Python code:

#######################

#Willard Wider

#7/11/18

#ELEC4400

#Lab 9

#######################

import matplotlib.pyplot as plt

import numpy as np

from scipy.fftpack import fft

from scipy.io import wavfile

import time

def myOwnDFT(u):

N = len(u)

Un = []

for n in np.arange(0,N):

Uk=0

for k in np.arange(0,N):

Uk+=u[k]\*np.exp((-1\*2j\*np.pi\*k\*n)/N)

Un.append(Uk)

return Un

if \_\_name\_\_ == "\_\_main\_\_":

#specify the stype of the plot to use

plt.style.use('ggplot')

#make a sine wave, duration 1s

x\_index = np.linspace(0,1,500)

x\_index\_frequnecy = np.arange(len(x\_index))

print("making sine wave 10hz...")

y\_values\_10 = np.sin(2\*np.pi\*10\*x\_index)

fourier\_10 = myOwnDFT(y\_values\_10)

print("making sine wave 20hz...")

y\_values\_20 = np.sin(2\*np.pi\*20\*x\_index)

fourier\_20 = myOwnDFT(y\_values\_20)

print("making sine wave 30hz...")

y\_values\_30 = np.sin(2\*np.pi\*30\*x\_index)

fourier\_30 = myOwnDFT(y\_values\_30)

#create the three graphs for fourier frequency plotting

f,xarr = plt.subplots(3)

xarr[0].set\_title('10-30Hz analysis')

xarr[0].plot(x\_index\_frequnecy, fourier\_10)

xarr[1].plot(x\_index\_frequnecy, fourier\_20)

xarr[2].plot(x\_index\_frequnecy, fourier\_30)

plt.show()

#show how long it takes for each sample ammount

samples\_array = [200,400,600,800,1000]

time\_array = []

for i in samples\_array:

time\_x\_index = np.linspace(0,1,i)

#https://docs.python.org/3/tutorial/inputoutput.html

print(f'running transform for {i} samples...')

time\_y\_values = np.sin(2\*np.pi\*10\*time\_x\_index)

#start the timer

start\_time = time.time()#time since epoch

time\_fourier = myOwnDFT(time\_y\_values)

time\_array.append(time.time()-start\_time)

plt.plot(samples\_array,time\_array)

plt.title('time of running each transform')

plt.show()

#estimate how long it will abe to run a transform of the wave file we used

print(f'it took {time\_array[4]} to run a fourier transform of {samples\_array[4]} samples')

fs, data = wavfile.read('kpt.wav')

print(f'the wavefile from previous lab is {len(data)} sampels...') #should be 561,152

ratio = time\_array[4] / samples\_array[4]

print(f'thus it will take {ratio\*len(data)} to complete this one')

#analyse the smaller sample to find the wave used

print('analysing the smaller sample...')

#https://stackoverflow.com/questions/2060628/reading-wav-files-in-python

fs, data = wavfile.read('kpt1note2k.wav')

#fast fourier transform of the data

#fftOut = fft(data)

fftOut = myOwnDFT(data)

#total ammount of time in the recording

length = len(data) / fs

#a list of sample indicies, probably audio frames

n = np.arange(len(data))

#fs the the sampling frequency, the number of samples per second

#we need the duration of time between samples

#time between is 1 (sec) divided by frequency

total\_frequencies = n / length

plt.plot(total\_frequencies,np.abs(fftOut))

#limit the plot

plt.xlim(([0,4186]))

plt.title("Piano Note Analysis")

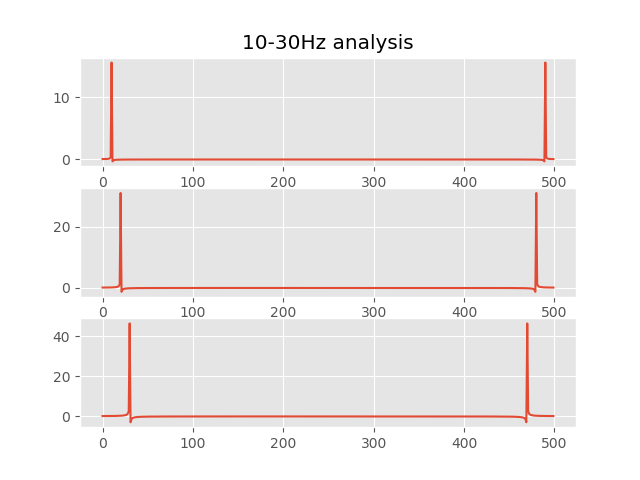
plt.xlabel("frequency (Hz)")

plt.ylabel("amplitude (Db)")

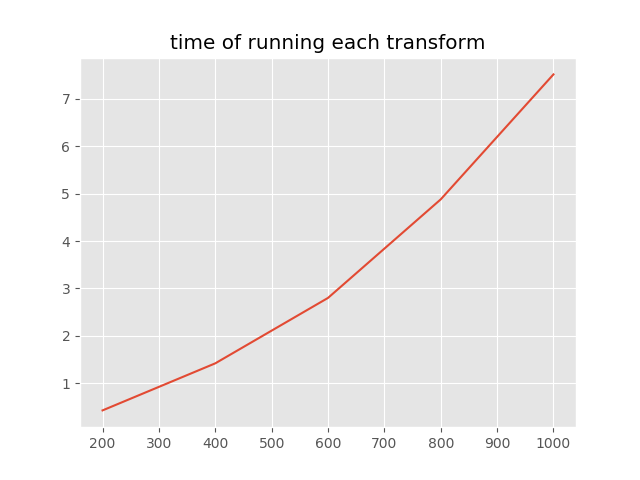
plt.show()

print('results show that it was 462Hz, close to 450Hz, A4')

Picture of graph of 3 frequency’s part 1:



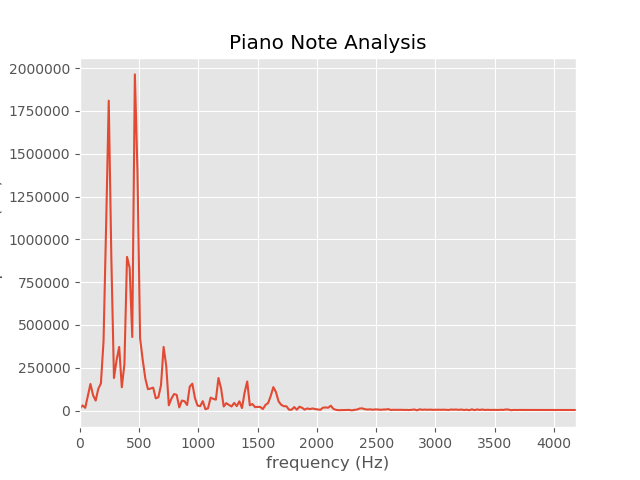
Picture of graph of number of samples to time part 2:



Picture of console output of time estimate part 3:



Picture of graph of analysis of the small wave file part 4:



And in case you want the console output

making sine wave 10hz...

making sine wave 20hz...

making sine wave 30hz...

C:\Users\widerw\AppData\Local\Programs\Python\Python36-32\lib\site-packages\numpy\core\numeric.py:492: ComplexWarning: Casting complex values to real discards the imaginary part

return array(a, dtype, copy=False, order=order)

running transform for 200 samples...

running transform for 400 samples...

running transform for 600 samples...

running transform for 800 samples...

running transform for 1000 samples...

it took 7.511427402496338 to run a fourier transform of 1000 samples

the wavefile from previous lab is 561152 sampels...

thus it will take 4215.052509765625 to complete this one

analysing the smaller sample...

results show that it was 462Hz, close to 450Hz, A4