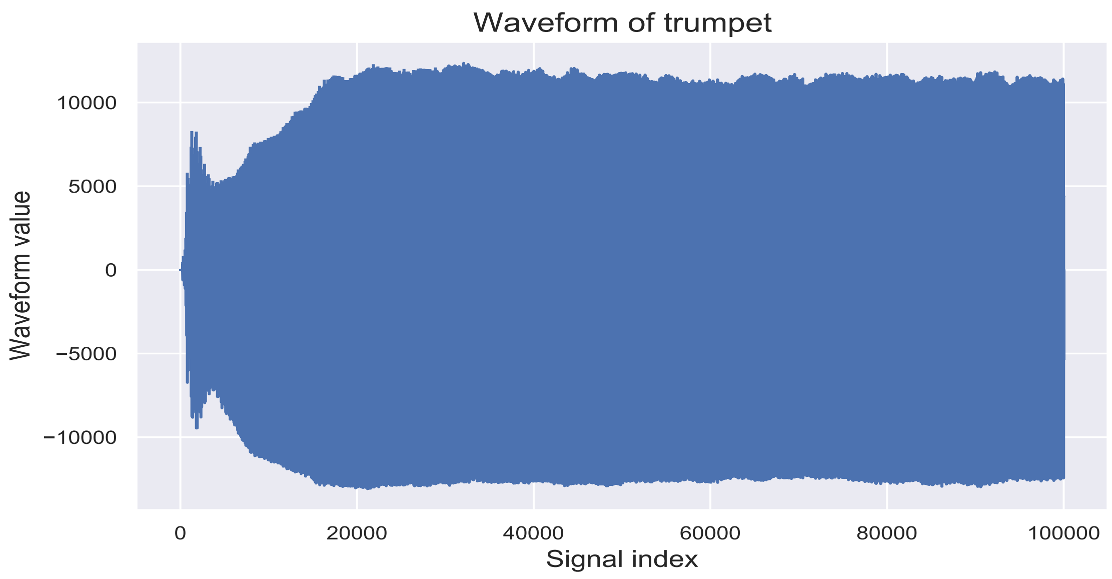

Figure 6: The plot showing the waveform for trumpet given in `trumpet.txt`.

Next, we apply the discrete Fourier transform to the given waveforms and plot the magnitude of the first $10,000$ coefficients.

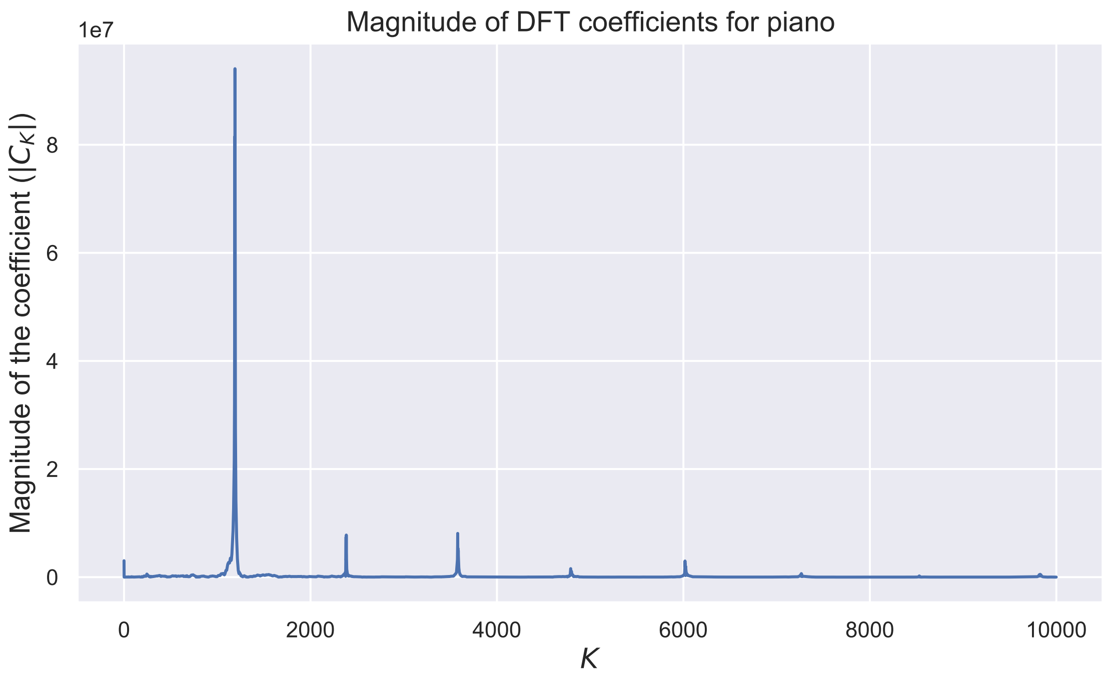

Figure 7: The plot showing the magnitude of the first $10,000$ coefficients after applying the discrete Fourier transform to the piano waveform shown in fig. 5. The horizontal axis in the graph measures k and the vertical axis measures the absolute value of the Fourier coefficient for the corresponding k.

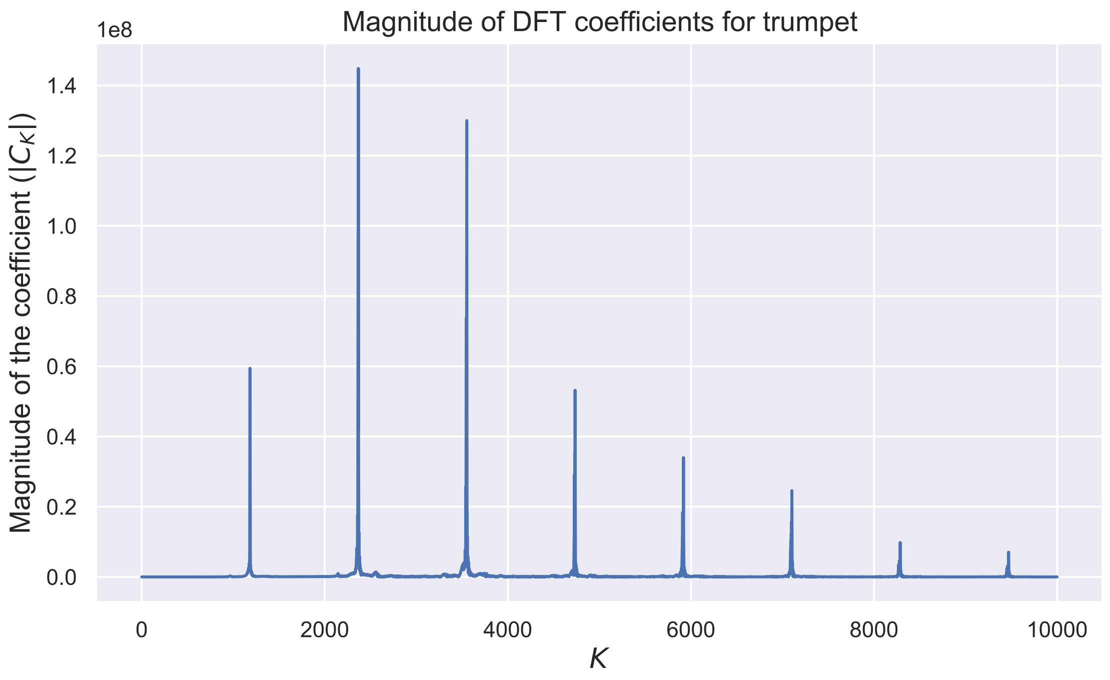

Figure 8: The plot showing the magnitude of the first $10,000$ coefficients after applying the discrete Fourier transform to the trumpet waveform shown in fig. 6. The horizontal axis in the graph measures k and the vertical axis measures the absolute value of the Fourier coefficient for the corresponding k.

Thus, from the plots of the Fourier coefficients fig. 7 and fig. 9, we can conclude that the sound of the piano for one note consists of only one major frequency. However, for the trumpet, the whole spectrum of frequencies needs to be excited to create one note as shown in fig. 8 and fig. 10.

(b) **Then, we use the Fourier transform results to calculate the note being played by each of the instruments.** To find the frequency corresponding to the first peak, we plot the magnitude $|C_K|^2$ versus the frequency of the instruments. The plots are shown below:

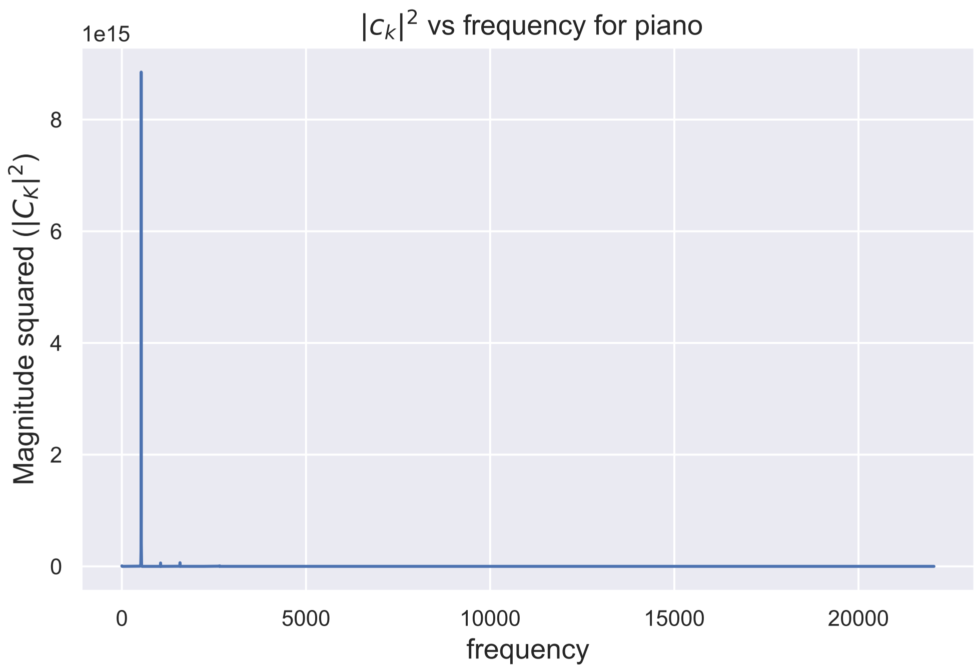

Figure 9: The plot showing the magnitude squared versus the frequency for the Fourier transform of waveform of piano.

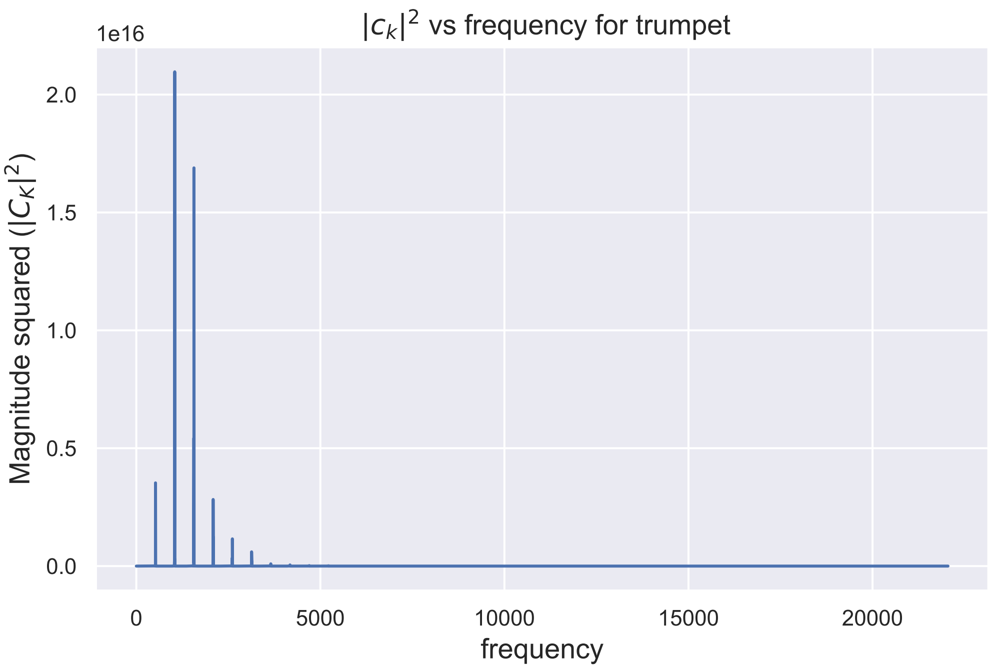

Figure 10: The plot showing the magnitude squared versus the frequency for the Fourier transform of waveform of trumpet.

Using fig. 9 and fig. 10, the frequency corresponding to the first peak was found which was then used to find the note of each of the instruments. The output from the code is given below:

```
Code Output:

The frequency of the first peak magnitude |C_K|^2 for piano is 523.467.
The note number being played by the piano is 3.0

The frequency of the first peak magnitude |C_K|^2 for trumpet is 521.703.
The note number being played by the trumpet is 3.0
```

To find the note number from a given frequency (f), we used the formula $12 * \log_2(f/440)$ and rounded it to the nearest number. This formula was derived from the relationship given in the handout. Hence, the note being played by the piano and the trumpet is 3 notes above A_4 which is C_5.

Question 2: Image Deconvolution

In this section we will use the Fourier transforms to un-blur a out-of-focus blurry image.

(a) **In this part, the image is read from the given file `blur.txt` which was blurred using a Gaussian point spread function of width $\sigma = 25$.** The image is shown below:

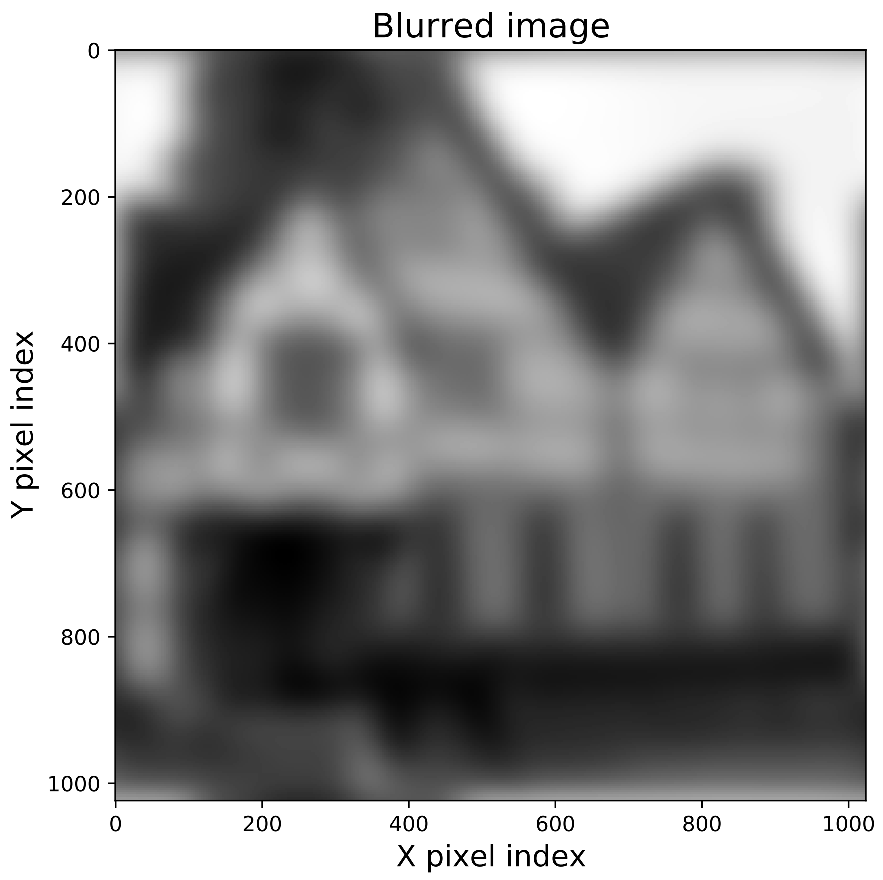

Figure 11: The given black and white photo from blur.txt that was deliberately blurred using a Gaussian point spread function of width $\sigma = 25$.

(b) **Next, we create an array of the same size as image containing the grid of samples drawn from a point spread function (Gaussian with $\sigma = 25$).** The density plot of this point spread function is visualized below:

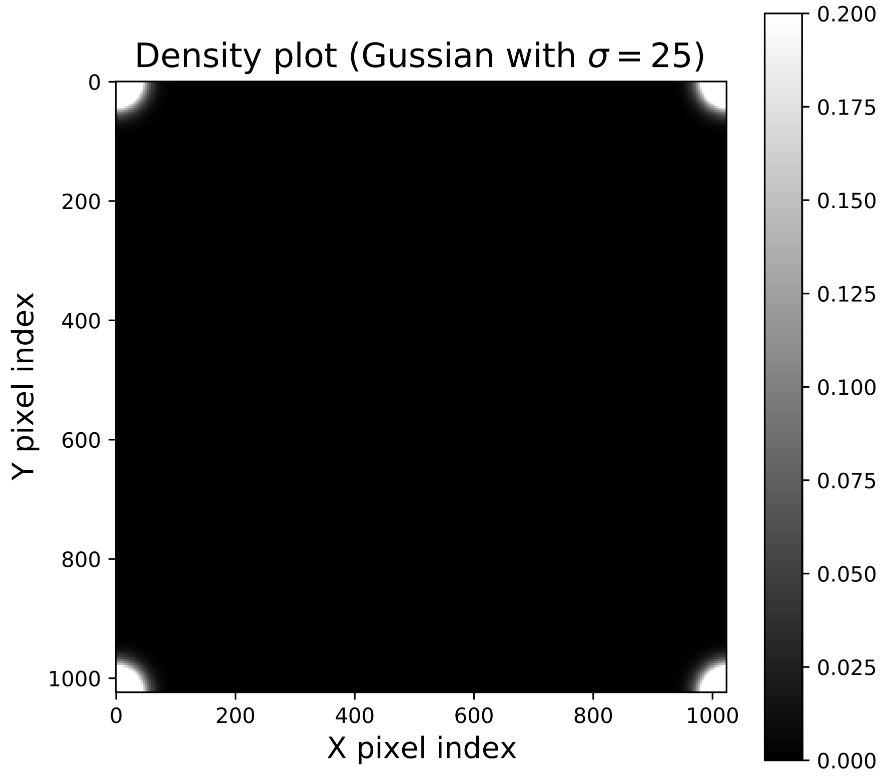

Figure 12: The density plot in this figure visualizes the point spread function given by a Gaussian with $\sigma = 25$. As the Gaussian is centered at the origin and the point spread function is periodic, we can see the bright patches in each of the four corners of the figure. The colorbar displays the colormap and indicates the mapping of data values into the colormap. The `vmax` for this plot was set to 0.2 so the colorbar goes from 0 (black) to 0.2 (white).

(c) Finally, we combine the above two programs and add Fourier transforms using the functions `rfft2` and `irfft2` from `numpy.fft` to make a program that de-blurrs the given image using the steps mentioned in the question. The result of implementing this program and applying it on fig. 11 is visualized below:

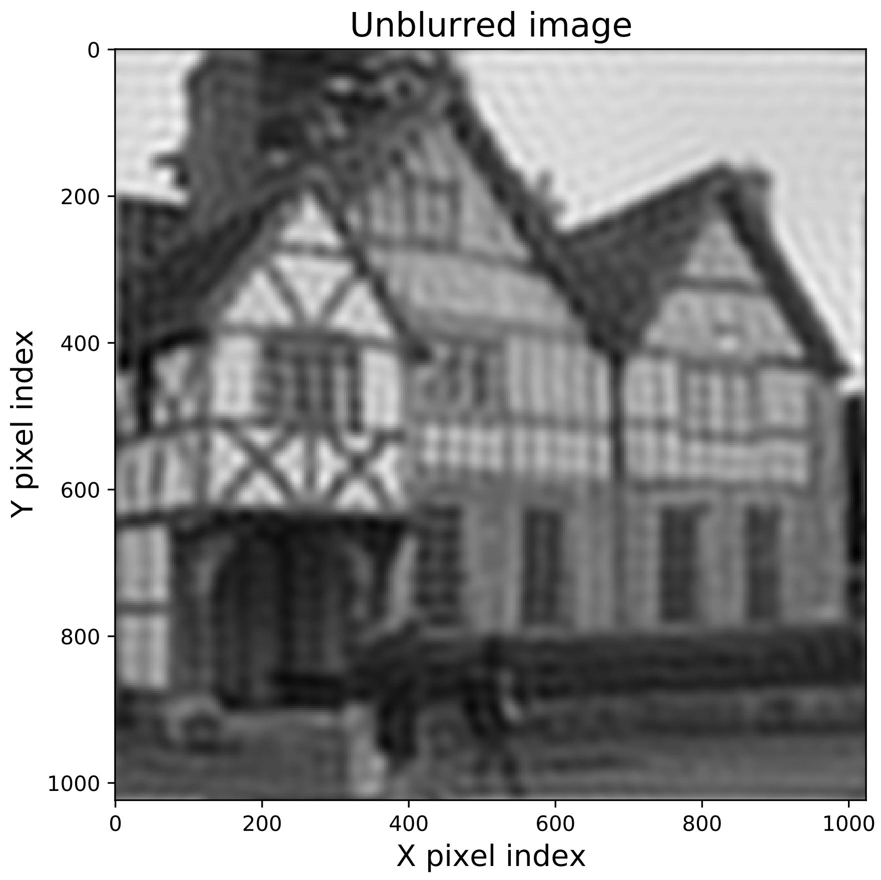

Figure 13: The plot shows the unblurred image of fig. 11 after applying the algorithm mentioned in Exercise 7.9 (c) of the textook.