Lab 7: Discrete Fourier Transform

Max Huggins and Chris Benson

February 9, 2021

Abstract

In this experiment, Python and the Discrete Fourier Transform was used to determine frequencies and recorded waveforms.

1 Objective

The objective of this lab is to understand the Discrete Fourier Transform and use it to understand various signals.

2 Experimental Setup

The written code is in the appendix. The frequency of a tuning fork was determined using audacity to record the signal and the file was read into the code for analysis and determination of the frequency. The frequencies of two tuning forks were also recorded together with audacity and this file was also read into the code for analysis. A signal from a HV capacitor charge-discharge circuit was also analyzed using the fourier transform.

3 Theory

Recalling that the Fourier Transformation converts a physical-space (or time-series) function into a frequency-space is given by the function:

$$F(k) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i x k} dx \tag{1}$$

Thinking of $F(k)$ as the amount of $f(x)$ represented by a frequency k, this representation is how the sum of the frequencies for the square wave and saw-tooth wave from chapter 1 were determined.

If the data is discrete points and not a continuous function the Discrete Fourier Transformation will be performed:

$$F_k = \sum_{n=0}^{N-1} f_n e^{\frac{-2\pi nk}{N}} \tag{2}$$

Where N is the total number of data values f_n is the amplitude at the n-th data point. For multiple frequencies the sum is taken over the N data points for each frequencies. It was then implemented in python.

4 Test Data

First some generic waveforms were analyzed using the program:

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Nov 13 15:49:21 2018

@author: maxhu
"""

# -*- coding: utf-8 -*-
"""
Created on Tue Nov 13 15:03:43 2018

@author: maxhu
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft

def DFT(f_n):
    N = len(f_n)
    f_k = np.zeros((N), dtype=np.complex128)
    j = complex(0,1)
    for k in range(N):
        for n in range(N):
            f_k[k] += f_n[n]*np.exp(-2.0*np.pi*j*n*k/N)

    return f_k

#Set some values for waveform
k_val_1 = 10
k_val_2 = 3
const = 5
N = 1000
t_initial = 0
t_final = 1
t = np.linspace(t_initial, t_final, N, endpoint = False)
data = const + np.cos(2*np.pi*k_val_1*t) + 2*np.sin(2*np.pi*k_val_2*t)
fig = plt.figure()
my_fig = fig.add_subplot(1,1,1)
```

```
plt.plot(t, data, color = 'black')
my_fig.set_xlabel('Amplitude')
my_fig.set_ylabel('time (s)')
my_fig.set_title('Wavefom')
plt.savefig('DFT_1_waveform.png', dpi=300)
plt.show()
plt.close()

fk_real = fft(data).real
fk_imag = fft(data).imag

fig = plt.figure()
my_fig = fig.add_subplot(1,1,1)
index_values = range(len(data))
plt.plot(index_values, fk_real, color='red')
plt.plot(index_values, fk_imag, color='blue')
my_fig.set_xlabel('index value')
my_fig.set_ylabel('counts')
my_fig.set_title('Discrete Fourier Transform')
plt.savefig('DFT_1_real_imag.png', dpi=300)
plt.show()
plt.close()

fk_mag = np.abs(DFT(data))
rate = len(data)/(t_final - t_initial)
k = np.arange(len(data))
T = len(data)/rate
freq_values = k/T

fig = plt.figure()
my_fig = fig.add_subplot(1,1,1)
index_values = range(len(data))
plt.plot(freq_values[0:20], fk_mag[0:20], color='green')
plt.plot(index_values, fk_imag, color='blue')
my_fig.set_xlabel('index value')
my_fig.set_ylabel('counts')
my_fig.set_title('Discrete Fourier Transform')
plt.savefig('DFT_1_mag.png', dpi=300)
plt.show()
plt.close()
```

The graphs outputted are figures 1, 2, and 3. The outcomes matched the expectations.

5 Audacity 1: Tuning Forks

Next, the code was altered slightly to take in WAV files for analysis. Tuning forks of a given frequency were stricken on rubber activators and recorded using audacity. The WAV files were analyzed using this code:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov 13 15:54:29 2018

```

```python
@author: maxhu
"""

# -*- coding: utf-8 -*-
"""
Created on Tue Nov 13 15:49:21 2018

@author: maxhu
"""

# -*- coding: utf-8 -*-
"""
Created on Tue Nov 13 15:03:43 2018

@author: maxhu
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft
from scipy.io import wavfile

rate,data = wavfile.read('384 fork.wav')
print('The data rate is = ', rate)
print('The shape of the data file is =',data.shape)
print('The data type is = ',data.dtype)
data_scaled = []
for element in data:
    data_scaled.append(element/(2**16.0))

plt.xlim(360,400)
data_fft_abs = abs(fft(data_scaled))
d = len(data_fft_abs)/2
plt.plot(data_fft_abs, 'r')
plt.show()
index_value = np.where(data_fft_abs == np.max(data_fft_abs))
print(index_value)
plt.savefig('DFT_1_waveform.png', dpi=300)

#Set some values for waveform
k = np.arange(len(data_fft_abs))
T = len(data_fft_abs)/rate
freqlabel = k/T

plt.plot(freqlabel, data_fft_abs, 'r')
plt.show()
```

A 384Hz tuning fork was used and the code outputted a peak signal at 385Hz. As can be seen from figure 4.

6 Audacity 2: High Voltage Flyback Transformer

Next, the same code was used to analyze a signal from a high voltage, high frequency transformer. This transformer operates around 30kV and has transient oscillations dependent on the amount of current that is drawn. At near zero currents the transformer operates at around 20kHz, as an arc is drawn the frequency can go ultrasonic to upwards of 100kHz. The outputted graphs are in figure 5. As can be seen there is a variety of frequencies that show up ranging from the expected 20kHz-100kHz.

7 Conclusion

This experiment was designed to allow the user to perform Discrete Fourier Transformations on functions and determine the frequencies of interest and display the frequencies onto a graph.

References

[1] L. E. Kinsler, A. R. Frey, A. B Coppens, J. V. Sanders, *Fundamentals of Acoustics*, (Hamilton Press, New York, 2000).

8 Appendix

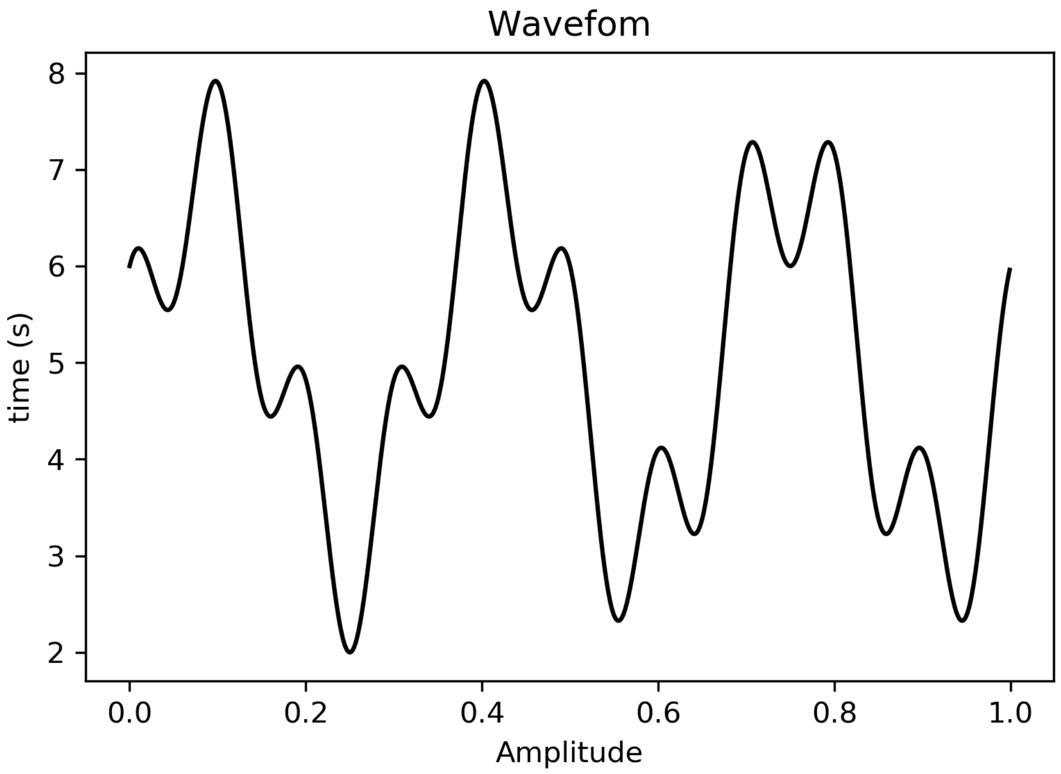


Figure 1: graph of the waveform of the test data.

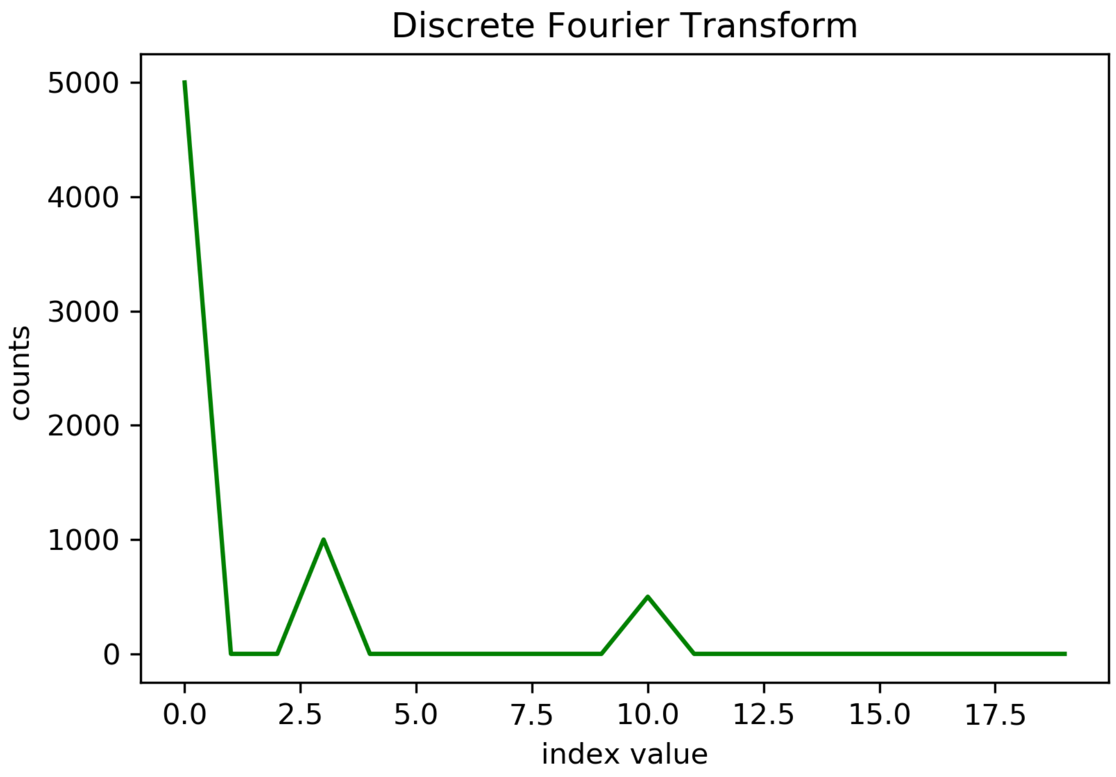


Figure 2: Plotted test data with the magnitude of the Discrete Fourier Transform simulated waveform.

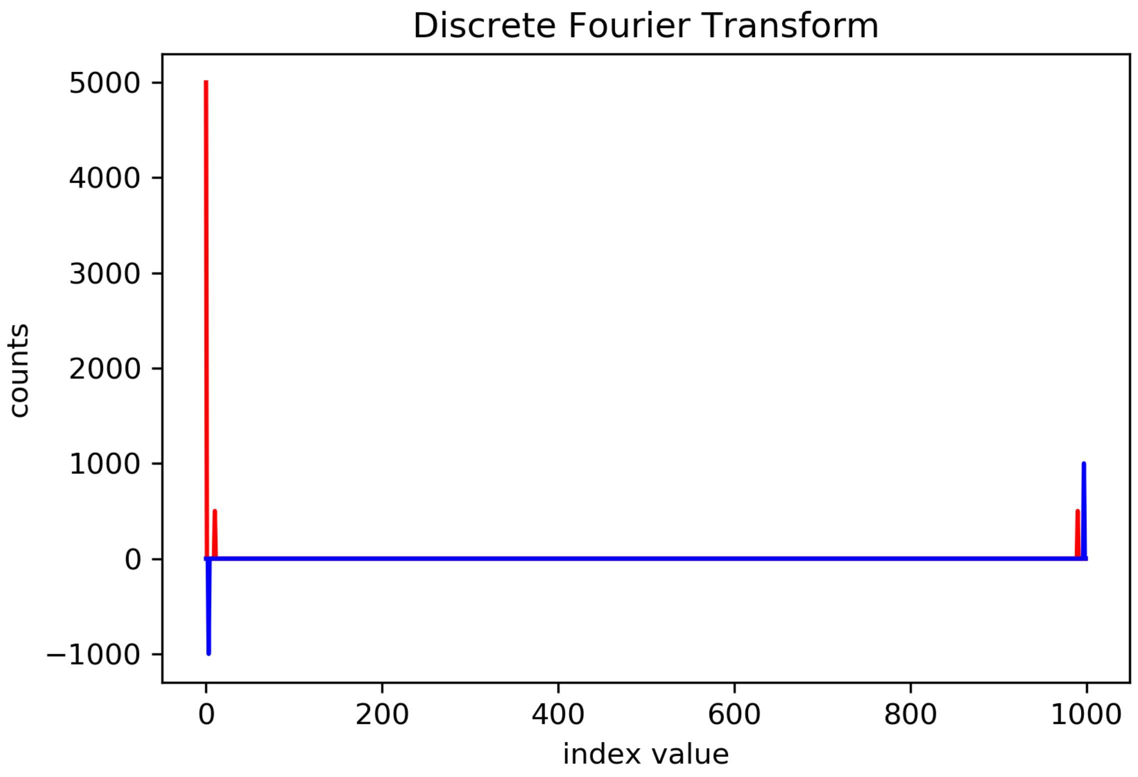
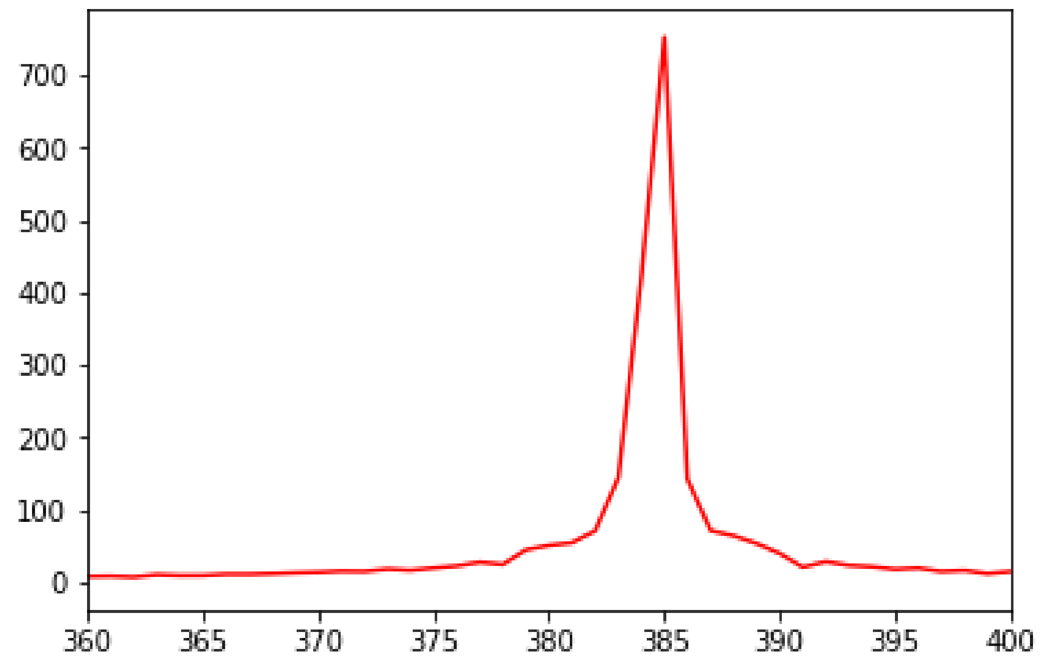


Figure 3: The real (red) and imaginary (blue) parts of our Discrete Fourier Transform of our simulated waveform.



```
(array([  385, 43715], dtype=int64),)
```

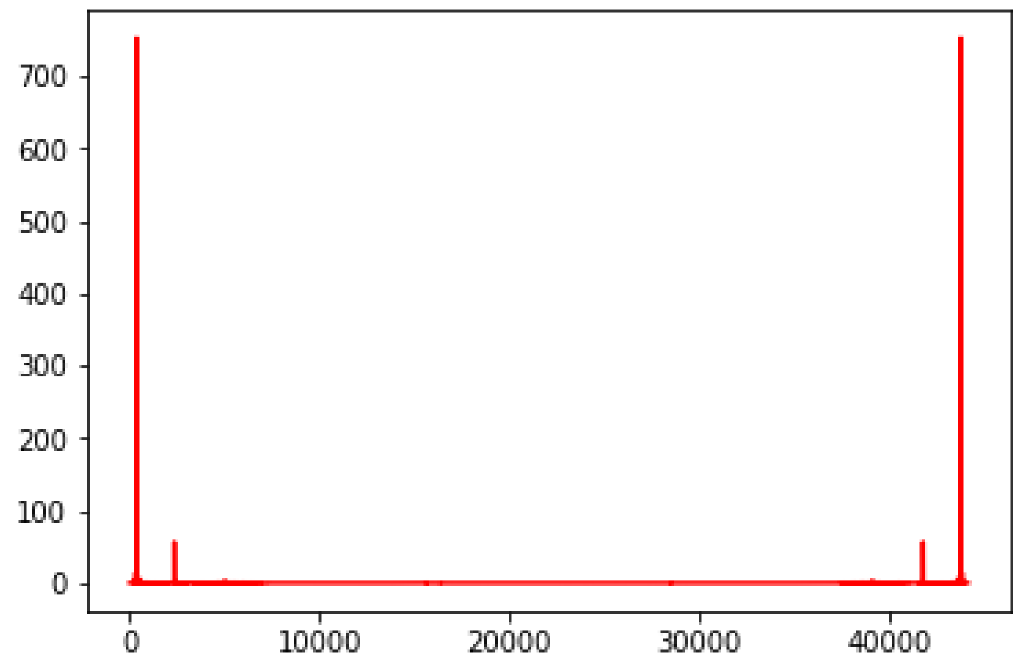
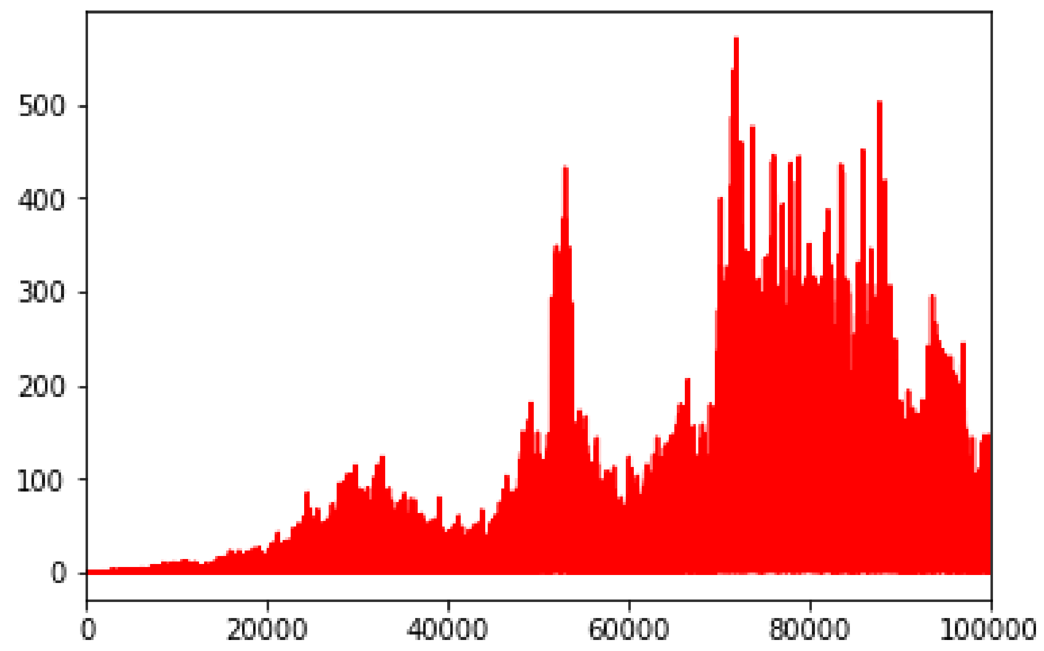


Figure 4: The real (red) and imaginary (blue) parts of our Discrete Fourier Transform of the tuning forks waveform.



```
(array([ 71949, 592187], dtype=int64),)
```

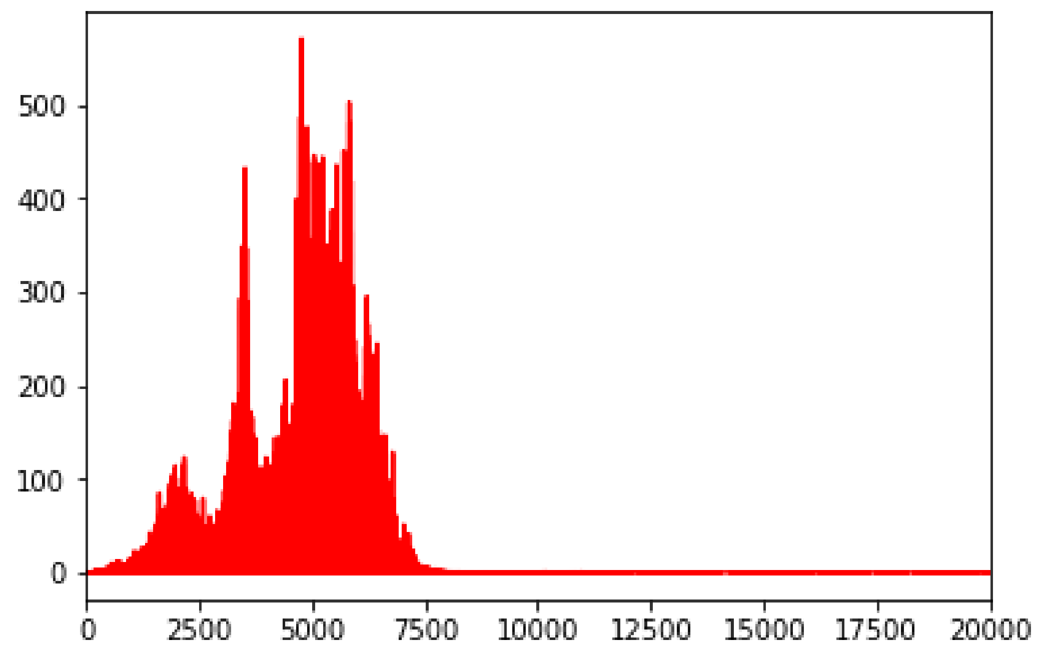


Figure 5: This is the original plot of the flyback transformer wavelengths.