ISTANBUL TECHNICAL UNIVERSITY

SIGNALS AND SYSTEMS

Homework Report

150160529

Nurettin Kağan Çocalak

Problem 1

a) In this part**,** I created fft function (Figure-1) and reArrange function (Figure-2) which is created for rearrange the array. And in reArrange function I used reverseBit function (Figure-3) for returning of reverse bits.

I tried check with a array (Figure-4). You can see the result of function in Figure-5.

```
def fft(a, n):                                  #fft fuction with two parameters that are a array and its size
    log2OfLength = np.log2(n)
    for innerIterator in range(int(log2OfLength), 0, -1): #dit, seperated like odds and evens but dif seperated first half part and second half part
        m = 2 ** innerIterator
        mid = m / 2
        for k in range(int(mid)):
            ex = cmath.exp(-2j * np.pi * k / m)      #exponent calculated
            if k/m == 0.25:
                ex = -1j
            for z in range(0, n, m):
                c = a[z + k]
                d = a[z + k + int(mid)]
                a[z + k] = (c + d)
                a[z + k + int(mid)] = (c - d) * ex
    reArrange(a, n)                                 #reArrange fuction call after the main loop
    print(a)
    return a
```

Figure-1

```
def reArrange(a, n):                        #this function call for rearrange to array
    for x in range(0, n - 1):
        reverse = reverseBit(x, n)
        if reverse > x:
            temp = a[x]
            a[x] = a[reverse]
            a[reverse] = temp
    return a
```

Figure-2

```
def reverseBit(x, n):                       #this function shall return the reversed bits of x
    l = 0
    log2OfLength = int(np.log2(n))
    while log2OfLength > 0:
        l = int(l << 1)
        l = l + int(x & 1)
        x = int(x >> 1)
        log2OfLength = log2OfLength - 1
    return l


fft(a, n)
```

Figure-3

```
a = [1, 1, -1, -1, -1, 1, 1, -1]            #this array is written for check the fft function
n = len(a)                                  #array size
```

Figure-4

```
[0, (2+2j), -4j, (2-2j), 0j, (2+2j), 4j, (2-2j)]
```

Figure-5

b) In this part, I used fft function and lpf function (low pass filter). You can see my lpf function in Figure-6 and in first part fft function told. In Figure-7 all functions is seen.

```
def lpf(samples,fc,delta,gain):                          #low pass filter function definition with samples, cut off, delta and gain inputs
    alpha = delta/(delta + 1 /(2*np.pi*fc))*gain         #calculation of alpha with gain for filters
    resultFirstOrder = samples.copy()                    #resultFirstOrder arrayz
    resultSecondOrder = samples.copy()                   #resultSecondOrder array

    resultFirstOrder[0] = alpha * samples[0]             #samples multiply with alpha for the first order filter
    resultSecondOrder[0] = alpha * resultFirstOrder[0]   #created first order filter multiply with alpha for the creation of second order filter

    for i in range(len(samples) - 1):

        resultFirstOrder[i+1] = resultFirstOrder[i]*(1-alpha) + alpha*samples[i+1]            #calculation for first order low pass filter for len(samples) inputs
        resultSecondOrder[i+1] = resultSecondOrder[i]*(1-alpha) + alpha*resultFirstOrder[i+1] #calculation for second order low pass filter thanks to first order

    return resultSecondOrder
```

Figure-6

```
def fft(a, n):                                           #fft fuction with two parameters that are a array and its size
    log2OfLength = np.log2(n)
    for innerIterator in range(int(log2OfLength), 0, -1):          #dit fuction seperated like odds and evens but dif seperated first half part and second half part
        m = 2 ** innerIterator
        mid = m / 2
        for k in range(int(mid)):                                   #every time looking in half
            ex = cmath.exp(-2j * np.pi * k / m)                     #exponent calculated
            if k/m == 0.25:
                ex = -1j
            for z in range(0, n, m):
                c = a[z + k]
                d = a[z + k + int(mid)]
                a[z + k] = (c + d)
                a[z + k + int(mid)] = (c - d) * ex
    reArrange(a, n)                                              #reArrange fuction call after the main loop
    print(a)
    return a


def reArrange(a, n):                                             #this function call for rearrange to array
    for h in range(0, n - 1):
        reverse = reverseBit(h, n)
        if reverse > h:
            temp = a[h]
            a[h] = a[reverse]
            a[reverse] = temp
    return a


def reverseBit(x, n):                                            #this function shall return the reversed bits of x
    l = 0
    log2OfLength = int(np.log2(n))
    while log2OfLength > 0:
        l = int(l << 1)
        l = l + int(x & 1)
        x = int(x >> 1)
        log2OfLength = log2OfLength - 1
    return l
```

Figure-7

```
def getSamples(i,points,delta,samples):                    #takes 256 point from i th second
    result = []
    start = int(1/delta * i)
    for j in range(points):
        result.append(samples[j + start])

    return result
```

Figure-8

In Figure-8, getSamples function is seen. This function takes 256 point from i th second.

```
def fftplt(a,sampling_rate,title,figure,ax):                    #this function ploting for fft function
    n = len(a)
    result = fft(a, n)

    xF = np.linspace(0.0, 1/sampling_rate/2.0, n//2)
    ax[figure].plot(xF, np.abs((result[:n//2])))
    ax[figure].set_title(title)
    plt.grid()
    plt.xlabel("Freq")
    plt.ylabel("Magn")
```

Figure-9

In Figure-9, this function created for ploting the fft function.

Finally functions definitions are over. So we will read wav file and and this input will process with lpf and fft functions.

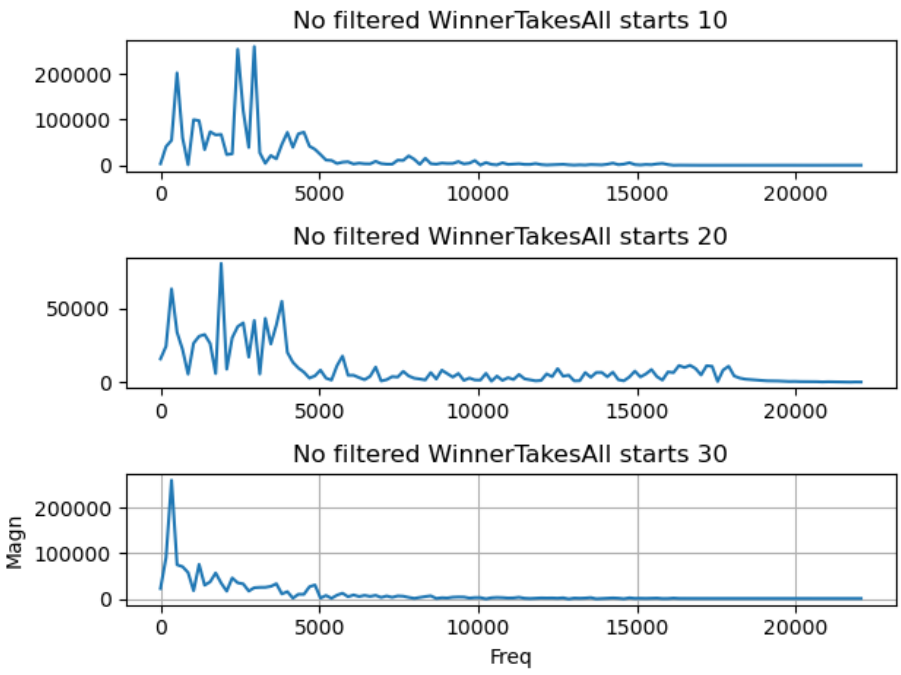
```
wavFileName= 'WinnerTakesAll.wav'

#read wav file
obj = wave.open(wavFileName,'rb')                    #WinnerTakesAll.wav is opened and readed
amplitudeWidth = obj.getsampwidth()                  #sample width to in bytes
frameRate = obj.getframerate()                       #sampling frequency
nTimesFrames = obj.getnframes()                      #number of audio frames
readFrames = obj.readframes(nTimesFrames)            #reads and returns at most n frames of audio, as a bytes object
samples = np.fromstring(readFrames, np.int16)        #create array with frames
obj.close()                                          #close the stream if it was opened by wave module
```

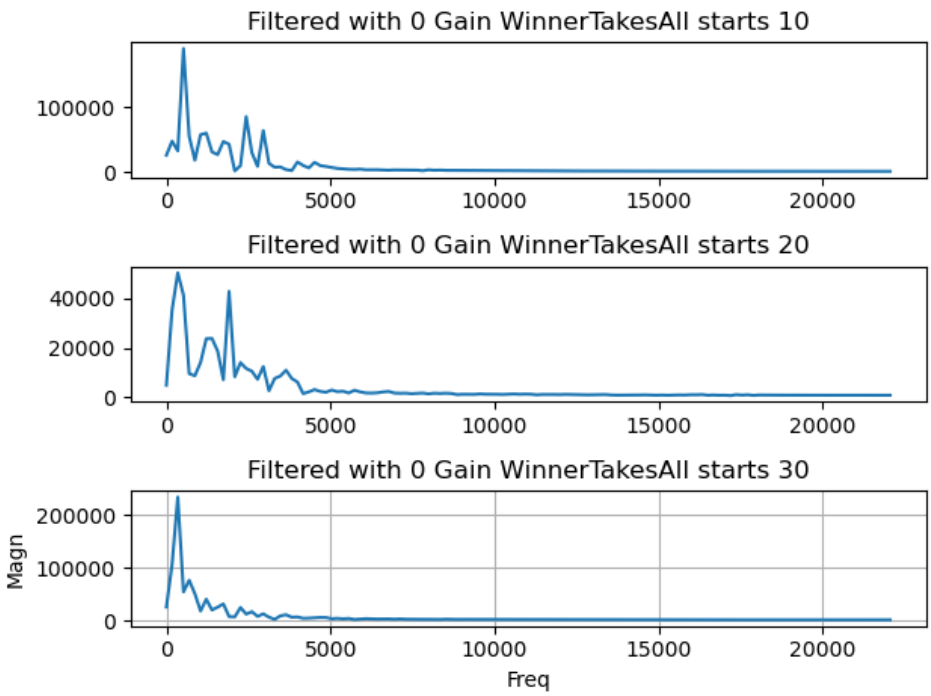
```
fc = 2000.0
delta = 1.0/44100.0

# In here fft function plotted without filtered wit low pass filter
figure = 0
fig, ax = plt.subplots(3, 1)                          #It is written for plot visualization
for i in range(10, 40, 10):                           #10, 20 and 30th seconds
    pointWith256 = getSamples(i,256,delta,samples)    #get 256 samples
    fftplt(pointWith256, delta, "No filtered WinnerTakesAll starts "+ str(i),figure,ax)
    figure += 1
plt.show()

# In here fft function plotted filtered with 0 gain low pass filter
figure = 0
fig, ax = plt.subplots(3, 1)  # It is written for plot visualization
for i in range(10, 40, 10):  # 10, 20 and 30th seconds
    gain = 1
    lpFilteredNoGain = lpf(samples, fc, delta, gain).astype(samples.dtype)  # lpf function is called for WinnerTakesAll.wav file with no gain
    pointWith256 = getSamples(i,256,delta,lpFilteredNoGain)                 #get 256 samples
    fftplt(pointWith256, delta, "Filtered with 0 Gain WinnerTakesAll starts "+ str(i),figure,ax)
    figure += 1
plt.show()

# In here fft function plotted filtered with 5 gain low pass filter
figure = 0
fig, ax = plt.subplots(3, 1)  # It is written for plot visualization
for i in range(10, 40, 10):  # 10, 20 and 30th seconds
    gain = pow(10, 0.25)
    lpFiltered = lpf(samples, fc, delta, gain).astype(samples.dtype)  # lpf function is called for WinnerTakesAll.wav file with 5dB gain
    pointWith256 = getSamples(i,256,delta,lpFiltered)                  #get 256 samples
    fftplt(pointWith256, delta, "Filtered with 5 Gain WinnerTakesAll starts "+ str(i),figure,ax)
    figure += 1
plt.show()
```

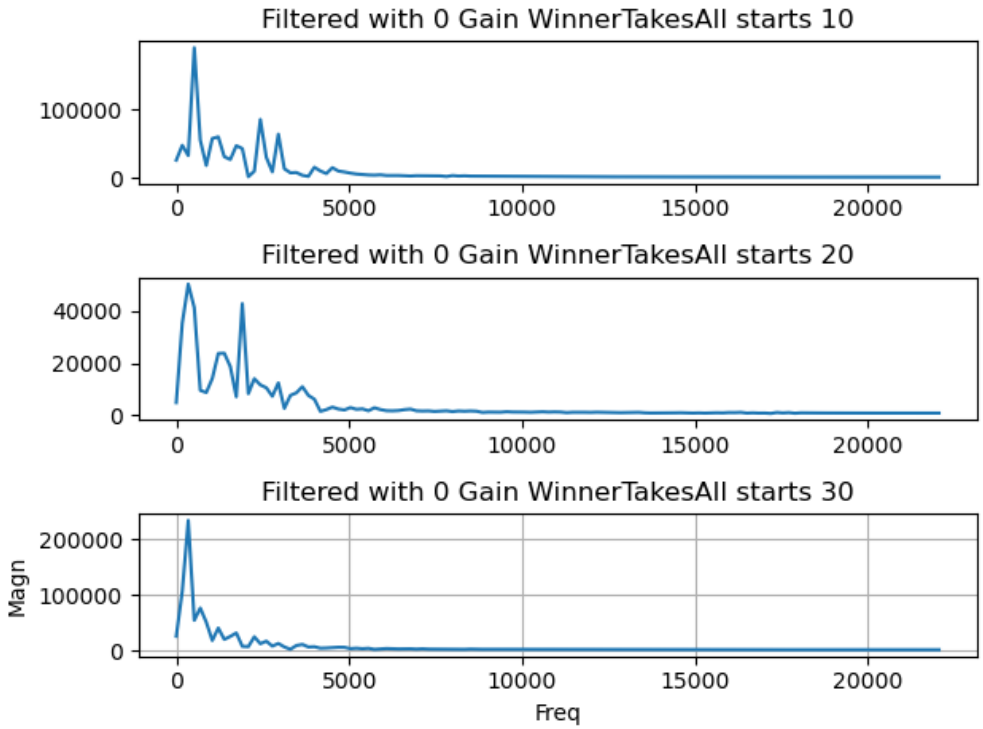
Figure-9

In Figure-9, plots drawed without low pass filtered and with 0 and 5 gain low pass filtered.

You can see the results of the plots below.




Filtered with 0 Gain WinnerTakesAll starts 10
100000
0
0
5000
10000
15000
20000
Filtered with 0 Gain WinnerTakesAll starts 20
40000
20000
0
0
5000
10000
15000
20000
Filtered with 0 Gain WinnerTakesAll starts 30
Magn
200000
100000
0
0
5000
10000
15000
20000
Freq


Filtered with 0 Gain WinnerTakesAll starts 10
100000
0
0
5000
10000
15000
20000
Filtered with 0 Gain WinnerTakesAll starts 20
40000
20000
0
0
5000
10000
15000
20000
Filtered with 0 Gain WinnerTakesAll starts 30
Magn
200000
100000
0
0
5000
10000
15000
20000
Freq