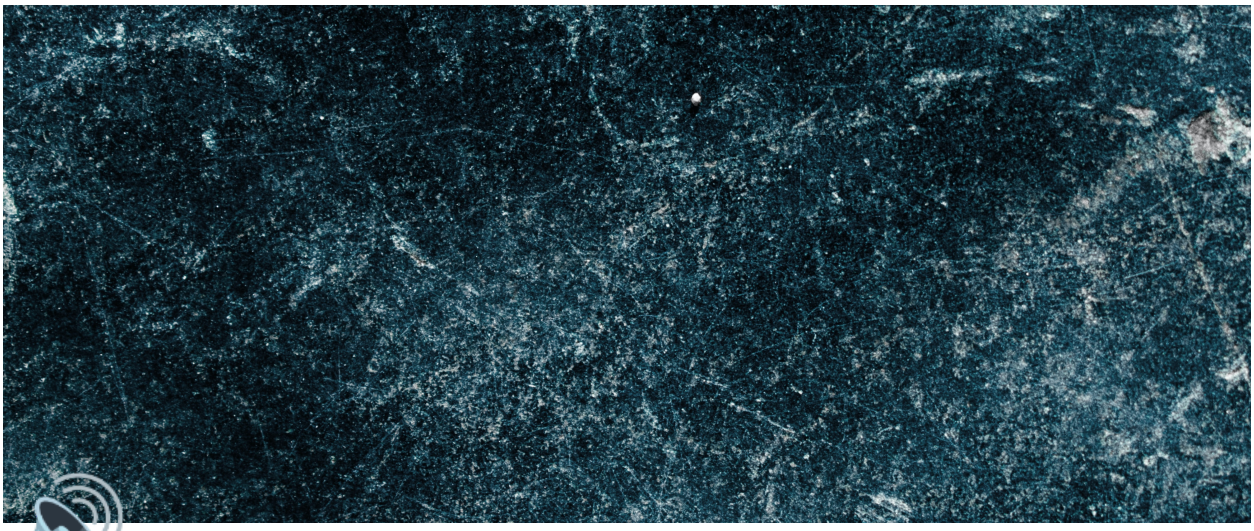


Noise Cancelation Report

Made By

Name : Mazen Mohamed Sayed

ID : 52-2735

T-26

💡 Continuing on Milestone 1

Noise Generation

```
def NoiseGeneration():
```

```
    t = np.linspace(0, 3, 3*1024)

    # Pick Two Random Frequencies
    f1 = np.random.randint(0,512,1)
    f2 = np.random.randint(0,512,1)

    # Generate Noise
    sin_1 = np.sin(2*f1*np.pi*t)
    sin_2 = np.sin(2*f2*np.pi*t)
    noise = sin_1 + sin_2

    '''
    # Display Noise

    plt.figure()

    plt.plot(t, noise)

    plt.grid('True')

    '''

    # Return Noise
    return noise
```

Description

1] Generate two random frequency from 0 to 512

2] substituting the two frequencies in the equation

$$n(t) = sin(2f_{n1}\pi t) + sin(2f_{n2}\pi t)$$

in order to generate the noise

Noise Cancellation

```
# Remove Noise from Signal
def NoiseCancellation(x_before,x_noised):

    # initialize two frequencies with zero
    f1s = 0
    f2s = 0

    # Transfer from Time Domain -> Frequency Domain
```

```
t = np.linspace(0, 3, 3*1024)
N = 3*1024
ff = np.linspace(0,512,int(N/2))
x_f = fft(x_noised)
x_f = 2/N * np.abs(x_f[0:np.int(N/2)])

x_bef = fft(x_before)
x_bef = 2/N * np.abs(x_bef[0:np.int(N/2)])

# Get Maximum Amplitude
max1 = np.max(x_bef)
print("Maximum1" ,max1)

freqArr = [] # create empty list for frequency
AmpArr  = [] # create empty list for amplitudes


i = 0
for z in x_f :
    if z > max1 :
        freqArr.append(i // 3)
        AmpArr.append(z)
    i+=1

print(freqArr)
# Generate F1
maximum = 0
f1s = 0
i = 0
for z in AmpArr :
    if z > maximum :
        f1s = freqArr[i]
        maximum = z
    i += 1

freqArr.remove(f1s)
AmpArr.remove(maximum)
sin_1 = np.sin(2*f1s*np.pi*t)
# Generate F2
maximum = 0
f2s = 0
i = 0
for z in AmpArr :
    if z > maximum :
        f2s = freqArr[i]
        maximum = z
    i += 1
sin_2 = np.sin(2*f2s*np.pi*t)


# Generate Filtered Signal
x_filtered = x_noised - (sin_1 + sin_2)

print("f1 ",f1s)
```

```
    print("f2 ",f2s)

    # Return Filtered Frequency
    return x_filtered
```

Description

1] Take as an input signal before adding noise to it x_{before} and signal after adding noise to it x_{after}

2] Set f_1 and f_2 to 0

3] Get Fourier Transform of $x_{before} \rightarrow x_{bef}$ and $x_{after} \rightarrow x_{ff}$

4] Put all the Amplitude and Frequencies that is greater than Maximum of x_{bef} in $freqArr$ and

$AmpArr$ respectively

5] Get Maximum Frequency and Amplitude in $freqArr$ and $AmpArr$ and remove them from the lists and set f_{1s} to Maximum Frequency (I did that as sometimes the noise had some width for example : 5 and 5.33333 which both are greater than Maximum of signal before adding noise)

$$f_{1s} = sin(2f_{n1}\pi t)$$

6] Similarly repeat process again to get f_{2s}

$$f_{2s} = sin(2f_{n2}\pi t)$$

7] Get filtered signal by subtracting f_{1s} and f_{2s} from the x_{noised}

Display

```
def Display(x_before,noise,x_filtered,x_noised):
```

```
    # Transfer From Time Domain -> Frequencyy Domain
    N = 3*1024
    f = np.linspace(0,512,int(N/2))


    x_before_f = fft(x_before)
    x_before_f = 2/N * np.abs(x_before_f[0:np.int(N/2)])

    noise_f  = fft(noise)
    noise_f  = 2/N * np.abs(noise_f [0:np.int(N/2)])

    x_filtered_f  = fft(x_filtered)
    x_filtered_f = 2/N * np.abs( x_filtered_f [0:np.int(N/2)])

    x_noised_f  = fft(x_noised)
    x_noised_f = 2/N * np.abs(  x_noised_f [0:np.int(N/2)])

    t = np.linspace(0, 3, 3*1024)

    # Display

    plt.subplot(4,2,1); plt.plot(t, x_before);
    plt.subplot(4,2,2); plt.plot(f, x_before_f);
    plt.subplot(4,2,3); plt.plot(t, noise);
    plt.subplot(4,2,4); plt.plot(f, noise_f);
    plt.subplot(4,2,5); plt.plot(t, x_noised);
    plt.subplot(4,2,6); plt.plot(f, x_noised_f);
    plt.subplot(4,2,7); plt.plot(t, x_filtered);
    plt.subplot(4,2,8); plt.plot(f, x_filtered_f);


''' Driver Code  '''

x_before = Play()

noise = NoiseGeneration()

x_noised = x_before + noise

x_filtered = NoiseCancellation(x_before,x_noised)

Display(x_before,noise,x_filtered,x_noised)

#sd.play(x_before,3*1024)

#sd.play(x_noised,3*1024)

sd.play(x_filtered,3*1024)
```

Input

n = 3072

l = 3

Left	Right	Ti	Tp
C3	C4	0	0.5
C3	D4	0.75	0.15
C3	C4	1	0.15
A3	done	1.25	0.15
E3	A4	1.5	0.25
E3	A4	2	0.15
C3	F4	2.30	0.5
E3	A4	2.9	0.1

Please Enter Number of Samples : 3072

Please Enter a Range : 3

Please Enter a Left Note : C3

Please Enter a Right Note : C4

Please Enter a Initial Time : 0

Please Enter a Period : 0.5

Please Enter a Left Note : C3

Please Enter a Right Note : D4

Please Enter a Initial Time : 0.75

Please Enter a Period : 0.15

Please Enter a Left Note : C3

Please Enter a Right Note : C4

Please Enter a Initial Time : 1

Please Enter a Period : 0.15

Please Enter a Left Note : A3

Please Enter a Right Note : done

Please Enter a Initial Time : 1.25

Please Enter a Period : 0.15

Please Enter a Left Note : E3

Please Enter a Right Note : A4

Please Enter a Initial Time : 1.5

Please Enter a Period : 0.25

Please Enter a Left Note : E3

Please Enter a Right Note : A4

Please Enter a Initial Time : 2

Please Enter a Period : 0.15

Please Enter a Left Note : C3

Please Enter a Right Note : F4

Please Enter a Initial Time : 2.3

Please Enter a Period : 0.5

Please Enter a Left Note : E3

Please Enter a Right Note : A4

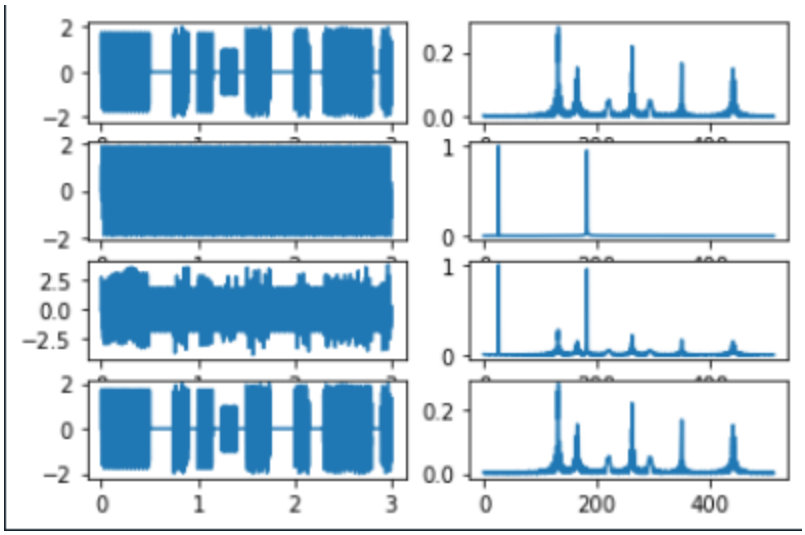
Please Enter a Initial Time : 2.9

Please Enter a Period : 0.1

Please Enter a Left Note : done

Please Enter a Right Note : done

Output


2
0
−2
0.2
0.0
2
0
−2
1
0
2.5
0.0
−2.5
1
0
2
0
−2
0.2
0.0
0
1
2
3
0
200
400