ASSIGNMENT 1

EC867 : FOURIER AND WAVELET ANALYSIS

COURSE INSTRUCTOR : DR. DEEPU VIJAYASENAN

SUBMITTED BY : AKASH A
ROLL NO : 191EC102

Q1: Record/synthesize the following signals. Plot in the time domain and frequency domain.

a] $x1[n] = \sin[\frac{\pi n}{8}](u[n] - u[n-32]) + \sin[\frac{2\pi n}{3}](u[n-32] - u[n - 64])$

b] x2[n] = Gaussian random noise with zero mean and unit variance (100 samples)

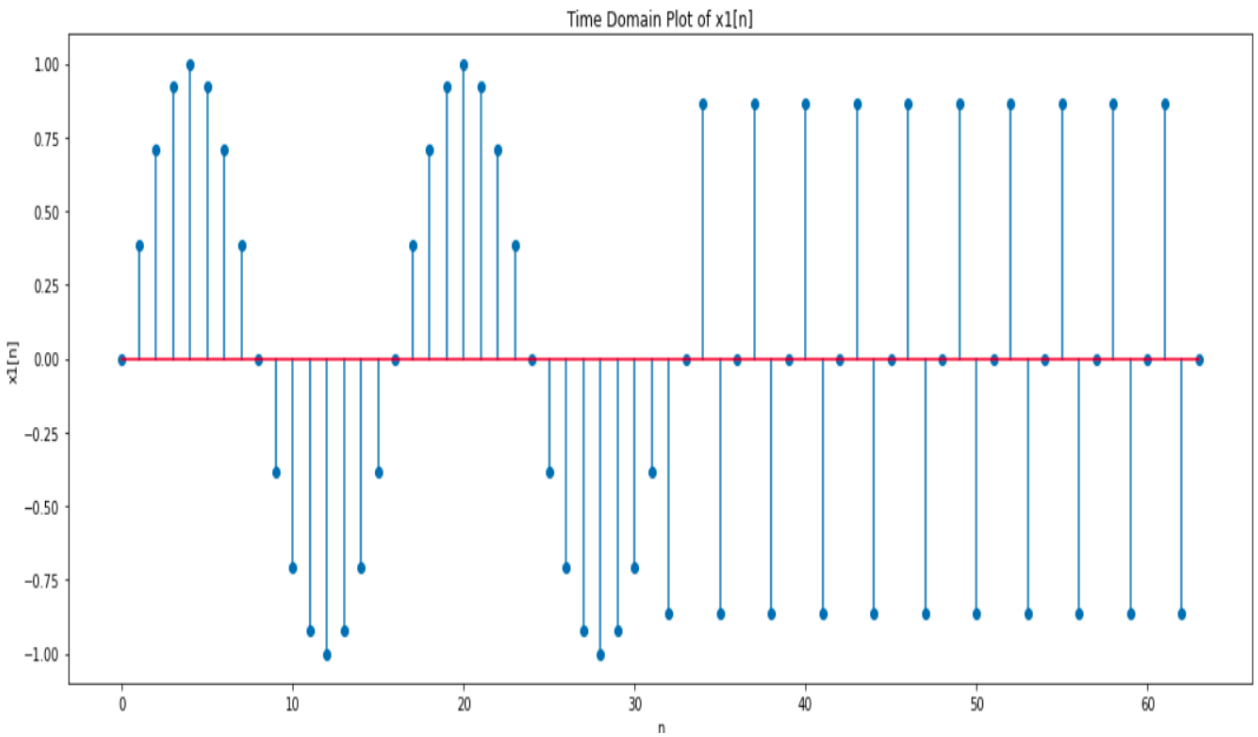
c] x3[n] = Sudden loud clap with 8kHz sampling rate

Ans. **Code:**

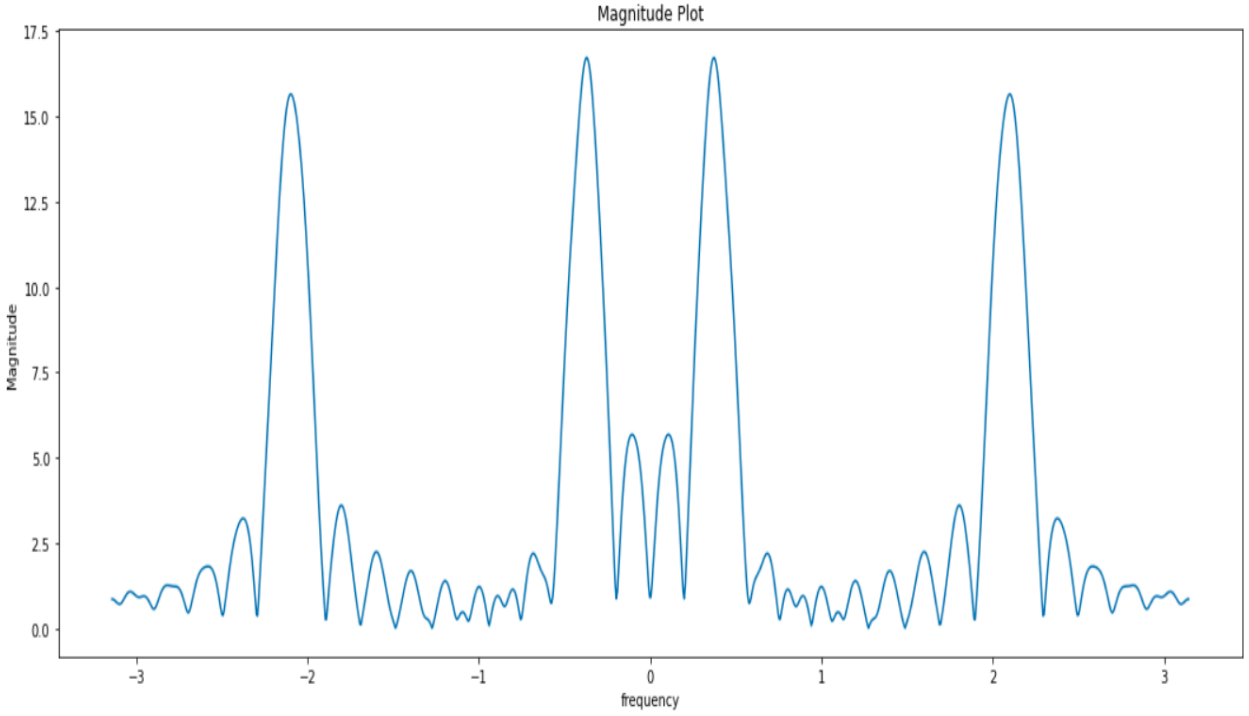
```
#Question 1

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

def plot_dtft(n, x_n):
omega = np.linspace(-np.pi, np.pi, 1000)
X_omega = np.zeros(len(omega), dtype=np.complex)
for i in range(len(omega)):
for j in range(len(x_n)):
X_omega[i] = X_omega[i] + x_n[j]*np.exp(-1j*omega[i]*n[j])
X_omega_mag = np.abs(X_omega)
X_omega_phase = np.angle(X_omega)
plt.figure(figsize=(18, 7))
plt.plot(omega, X_omega_mag)
plt.xlabel('frequency')
plt.ylabel('Magnitude')
plt.title('Magnitude Plot')
plt.show()
print('\n')
plt.figure(figsize=(18, 7))
plt.plot(omega, X_omega_phase)
plt.xlabel('frequency')
plt.ylabel('Phase Angle')
plt.title('Phase Plot')
plt.show()
print('\n')

#Part 1
n = np.arange(0, 64)
x1_n = np.hstack((np.sin(np.pi/8*n[0:32]),
np.sin(2*np.pi/3*n[32:64])))
plt.figure(figsize=(18, 7))
plt.stem(n, x1_n)
plt.xlabel('n')
plt.ylabel('x1[n]')
plt.title('Time Domain Plot of x1[n]')
plt.show()
print('Frequency Domain Plots of x1[n] : \n')
plot_dtft(n, x1_n)
```

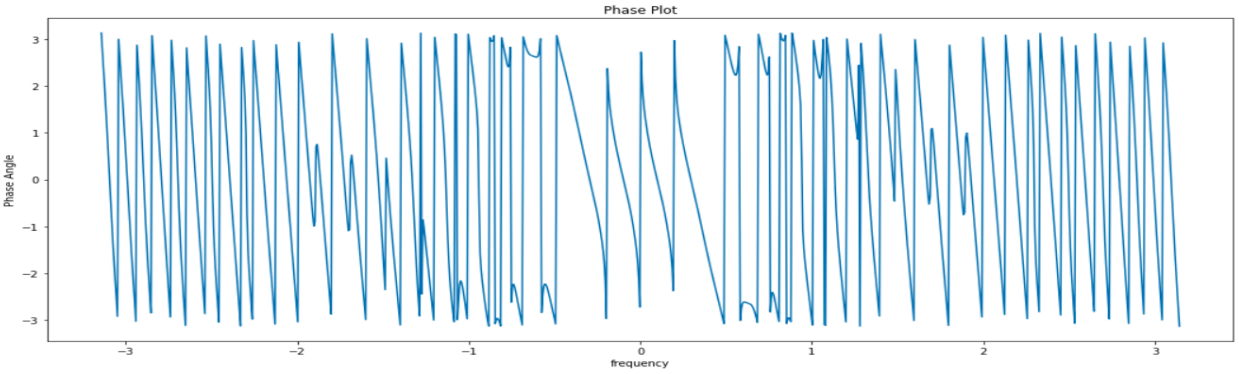
```
#Part 2
n = np.arange(0, 100)
x2_n = np.random.normal(0, 1, 100)
plt.figure(figsize=(18, 7))
plt.stem(n, x2_n)
plt.xlabel('n')
plt.ylabel('x2[n]')
plt.title('Time Domain Plot of x2[n]')
plt.show()
print('Frequency Domain Plots of x2[n] : \n')
plot_dtft(n, x2_n)

#Part 3
sample_rate, x3_n = wavfile.read('/content/gdrive/My Drive/5th
Semester/EC867/loudclap.wav')
print('Sample Rate = %d\n'%(sample_rate))
n = np.arange(len(x3_n))
plt.figure(figsize=(18, 7))
plt.stem(n, x3_n)
plt.xlabel('n')
plt.ylabel('x3[n]')
plt.title('Time Domain Plot of x3[n]')
plt.show()
print('Frequency Domain Plots of x3[n] : \n')
plot_dtft(n, x3_n)
```

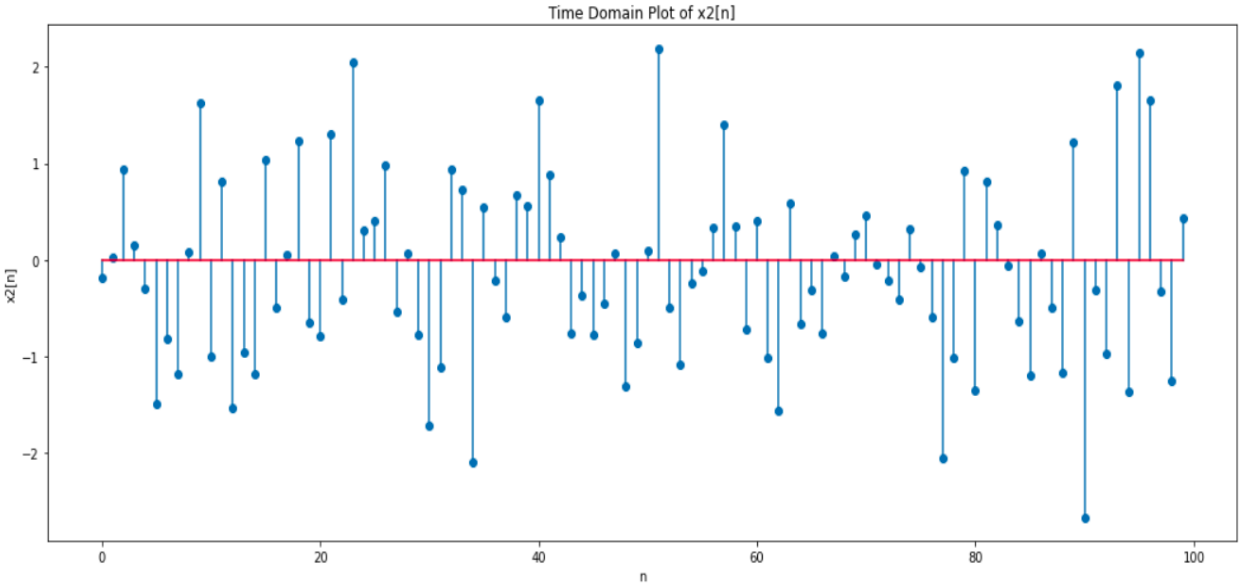
Output:



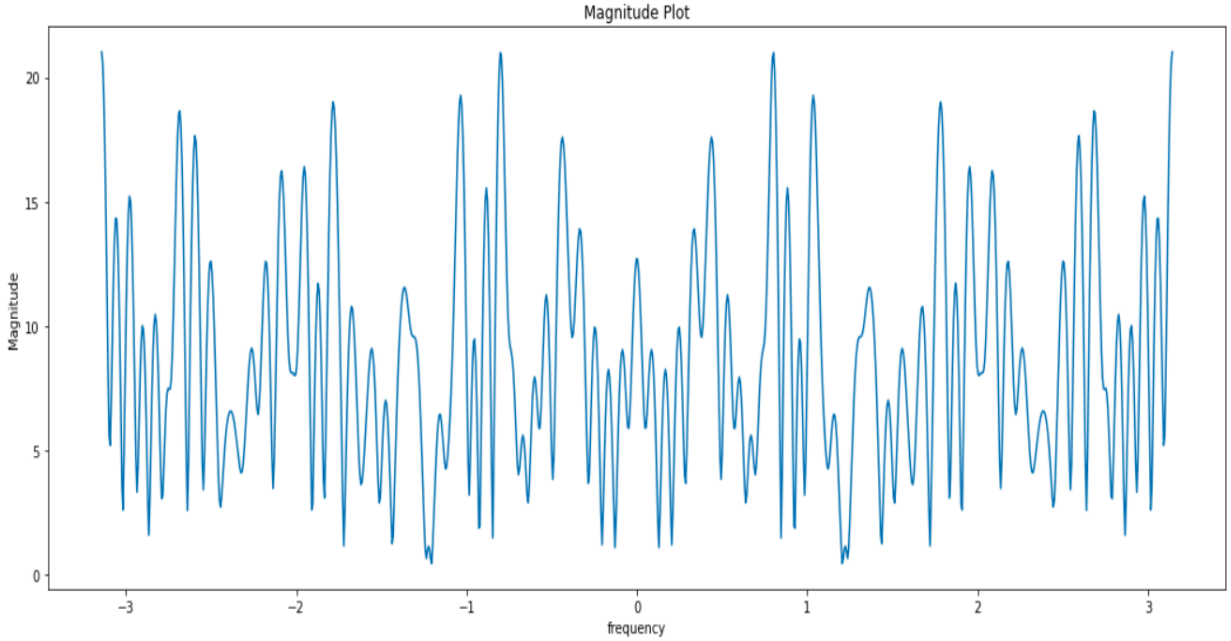
Frequency Domain Plots of x1[n] :

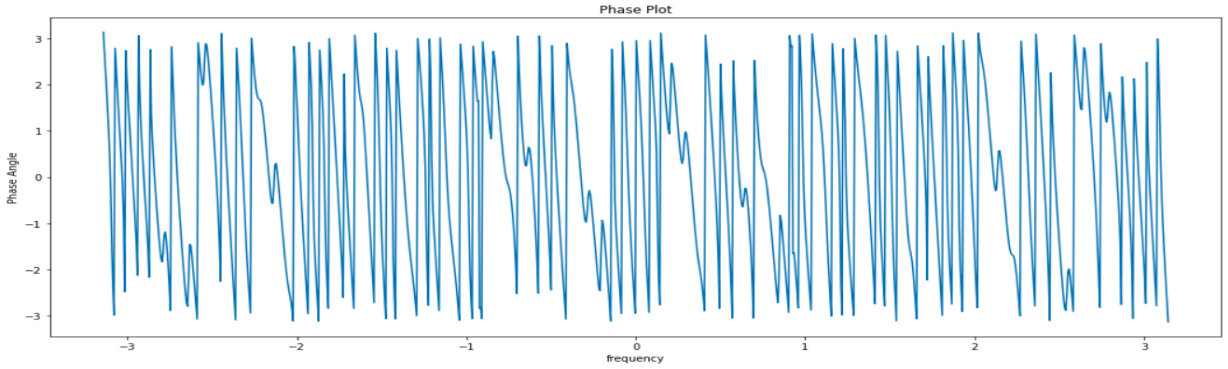


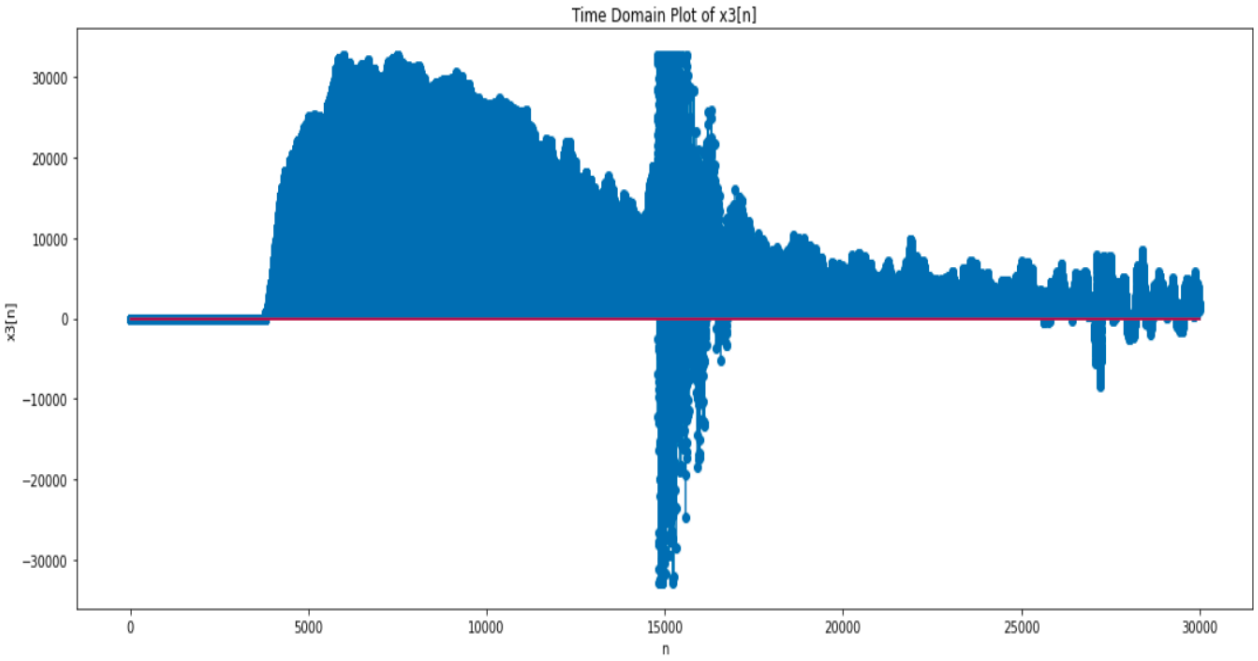


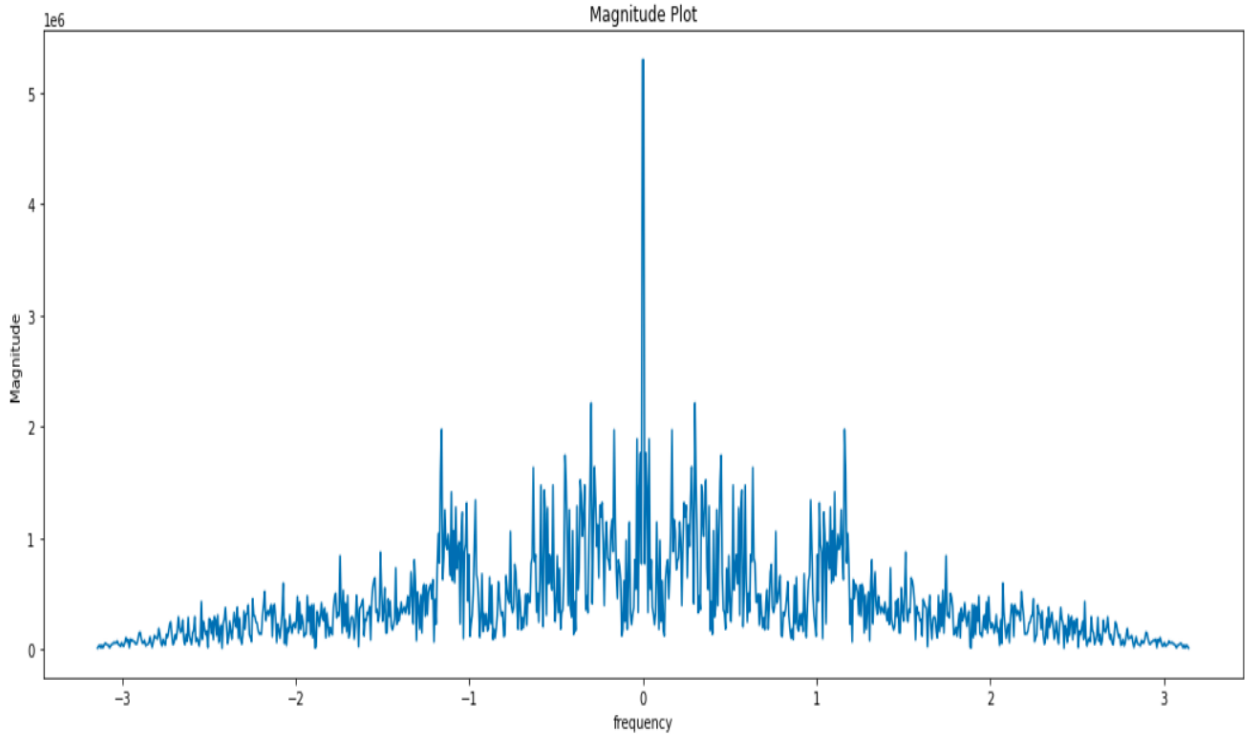


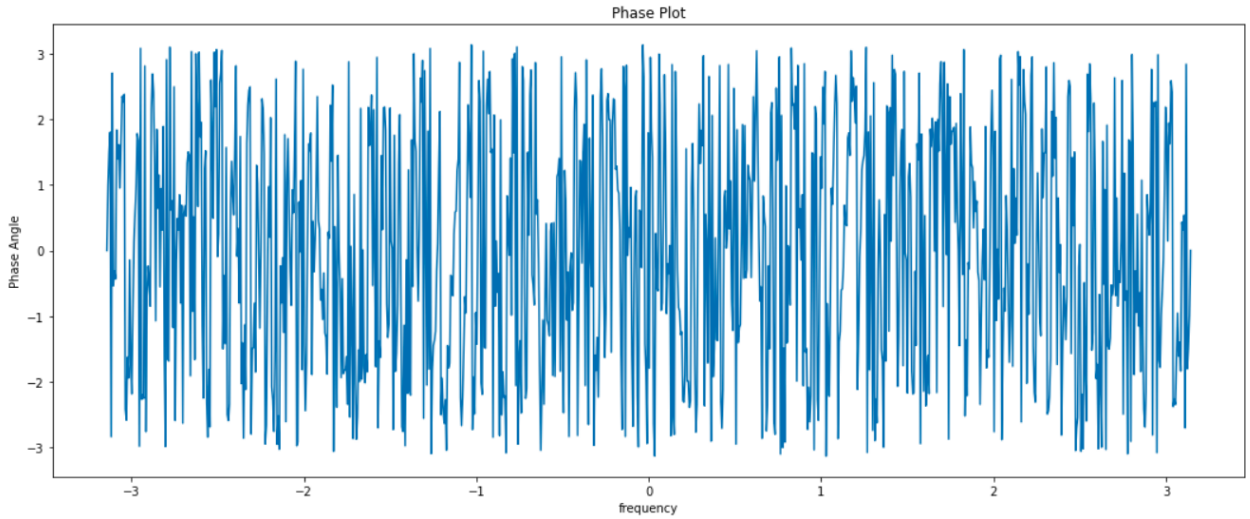
Frequency Domain Plots of x2[n] :











Inferences:

- The signal of part (a) has a combination of two sinusoids of different periods. The first 31 samples of the signal corresponds to a sinusoid of period N = 16 while the next 32 samples of the signal correspond to a sinusoid of period N = 3 as can be observed from the time domain plots and can also be verified theoretically by setting $x[n] = x[n + N]$. The frequencies of the 2 sinusoids are given by $\frac{\pi}{8} \sim 0.4$ and $\frac{2\pi}{3} \sim 2.1$. We can observe that the magnitude spectrum has peaks at these two frequencies and it also has non – zero values at other frequencies which must arise due to the interface between the two sinusoids which is sharp.

- For the Gaussian sequence of part (b), we observe in the time domain that the signal amplitudes are almost equally likely to be positive or negative, thus justifying the fact that

we have a zero mean. Further, the signal amplitudes rarely go beyond 1 or -1 mark which is the interpretation of the unit variance. Since the signal is random, it is difficult to make predictions about the signal in the frequency domain. From the frequency domain plots, we observe that both low and high frequencies are present in the generated Gaussian noise. It is somewhat similar to white noise but the magnitude spectrum is not exactly uniform.

- The signal corresponding to a loud clap in part (c) must theoretically have been an impulse function and should have had a uniform frequency distribution at all frequencies. However, we observe that the practical audio signal which we have generated has a lot of noise and the impulse function seems more rounded instead of a single value. Thus, the frequency spectrum is also not corresponding to an exact straight line but seems to have considerable proportions of all frequencies albeit in varying amounts.

Q2: Consider the following two channel orthonormal basis expansion of the signal

$$x[n] = \sum_{k\epsilon\mathbb{Z}} \alpha[k]g[n-2k] + \sum_{k\epsilon\mathbb{Z}} \beta[k]h[n-2k]$$

$$\text{where,} \quad g[n] = \frac{1}{\sqrt{2}}[\delta[n] + \delta[n-1]]$$

$$\text{and,} \quad h[n] = \frac{1}{\sqrt{2}}[\delta[n] - \delta[n-1]] \ \backslash$$

Compute $\alpha[k]$ and $\beta[k]$ for each of the signals in question 1 and plot. Also synthesize and plot the individual channel outputs $\sum_{k\epsilon\mathbb{Z}} \alpha[k]g[n-2k]$ and $\sum_{k\epsilon\mathbb{Z}} \beta[k]h[n-2k]$. Record your inferences.

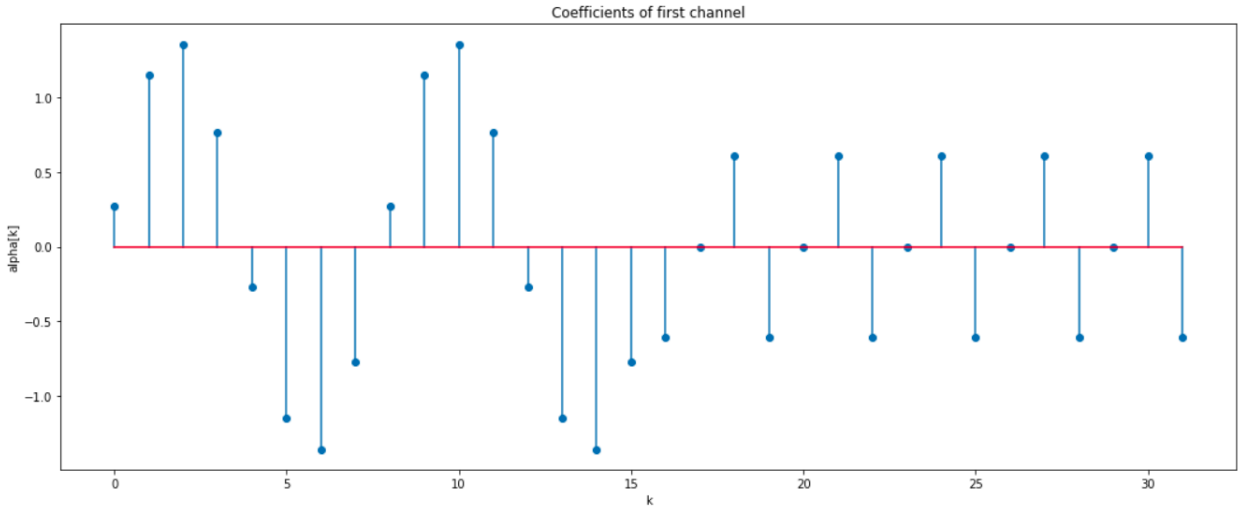
Ans. **Code:**

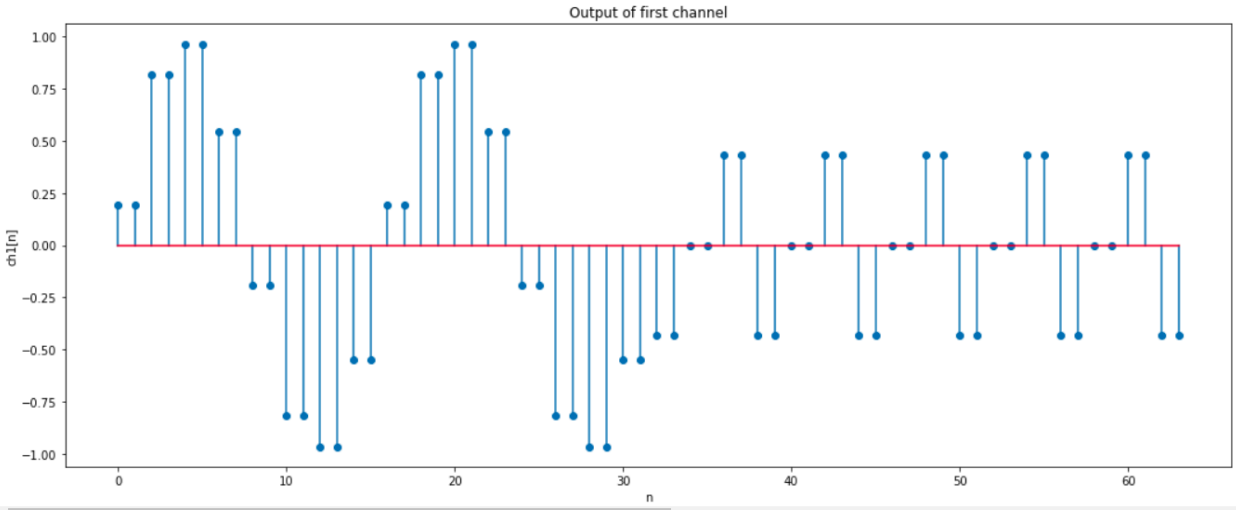
```
#Question 2
def two_channel_ONFB(n, x_n):
if (len(n)%2 != 0):
x_n = np.hstack((x_n, np.array([0])))
n = np.hstack((n, np.array([len(n)])))
temp_index = np.arange(2)
g_n = np.zeros(2, dtype = np.float)
g_n[0] = 1/np.sqrt(2)
g_n[1] = 1/np.sqrt(2)
g_neg_n = g_n
a_k = np.convolve(x_n, g_neg_n)
alpha_k = a_k[1::2]
alpha_k = np.reshape(alpha_k, (-1, 1))
a1_k = np.hstack((alpha_k, np.reshape(np.zeros(len(alpha_k)), (-1,
1))))
a1_k = np.reshape(a1_k, (1, -1))
a1_k = a1_k[0]
a1_k = a1_k[0:len(a1_k)-1]
ch1_out = np.convolve(a1_k, g_n)
```

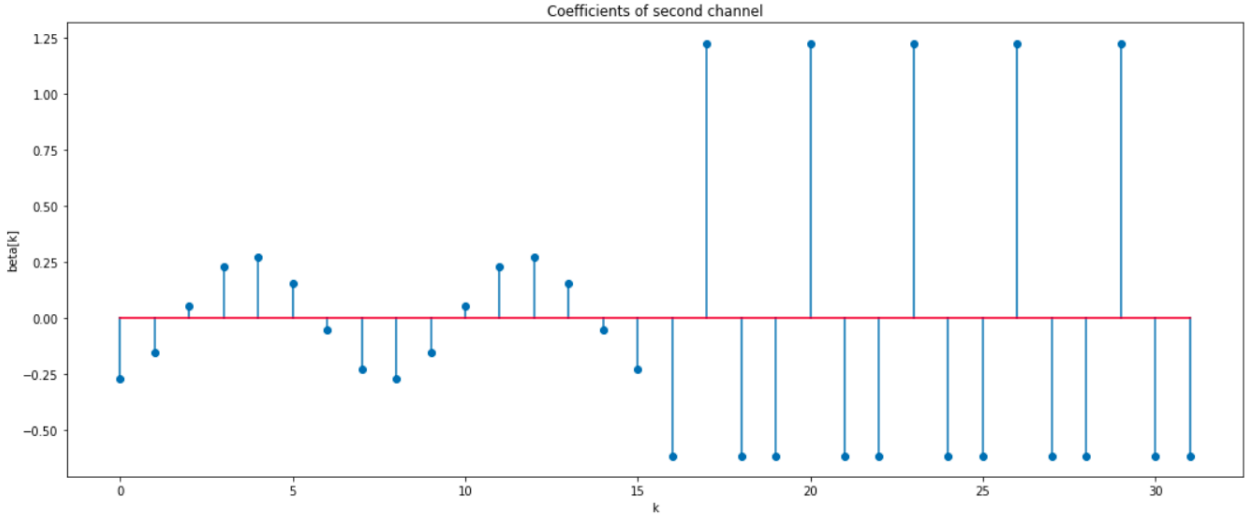
```
h_n = np.zeros(2, dtype = np.float)
h_n[0] = 1/np.sqrt(2)
h_n[1] = -1/np.sqrt(2)
h_neg_n = h_n[::-1]
b_k = np.convolve(x_n, h_neg_n)
beta_k = b_k[1::2]
beta_k = np.reshape(beta_k,(-1, 1))
b1_k = np.hstack((beta_k, np.reshape(np.zeros(len(beta_k)),(-1, 1))))
b1_k = np.reshape(b1_k, (1, -1))
b1_k = b1_k[0]
b1_k = b1_k[0:len(b1_k)-1]
ch2_out = np.convolve(b1_k, h_n)
plt.figure(figsize = (18, 7))
plt.stem(alpha_k)
plt.xlabel('k')
plt.ylabel('alpha[k]')
plt.title('Coefficients of first channel')
plt.show()
plt.figure(figsize=(18, 7))
plt.stem(ch1_out)
plt.xlabel('n')
plt.ylabel('ch1[n]')
plt.title('Output of first channel')
plt.show()
plt.figure(figsize=(18, 7))
plt.stem(beta_k)
plt.xlabel('k')
plt.ylabel('beta[k]')
plt.title('Coefficients of second channel')
plt.show()
plt.figure(figsize=(20, 10))
plt.stem(ch2_out)
plt.xlabel('n')
plt.ylabel('ch2[n]')
plt.title('Output of second channel')
plt.show()

#Part 1
n = np.arange(0, 64)
print('Part 1 : \n')
two_channel_ONFB(n, x1_n)
print('\n')

#Part 2
n = np.arange(0, 100)
print('Part 2 : \n')
two_channel_ONFB(n, x2_n)
print('\n')

#Part 3
n = np.arange(len(x3_n))
print('Part 3 : \n')
two_channel_ONFB(n, x3_n)
print('\n')
```

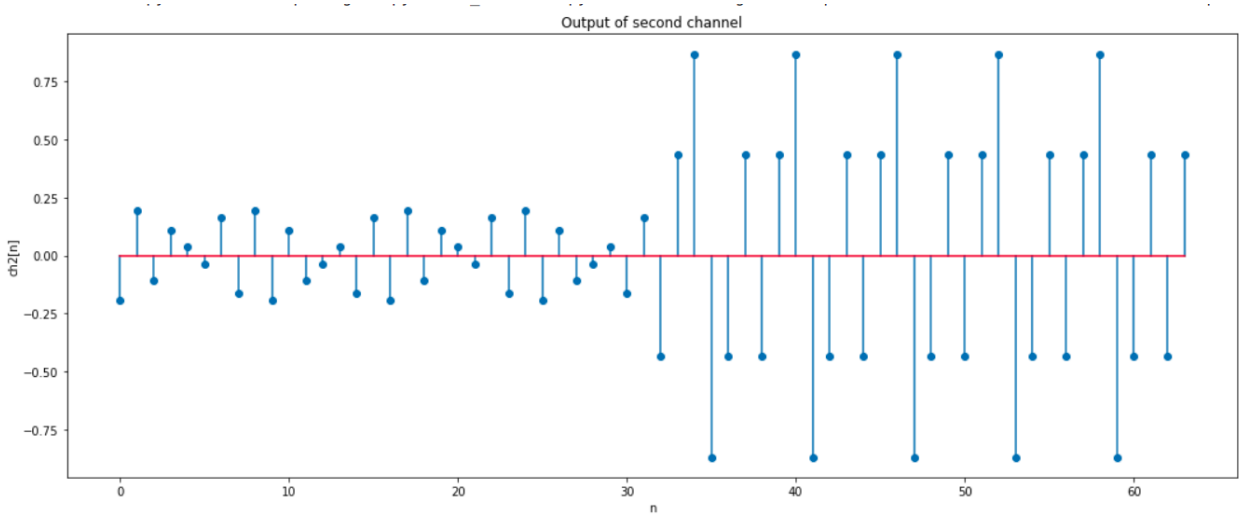
Output:

1. Part (a) : x1[n]

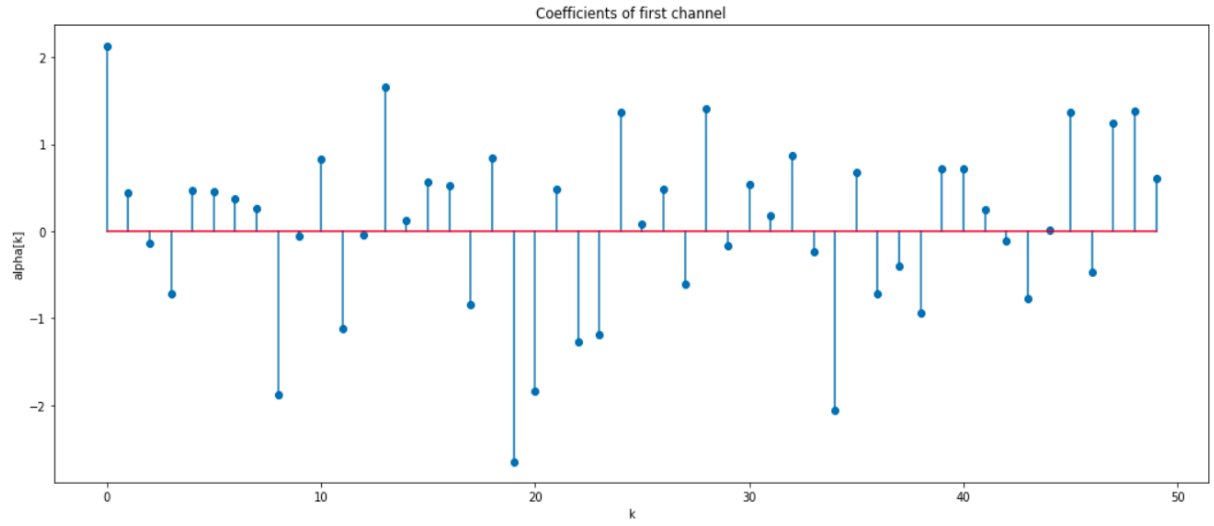


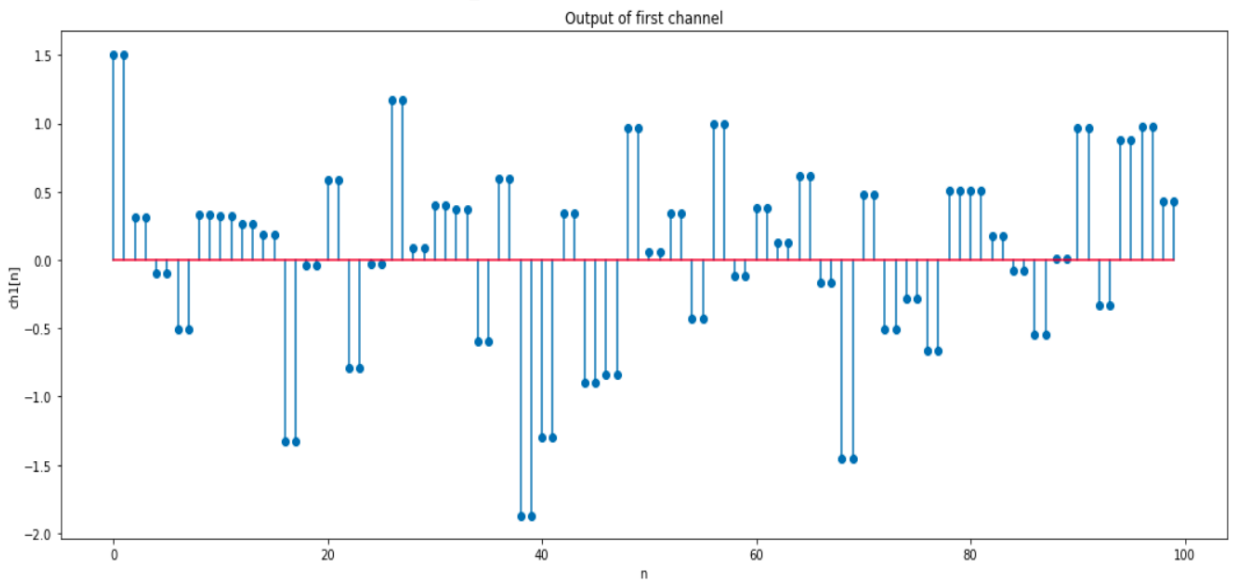


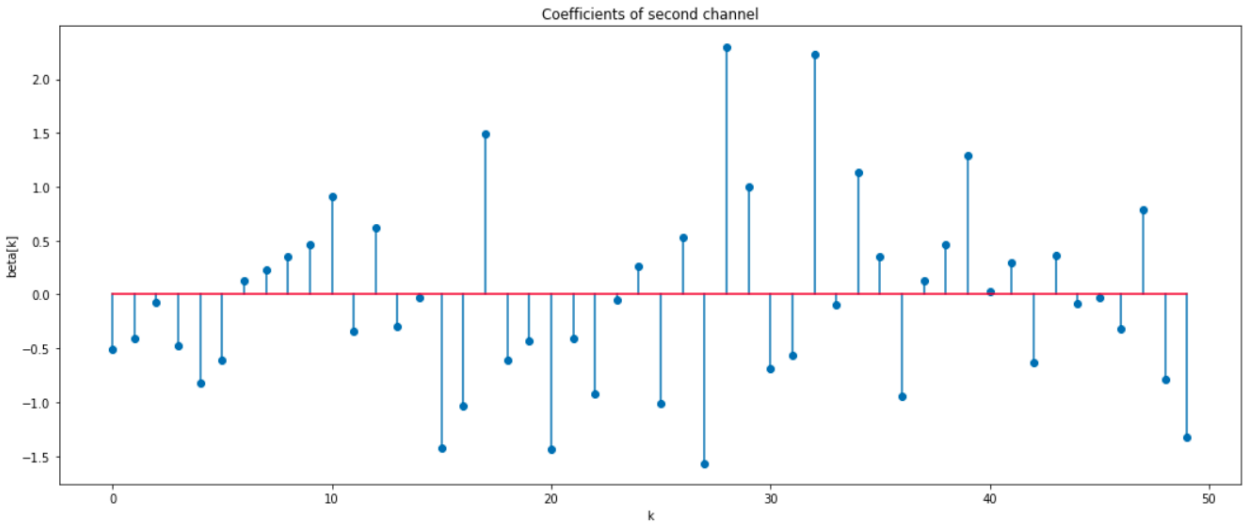


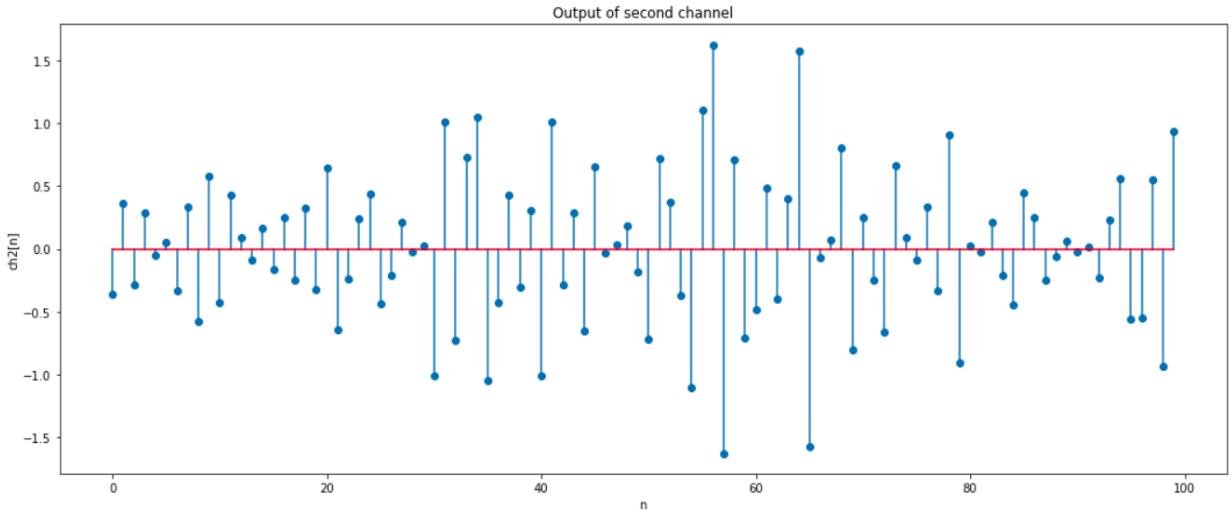

Output of second channel
ch2[n]
n

Part (b) : x2[n]

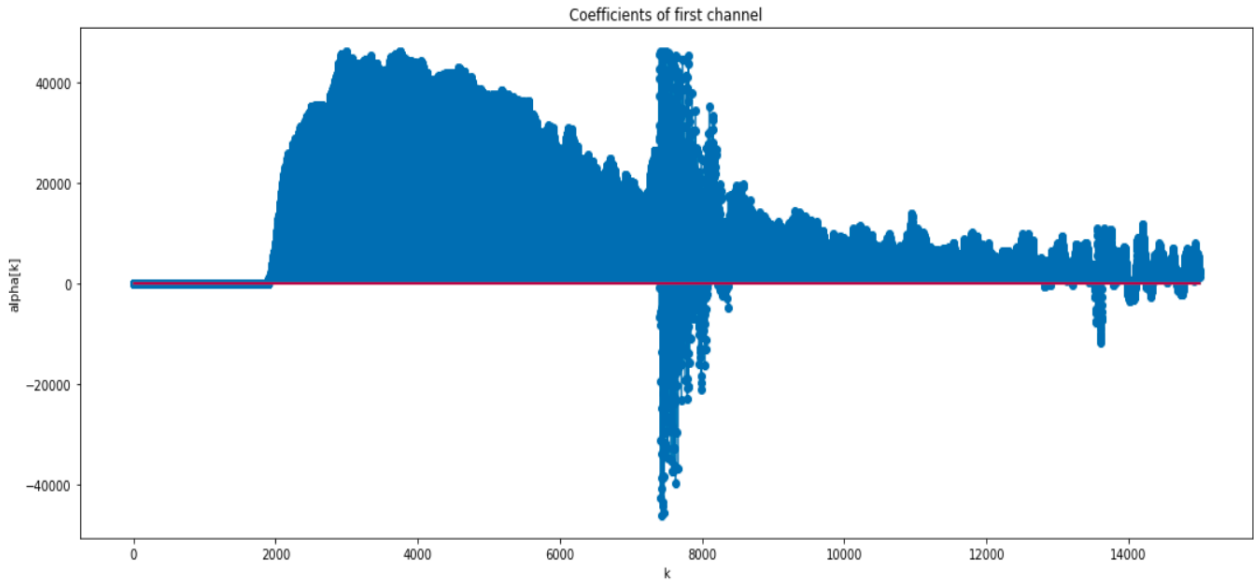

Coefficients of first channel
alpha[k]
k

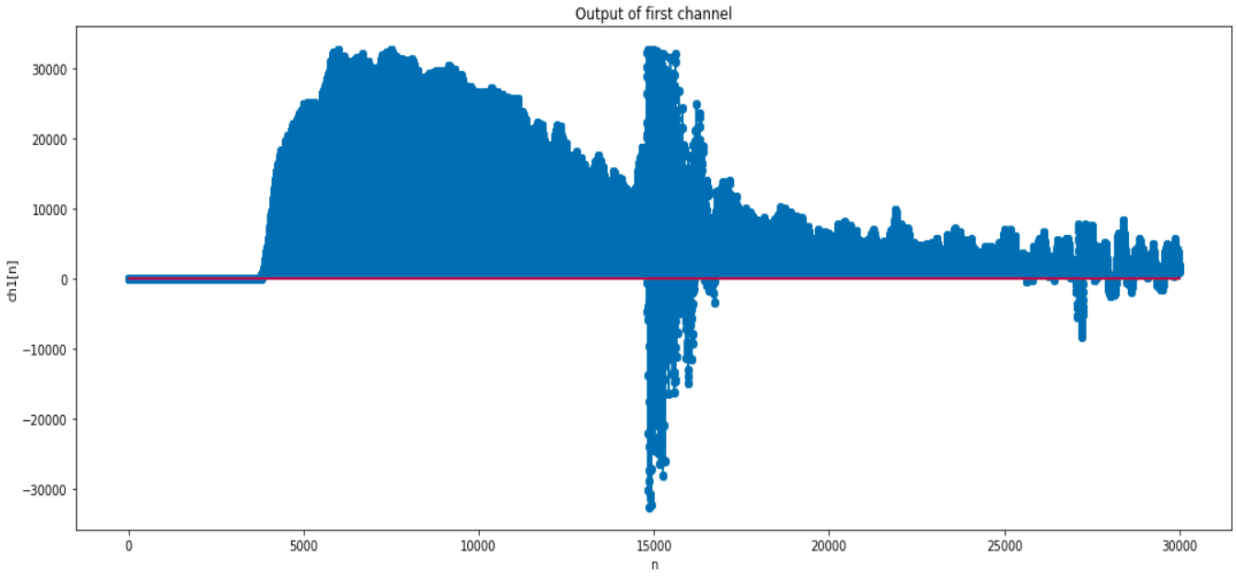

Output of first channel
ch1[n]
n

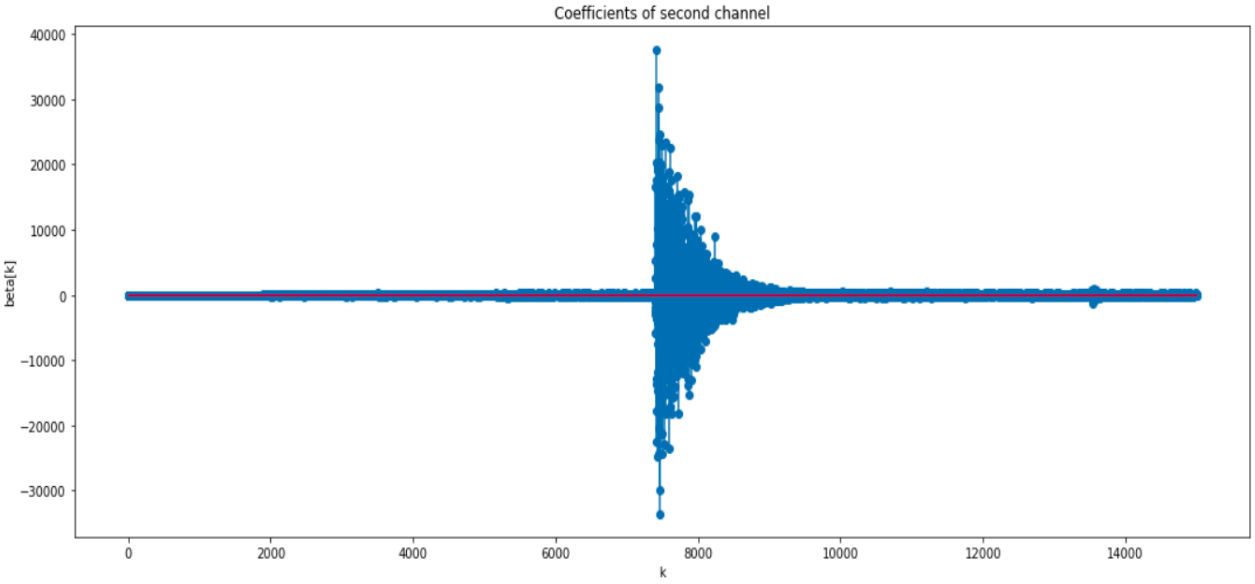

Coefficients of second channel
beta[k]
k

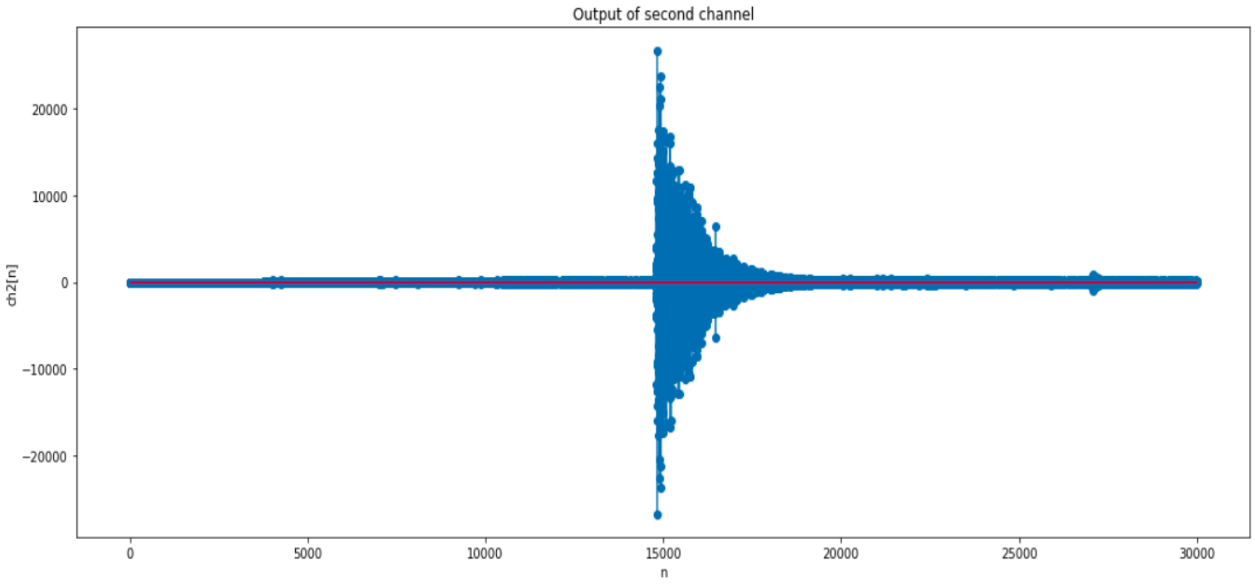

Output of second channel
ch2[n]
n

Part (c) : x3[n]


Coefficients of first channel
alpha[k]
k


Output of first channel
ch1[n]
n


Coefficients of second channel
beta[k]
k


Output of second channel
ch2[n]
n

Inferences:

- The interpretation of the individual channel coefficients $\alpha[k]$ and $\beta[k]$ is as follows. Each $\alpha[k]$ denotes the amout of the signal g[n-2k] present at the time index n = 2k and n = (2k+1). For instance, $\alpha[0]$ tells us the weight attached to g[n] at time indices 0 and 1. $\beta[k]$ conveys the exactly same informatiob about the signal h[n]. It can further be shown from frequency domain plots of g[n] and h[n] that g[n] acts as a low pass filter while h[n] acts as a high pass filter. Thus the output of channel 1 must be a low pass signal while that of channel 2 musrt be a high pass signal

- Looking at the coefficients for channel 1 for x1[n], we observe that the coefficients tend to have almost the same shape as the original time domain signal. However, there is a significant difference in the fact for the first 32 samples, the coefficients of the first channel appear to be larger while for the next 32 samples, the coefficients of the first channel appear to be smaller while those of the second channel appear to be larger. This tells us that the first 32 samples have a predominant low frequency component while the next 32 samples have a predominant high frequency component. This is as we might expect theoretically since the first 32 samples of x1[n] correspond to a signal of period N = 16 i.e. lower frequency while the next 32 samples correspond to a signal of period N =3 i.e. higher frequency.

- The output of the first channel (low pass channel) suppresses the high frequency component from samples 32 to 63 whereas the low frequency sinusoid in samples numbered 0 to 31 are passed almost entirely. The output of the second channel (high pass channel) suppresses the low frequency sinusoid (first 32 samples) while almost entirely passing the high frequency sinusoid (latter 32 samples).

- Looking at the coefficients of the two channels for the gaussian signal x2[n], we can predict that the first few samples asnd the last few samples have, in general, a lower frequency and thus the coefficients of the first channel are domainant near 0 and 50 (half the total number of samples which is the number of coefficients of each of the channels) while the coefficients of the second channel seem to be dominant somewhere near n = 30. This is expected to vary with execution since the signal generated is random and does not provide any information in general about a Gaussian signal. Similar observations can mbe made for the output of each of the channels. The first channel is a low frequency channel and has high values near 0 and 100 and somewhere near n = 40. This means that the Gaussian signal we generated has lower frequencies near these indices of time. Similarly, the second cbannel which is high pass has high amplitudes near n = 60 which signifies that the signal has high frequency components near these time indices.

- For the signal x3[n], we can observe that the coefficients of the first channel and its output almost exactly resemble the original signal whereas the second channel output has component somewhere close to n = 15000. This can be justified by the fact that the DTFT of the overall signal when plotted has a huge peak at zero frequency which signifies that most of the signal is actuall low pass and thus the first channel contains most of the information about the signal. Another interpretation can be made in this cas. Near n = 15,000 we can observe that the signal resembles an impulse function. We know that in the frequency domain an impulse function gives a umiform plot. Thus, both the first and second channel give dominant values near n = 15,000 verifying that our signal represents an impulse function at this time index.

Q3: Consider the following 8 channel orthonormal basis expansion of the signal

$$x[n] = \sum_{l=0}^{7} \sum_{k \epsilon \mathbb{Z}} X_l[k] h_l[n-2k]$$

$$\text{where,} \quad h_l[n] = \frac{1}{\sqrt{8}} e^{\frac{j2\pi ln}{8}} [u[n] - u[n-8]]$$

Compute $X_l[k]$, $l = 0, \ldots, 7$ for each of the signals in question 1 and plot. Also synthesize and plot the individual channel outputs $X_l[k] = \sum_{k \epsilon \mathbb{Z}} X_l[k] h_l[n-8k]$. Record your inferences.

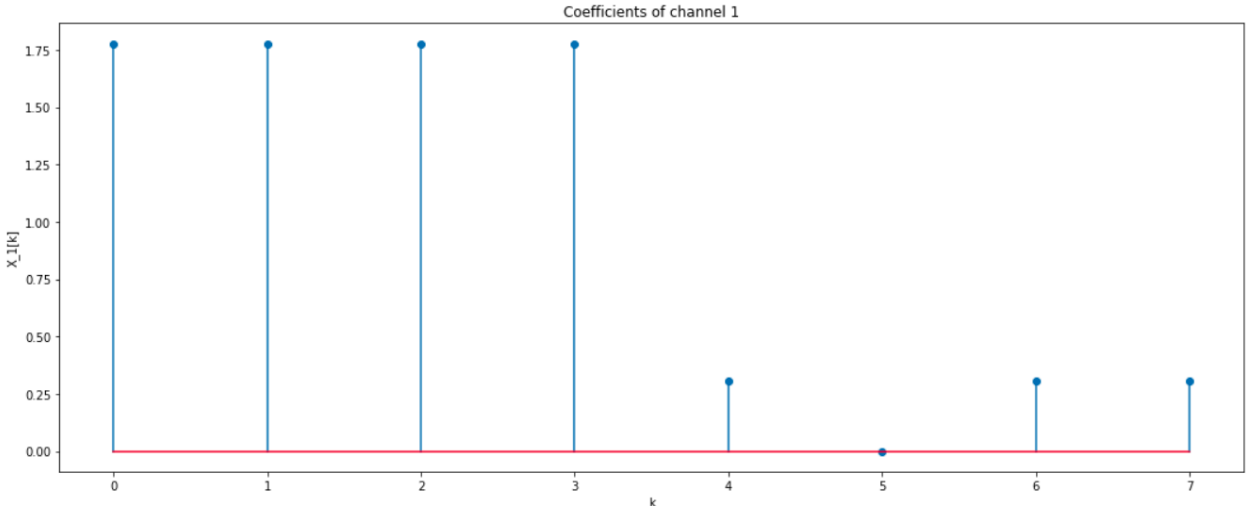
Ans. **Code:**

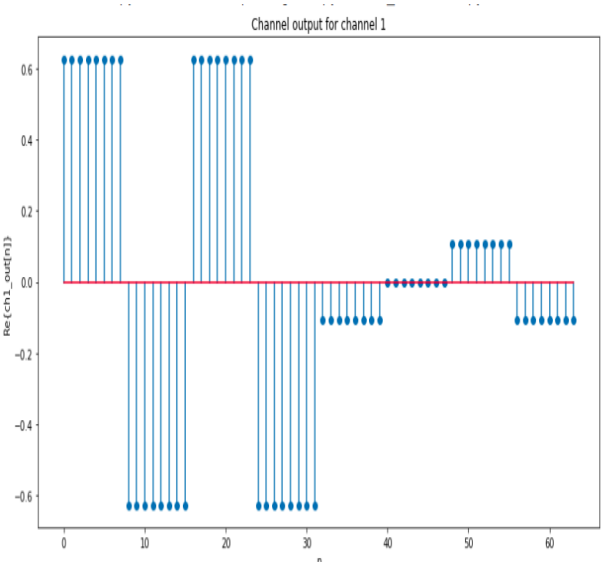
```
#Question 3

def eight_channel_ONFB(n, x_n, h_l_n, l, x_recover_n):
#Finding the coefficients by filtering
h_l_conj_n = np.conjugate(h_l_n)
h_l_conj_neg_n = h_l_conj_n[::-1]
temp_k = np.convolve(x_n, h_l_conj_neg_n)
X_l_k = temp_k[7::8]
#Finding output of lth channel
zero_array = np.zeros((len(X_l_k), 7))
X_l_k = np.reshape(X_l_k, (-1, 1))
temp1_k = np.hstack((X_l_k, zero_array))
temp1_k = np.reshape(temp1_k, (1, -1))
temp1_k = temp1_k[0]
temp1_k = temp1_k[0:len(temp1_k)-7]
chl_out_n = np.convolve(temp1_k, h_l_n)
#Plot the coefficients and channel output for the lth channel
print('Results for channel %d : \n'%(l))
plt.figure(figsize=(18, 7))
plt.stem(np.abs(X_l_k))
plt.xlabel('k')
plt.ylabel('X_%d[k]'%(l))
plt.title('Coefficients of channel %d'%(l))
plt.show()
print('\n')
plt.figure(figsize=(29, 7))
plt.subplot(1, 2, 1)
plt.stem(np.real(chl_out_n))
plt.xlabel('n')
plt.ylabel('Re{ch%d_out[n]}'%(l))
plt.title('Channel output for channel %d'%(l))
plt.subplot(1, 2, 2)
plt.stem(np.imag(chl_out_n))
plt.xlabel('n')
plt.ylabel('Im{ch%d_out[n]}'%(l))
plt.title('Channel output for channel %d'%(l))
plt.show()
print('\n')
x_recover_n = x_recover_n+chl_out_n
```

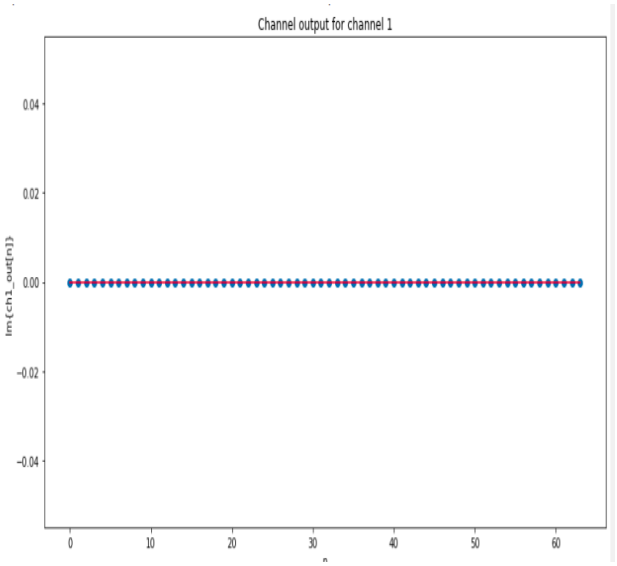
```
if (l==8):
print('Recovered Signal : \n')
plt.figure(figsize=(20, 10))
plt.stem(x_recover_n)
plt.show()
print('\n')
return x_recover_n

def eight_channel_ONFB_generatebasis(n, x_n, x_recover_n):
#Zero Padding if necessary
if (len(n)%8 != 0):
num_zeros = 8*((int)(len(n)/8)+1)-len(n)
n = np.hstack((n, np.zeros(num_zeros)))
x_n = np.hstack((x_n, np.zeros(num_zeros)))
x_recover_n = np.hstack((x_recover_n, np.zeros(num_zeros)))
n = np.arange(8)
for l in range(8):
h_l_n = np.zeros(8, dtype = np.complex)
for i in range(8):
h_l_n[i] = 1/np.sqrt(8)*np.exp(1j*2*np.pi*l*i/8)
x_recover_n = eight_channel_ONFB(n, x_n, h_l_n, l+1, x_recover_n)

#Part 1
n = np.arange(64)
print('Part 1 : \n')
x_recover_n = np.zeros(64, dtype=np.complex)
eight_channel_ONFB_generatebasis(n, x1_n, x_recover_n)
print('\n')

#Part 2
n = np.arange(100)
print('Part 2 : \n')
x_recover_n = np.zeros(100, dtype = np.complex)
eight_channel_ONFB_generatebasis(n, x2_n, x_recover_n)
print('\n')

#Part 3
n = np.arange(len(x3_n))
print('Part 3 : \n')
x_recover_n = np.zeros(len(x3_n), dtype = np.complex)
eight_channel_ONFB_generatebasis(n, x3_n, x_recover_n)
```

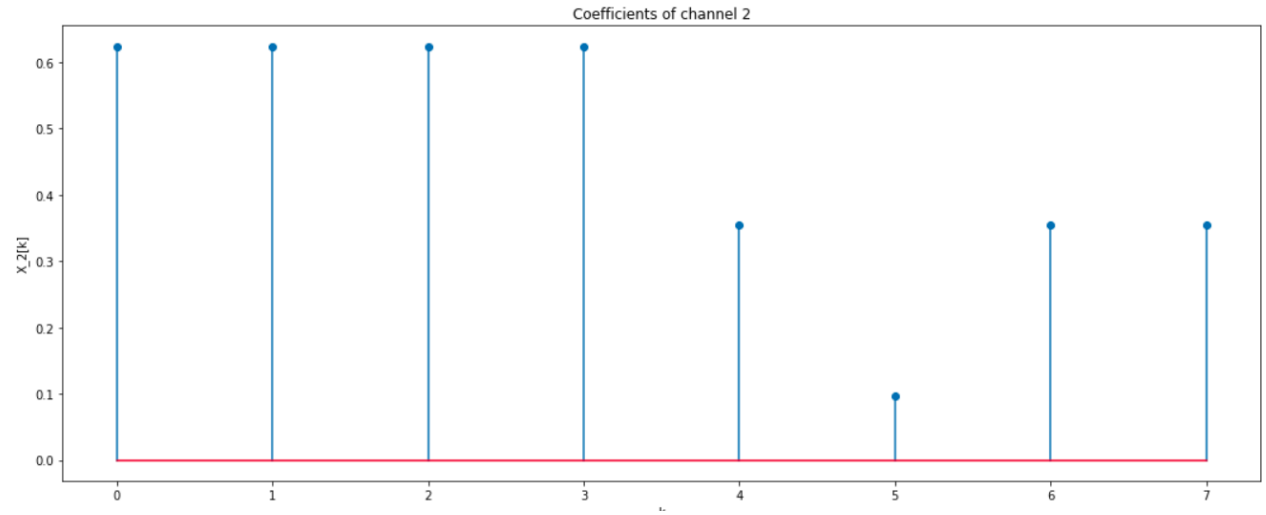
Output:

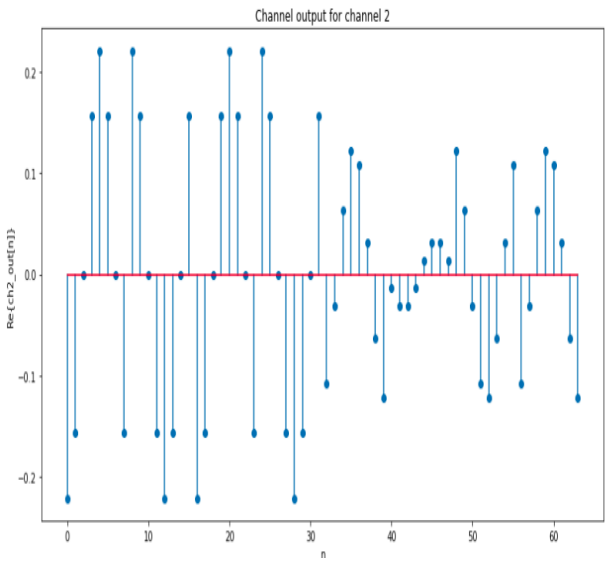
1. Part (a) : x1[n]

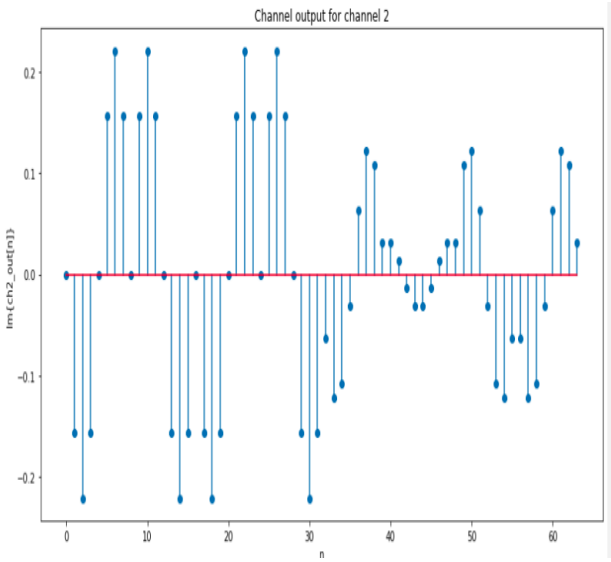


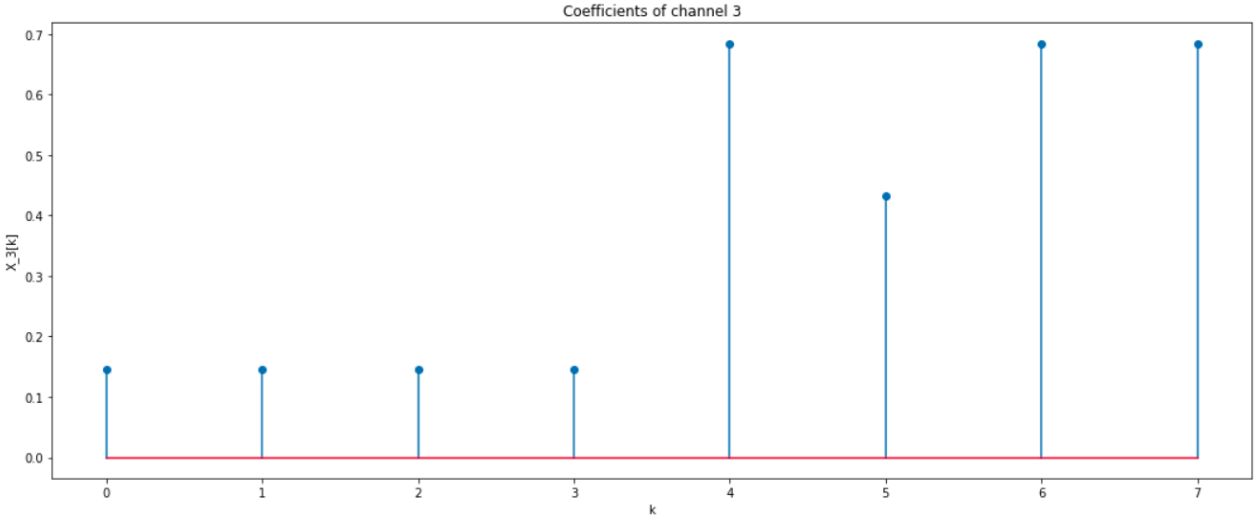


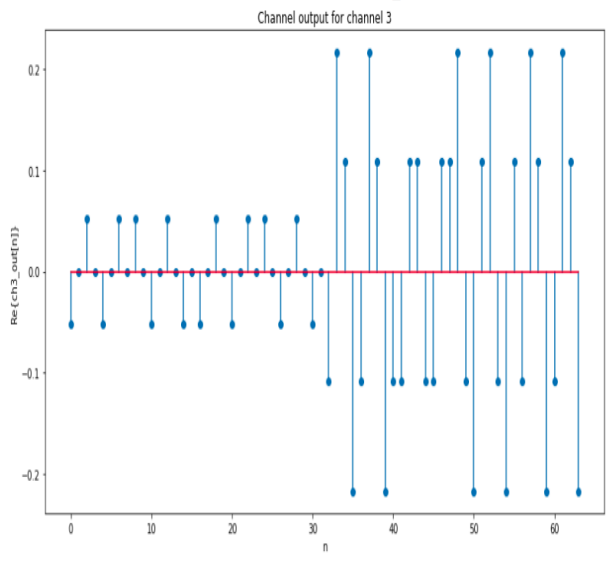


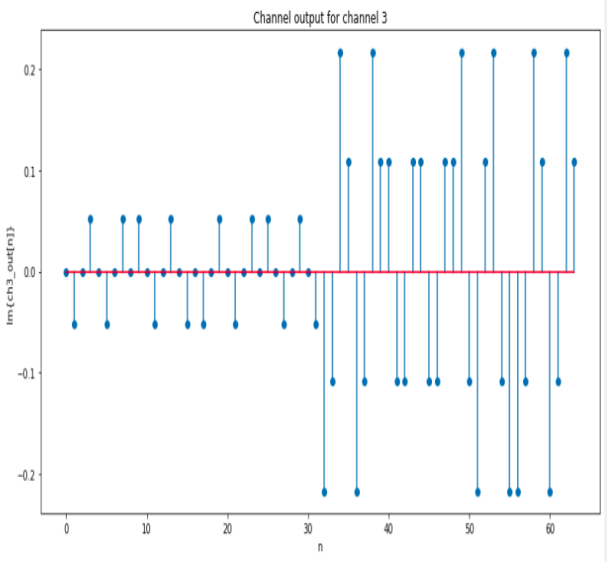


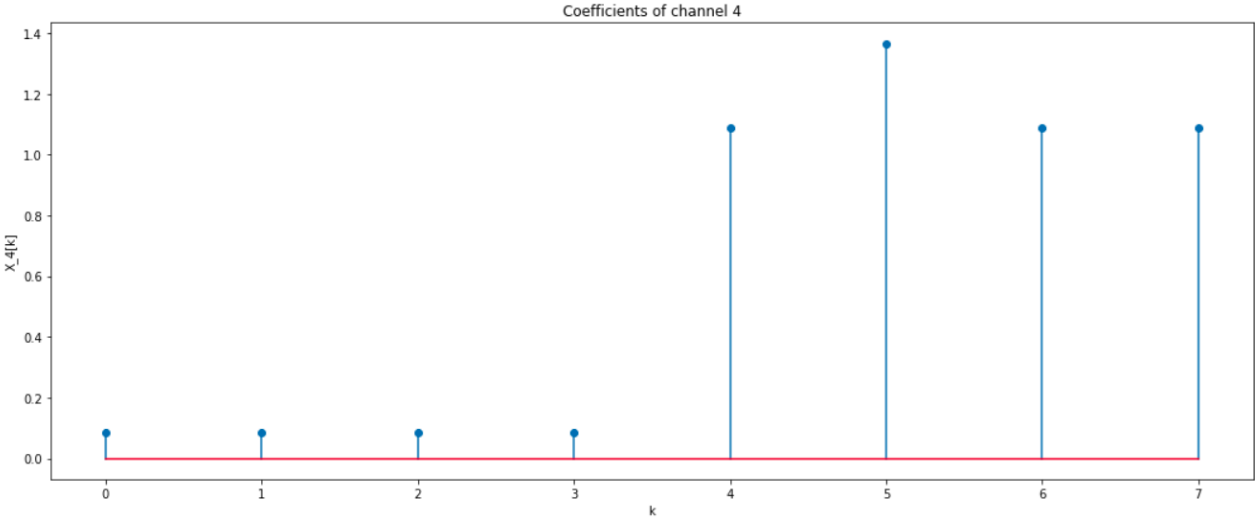

Channel output for channel 2
Re{ch2_out[n]}
n

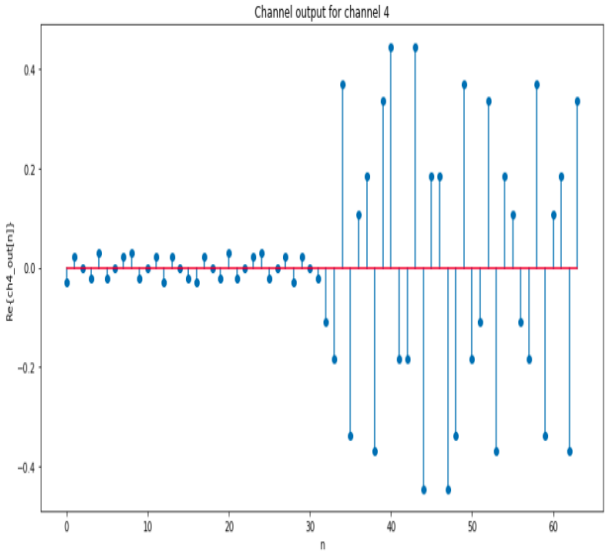

Channel output for channel 2
Im{ch2_out[n]}
n

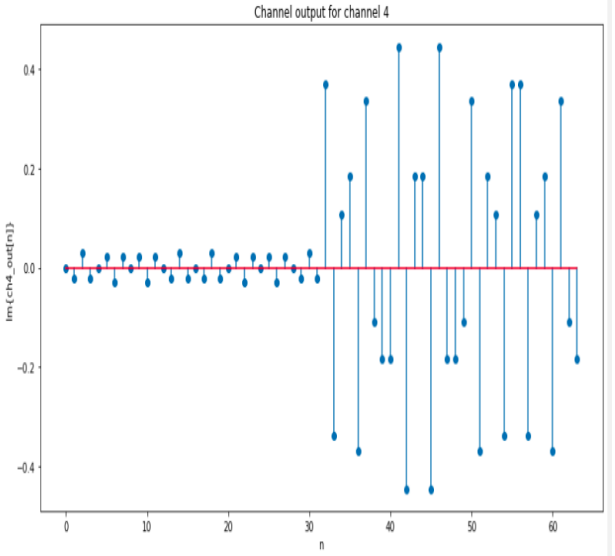

Coefficients of channel 3
X_3[k]
k

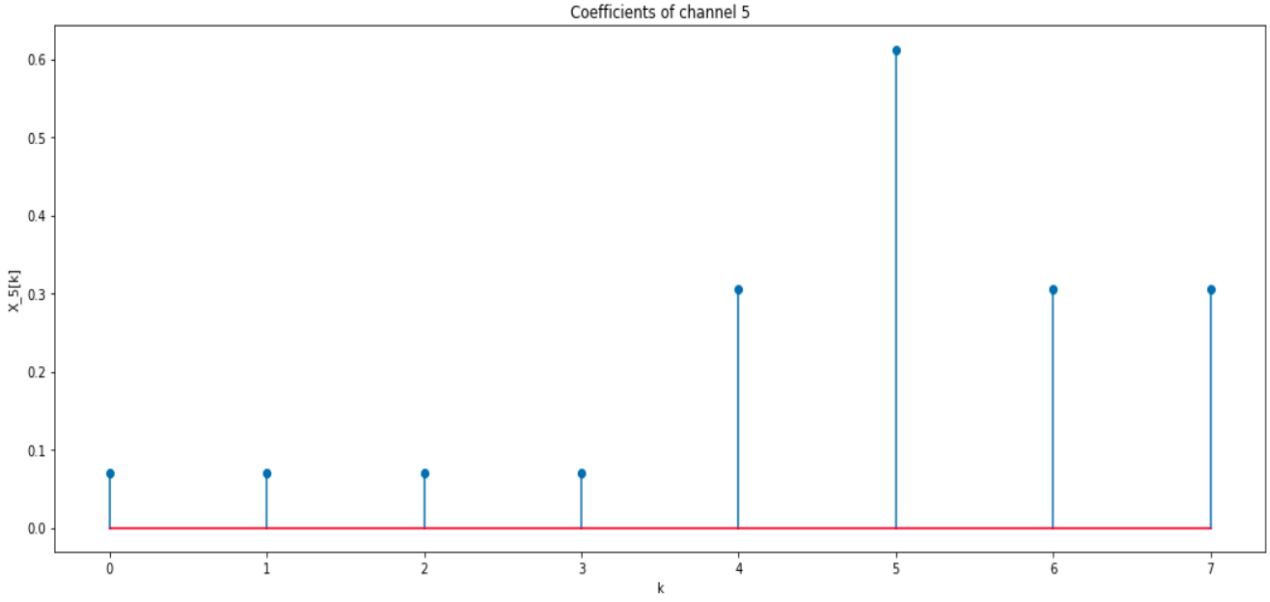

Channel output for channel 3
Re{ch3_out[n]}
n

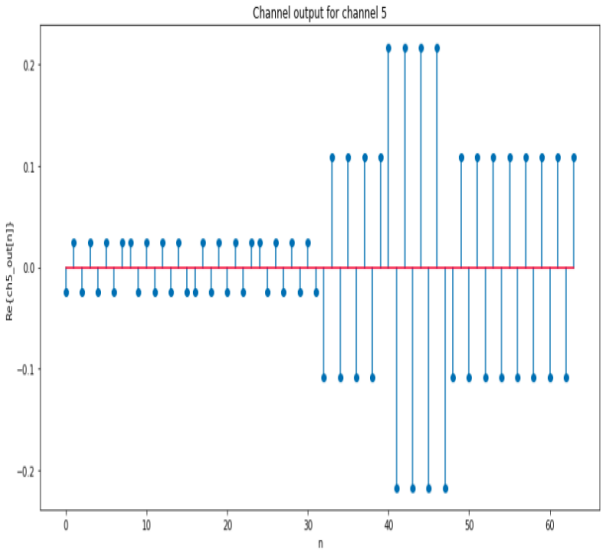

Channel output for channel 3
Im{ch3_out[n]}
n

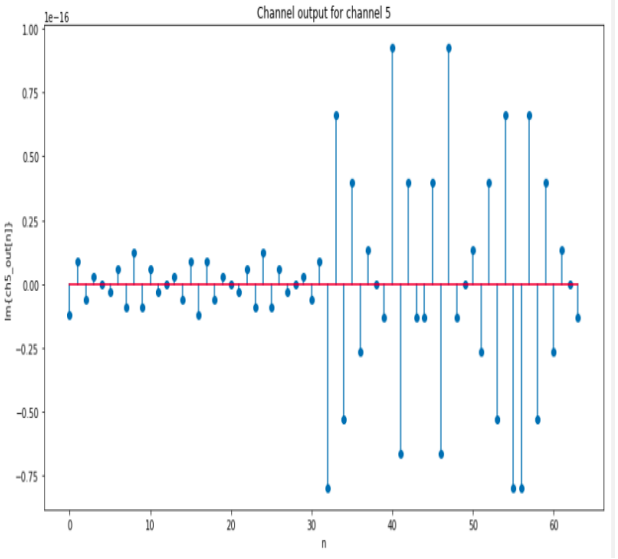

Coefficients of channel 4
X_4[k]
k

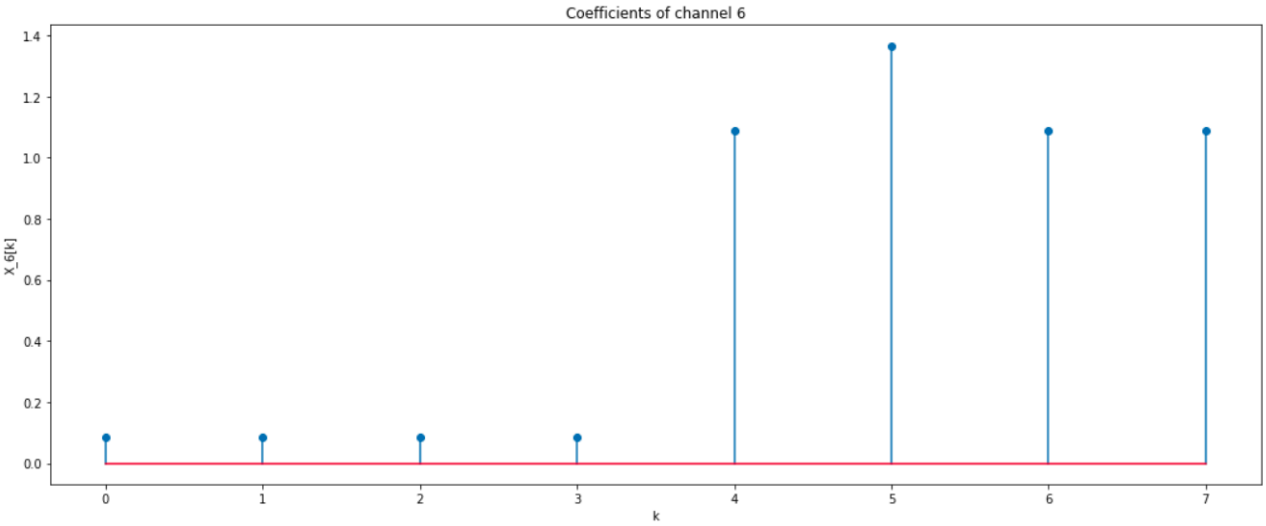

Channel output for channel 4
Re{ch4_out[n]}
n

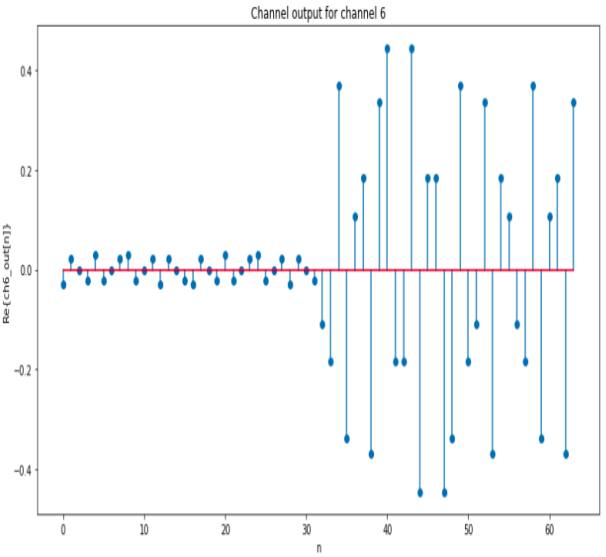

Channel output for channel 4
Im{ch4_out[n]}
n

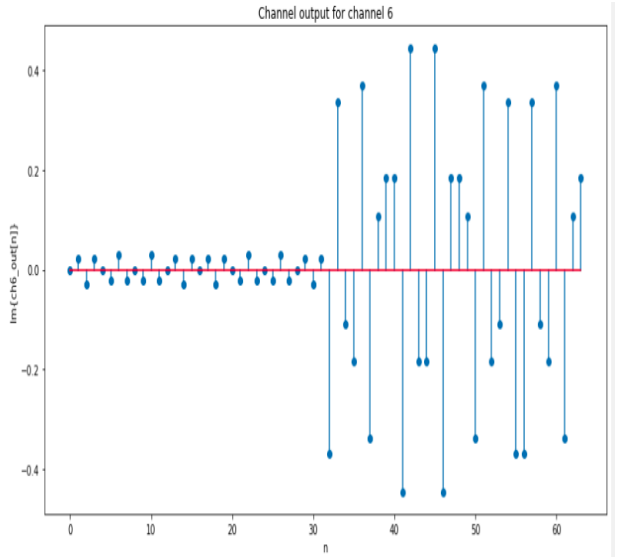

Coefficients of channel 5
X_5[k]
k

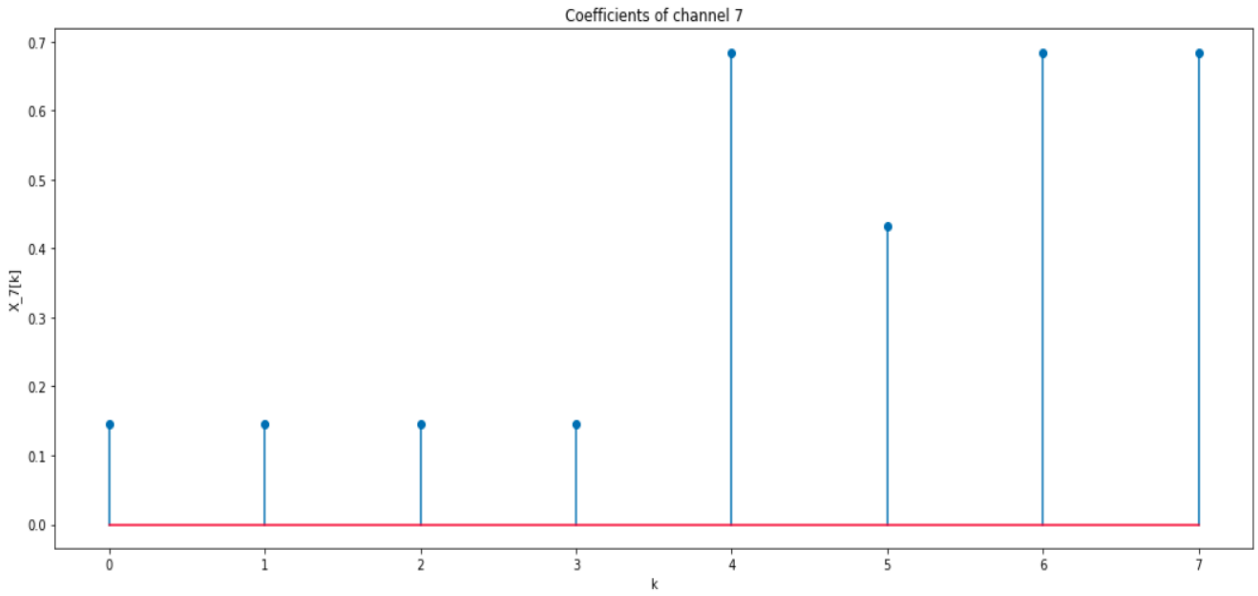

Channel output for channel 5
Re{ch5_out[n]}
n

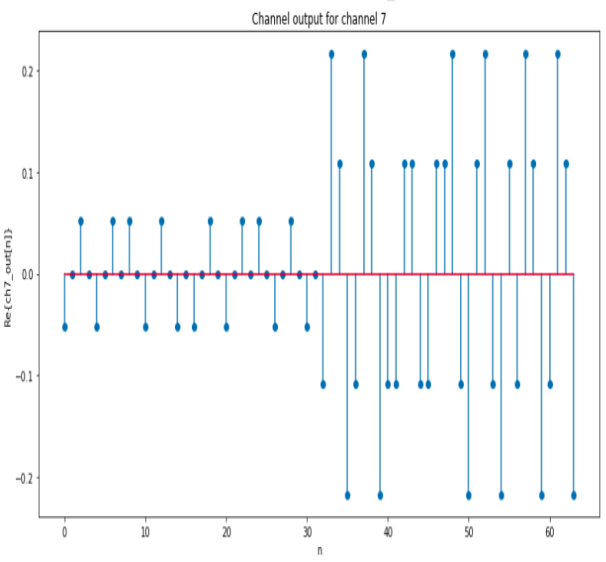

Channel output for channel 5
1e−16
Im{ch5_out[n]}
n

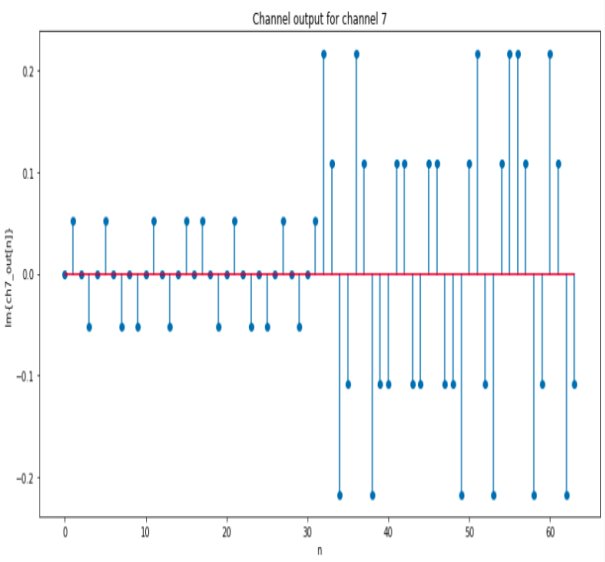

Coefficients of channel 6
X_6[k]
k

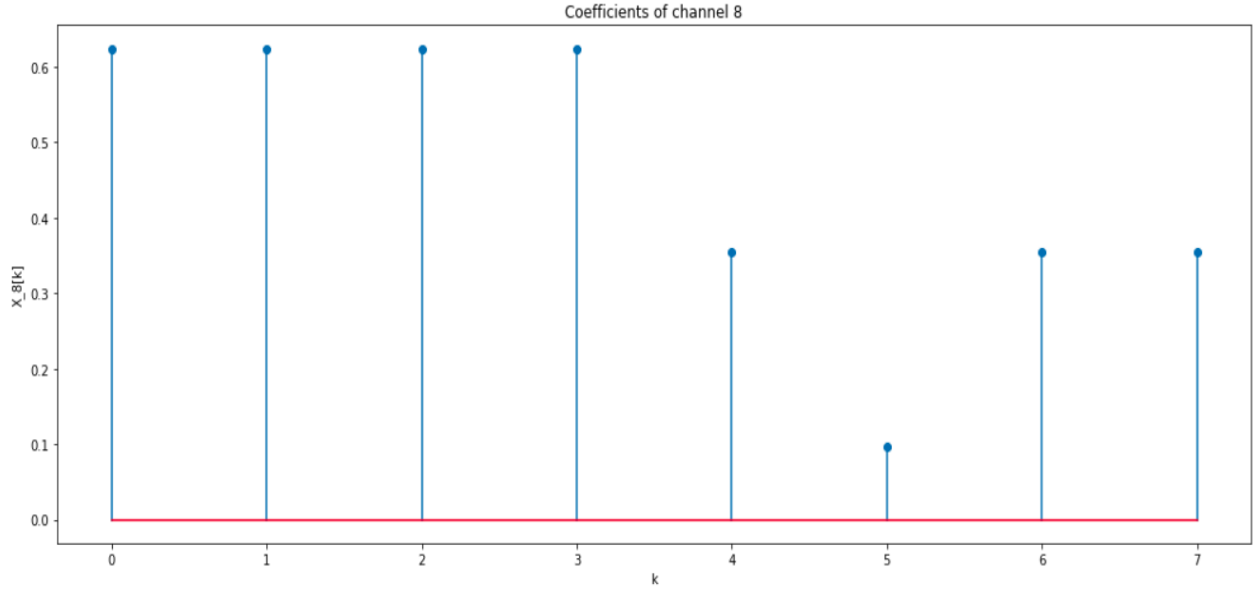

Channel output for channel 6
Re{ch6_out[n]}
n

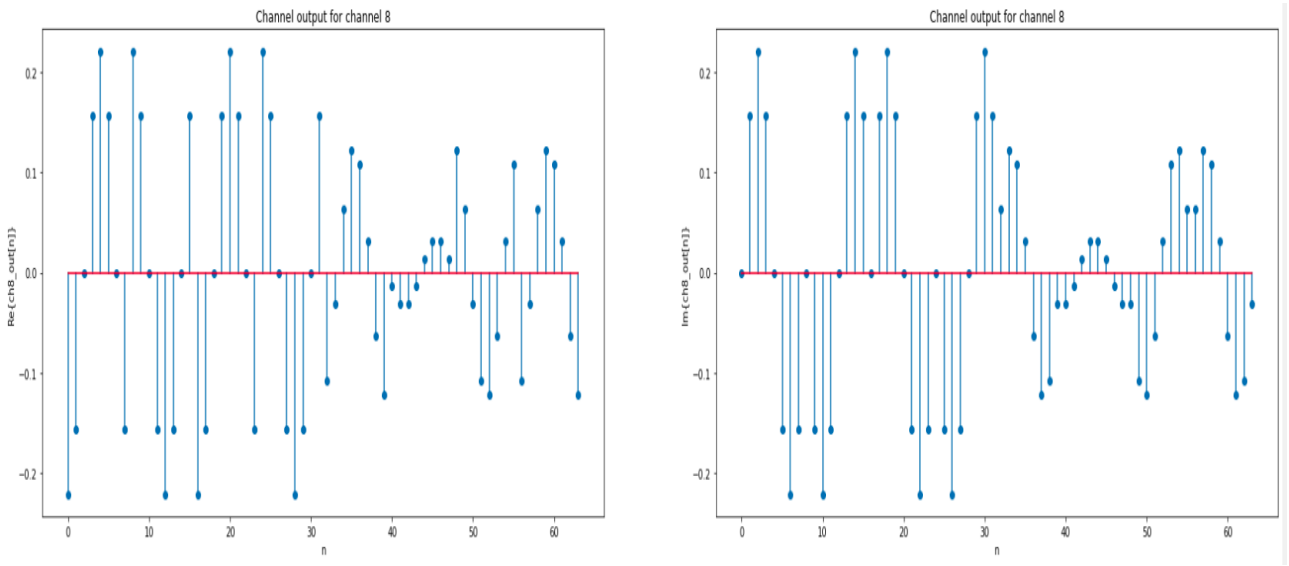

Channel output for channel 6
Im{ch6_out[n]}
n


Coefficients of channel 7
X_7[k]
k

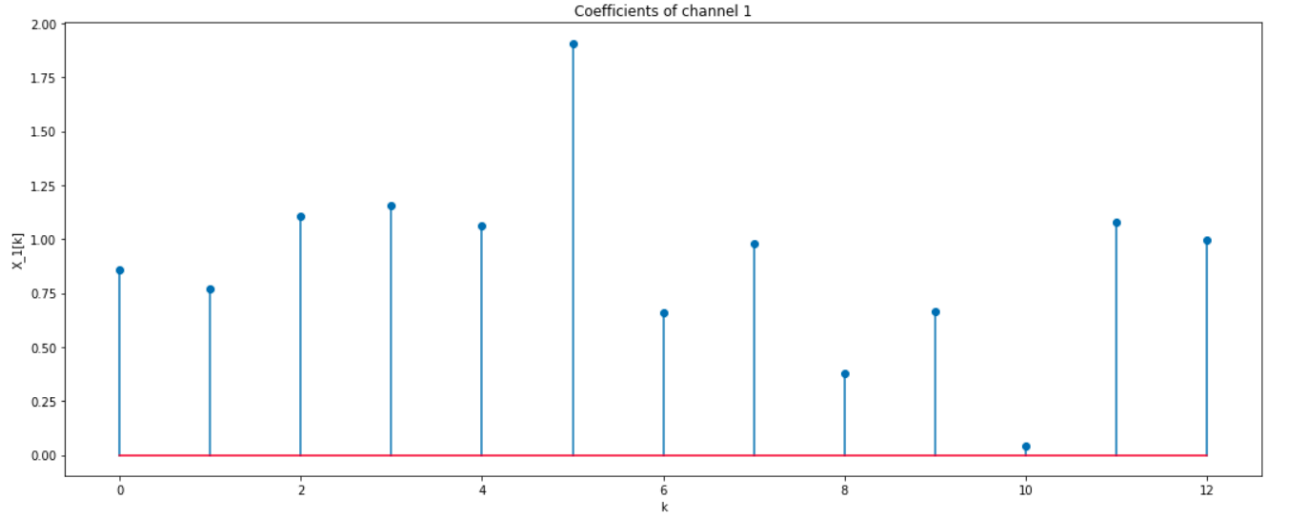

Channel output for channel 7
Re{ch7_out[n]}
n

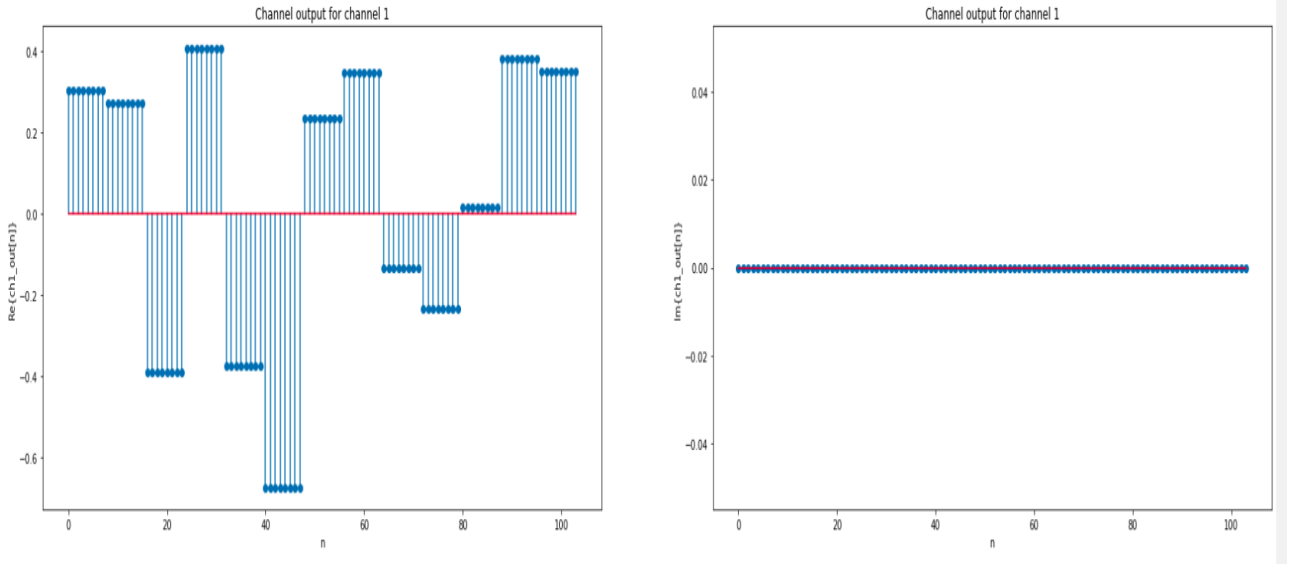

Channel output for channel 7
Im{ch7_out[n]}
n

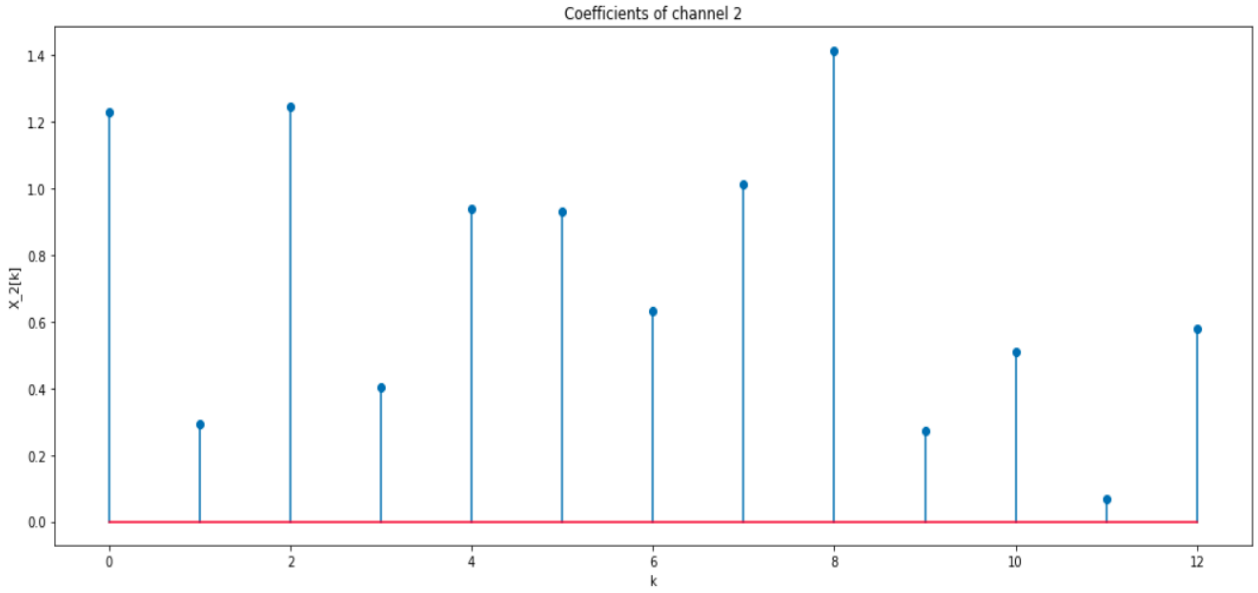

Coefficients of channel 8
X_8[k]
k

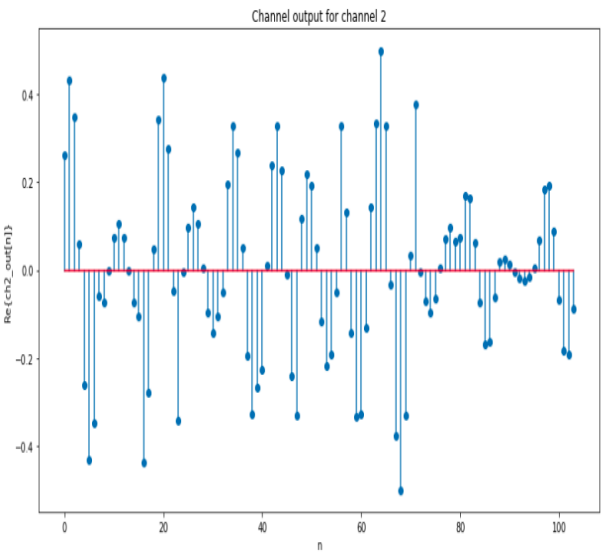

Channel output for channel 8
Re{ch8_out[n]}
n
Channel output for channel 8
Im{ch8_out[n]}
n

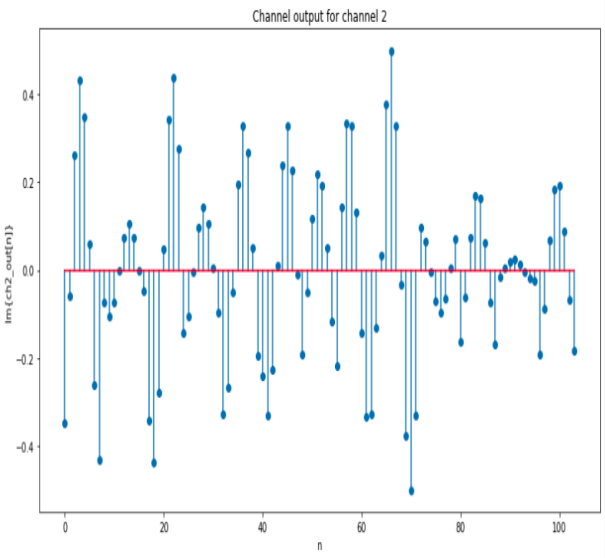
Part (b) : x2[n]

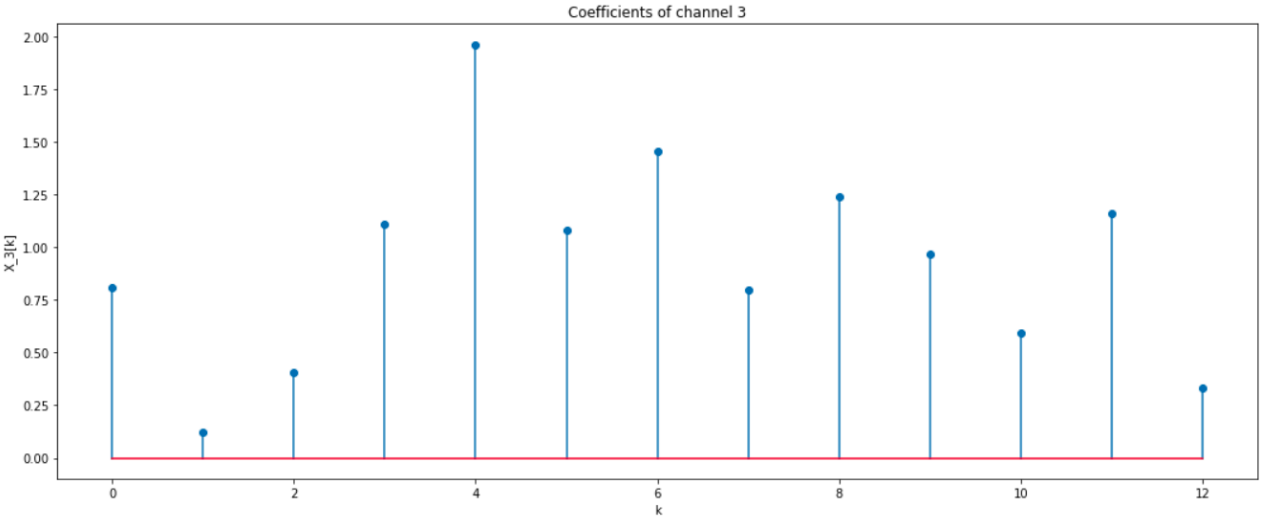

Coefficients of channel 1
X_1[k]
k

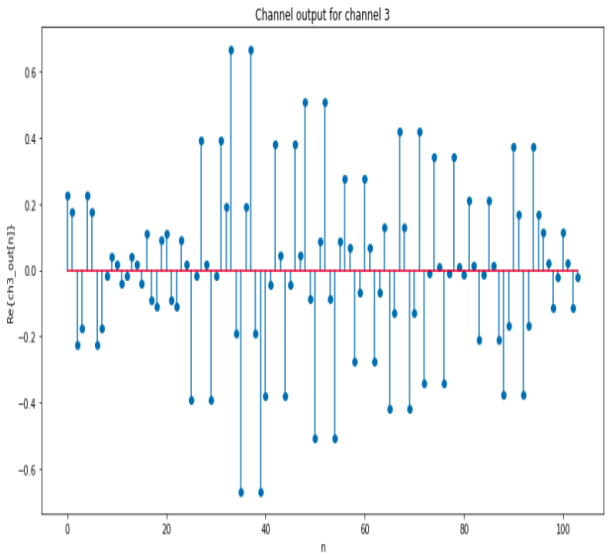

Channel output for channel 1
Re{ch1_out[n]}
n
Channel output for channel 1
Im{ch1_out[n]}
n

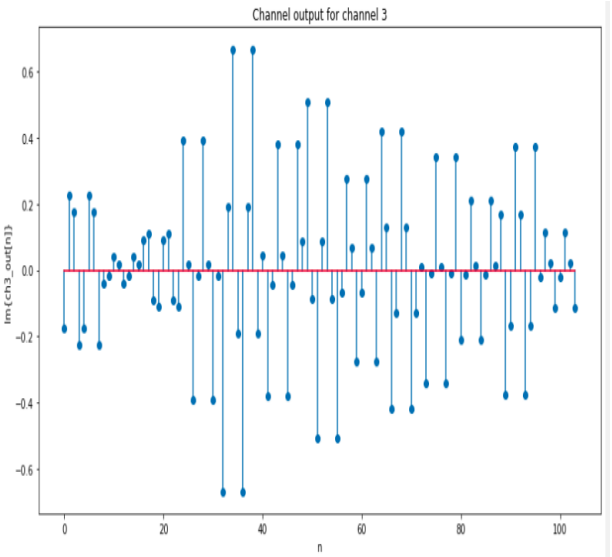

Coefficients of channel 2
X_2[k]
k

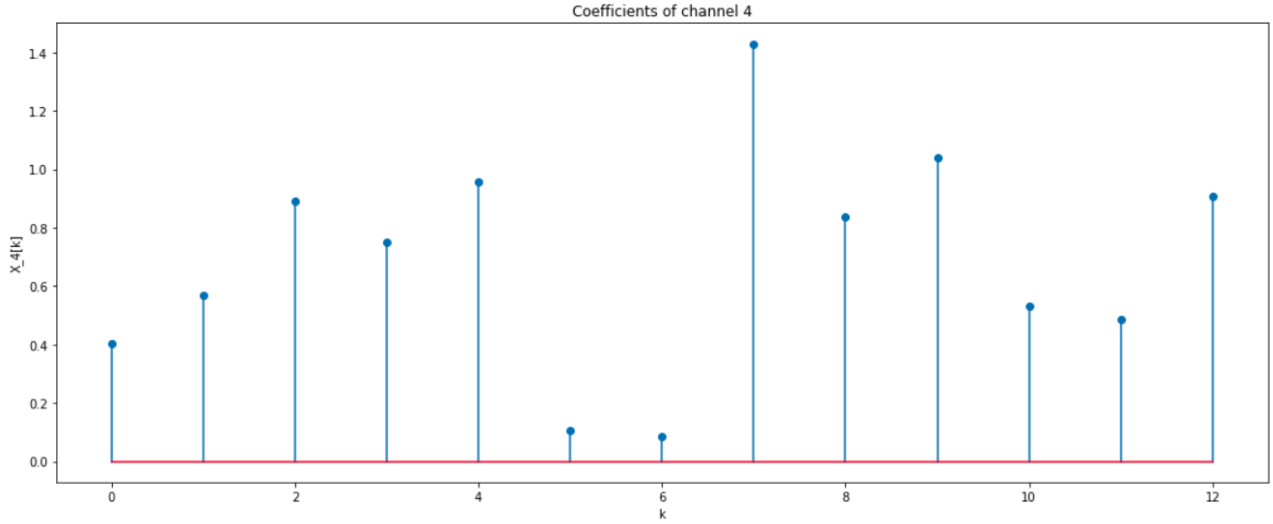

Channel output for channel 2
Re{ch2_out[n]}
n

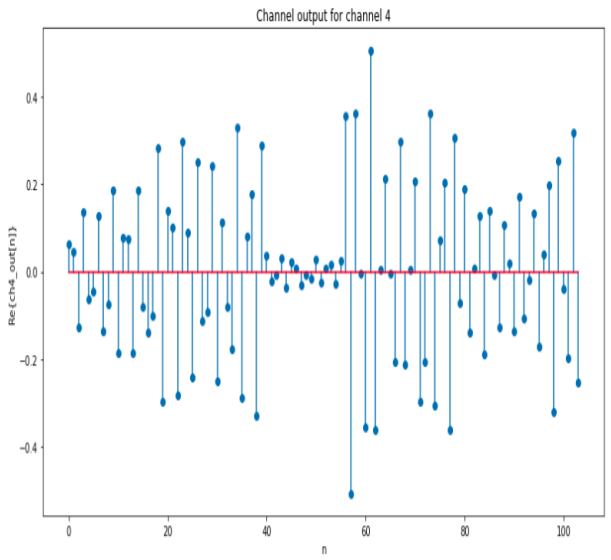

Channel output for channel 2
Im{ch2_out[n]}
n

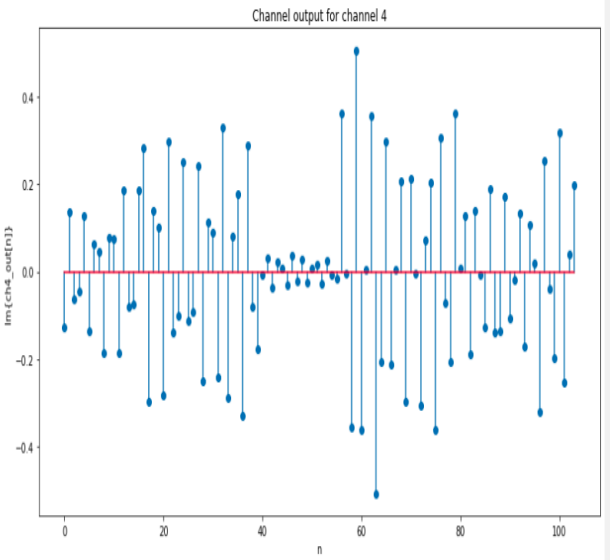

Coefficients of channel 3
X_3[k]
k

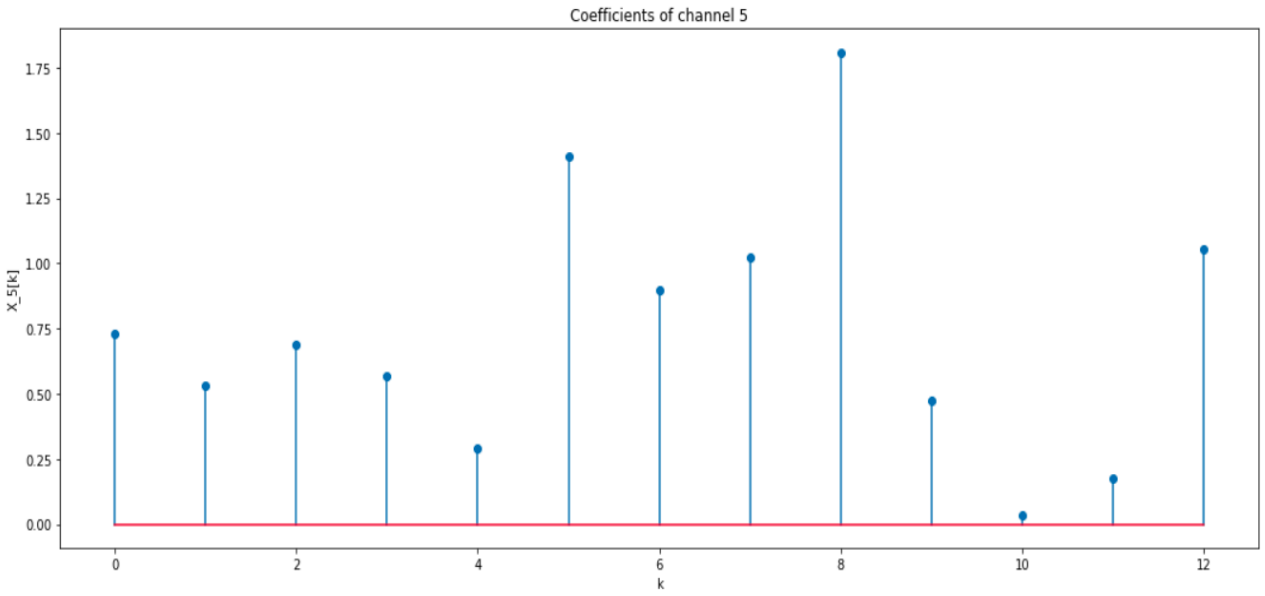

Channel output for channel 3
Re{ch3_out[n]}
n

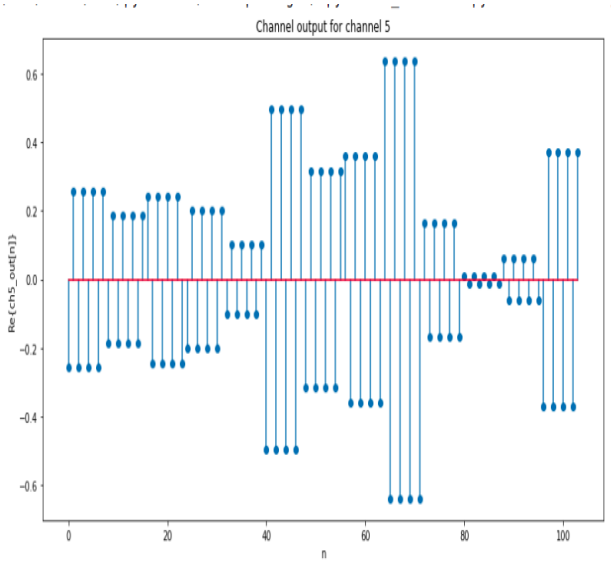

Channel output for channel 3
Im{ch3_out[n]}
n

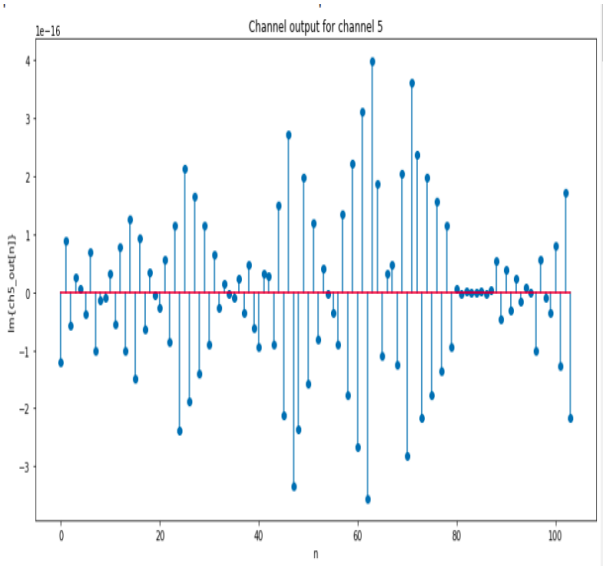

Coefficients of channel 4
X_4[k]
k

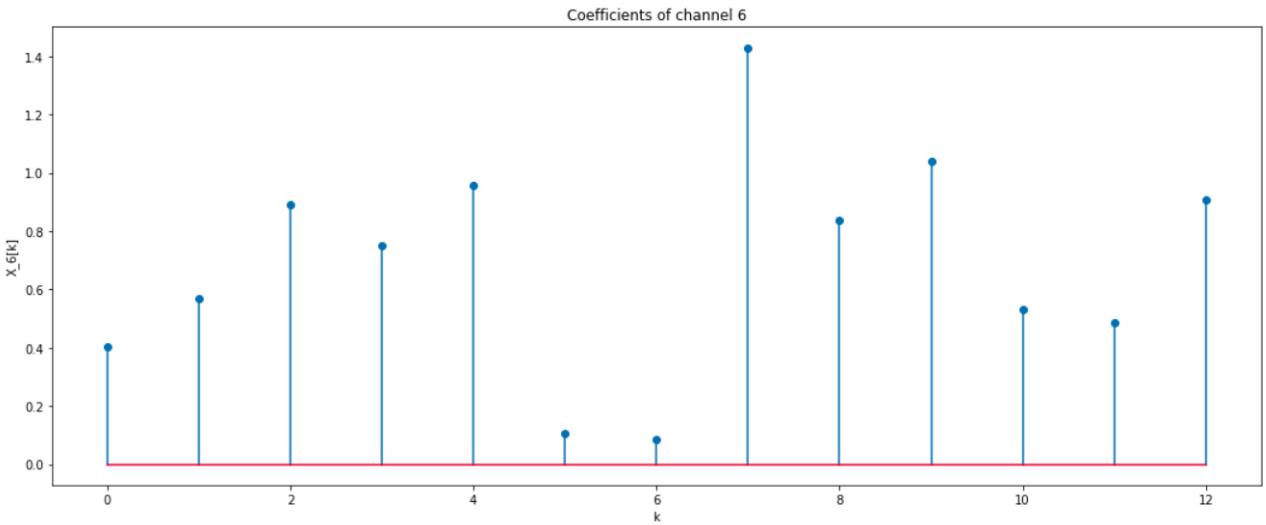

Channel output for channel 4
Re{ch4_out[n]}
n

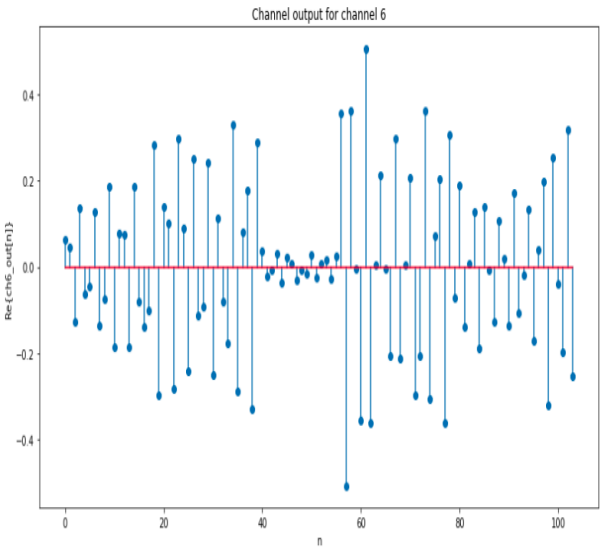

Channel output for channel 4
Im{ch4_out[n]}
n

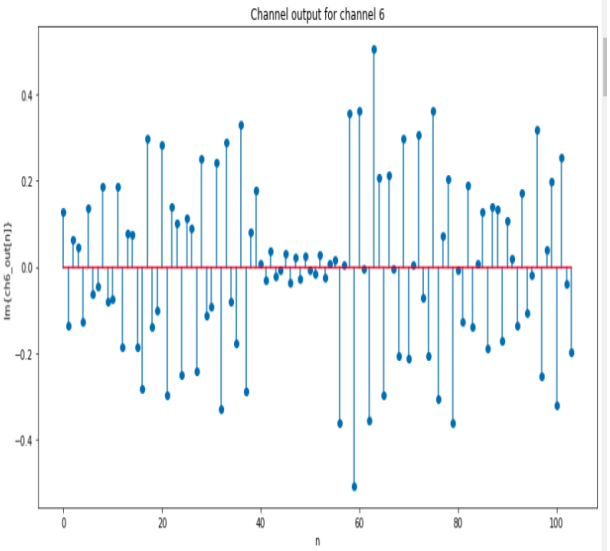

Coefficients of channel 5
X_5[k]
k

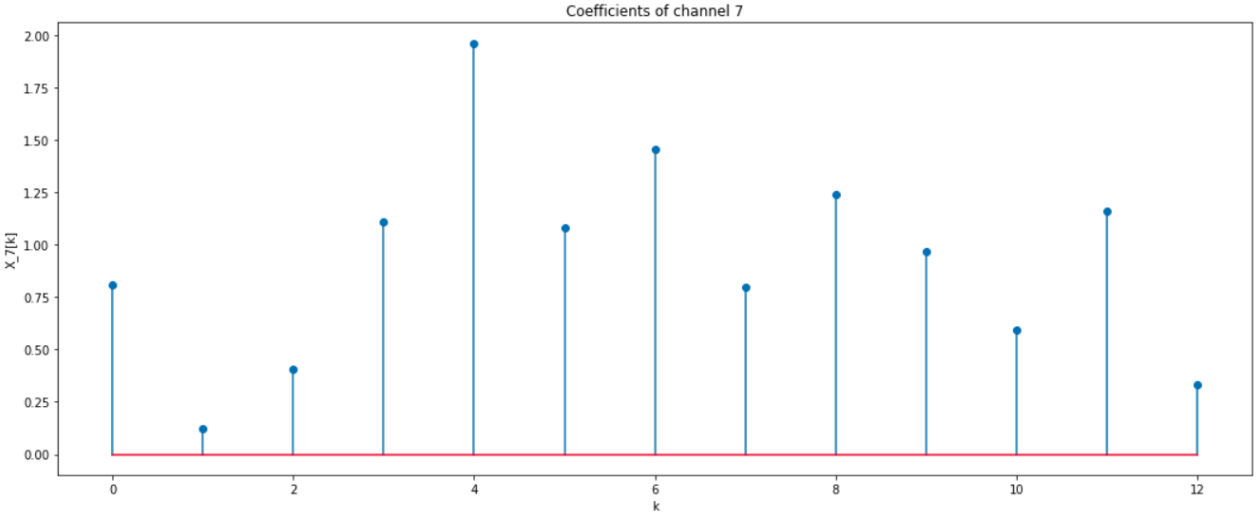

Channel output for channel 5
Re{ch5_out[n]}
n

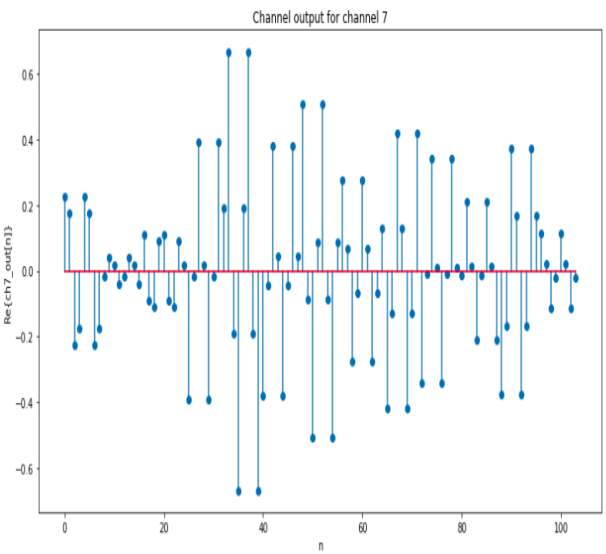

Channel output for channel 5
1e−16
Im{ch5_out[n]}
n

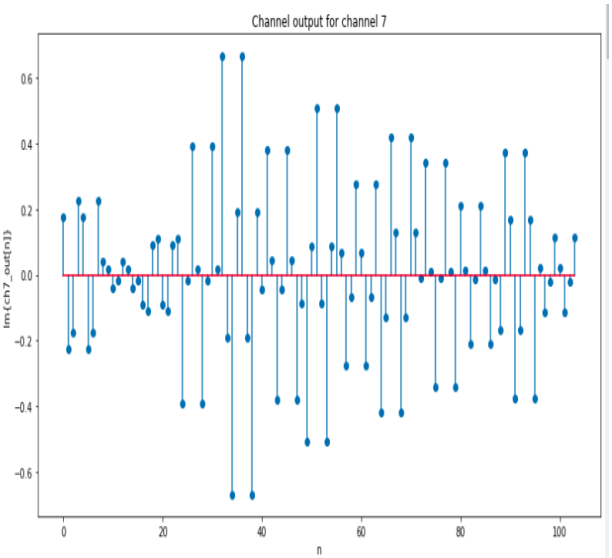

Coefficients of channel 6
X_6[k]
k

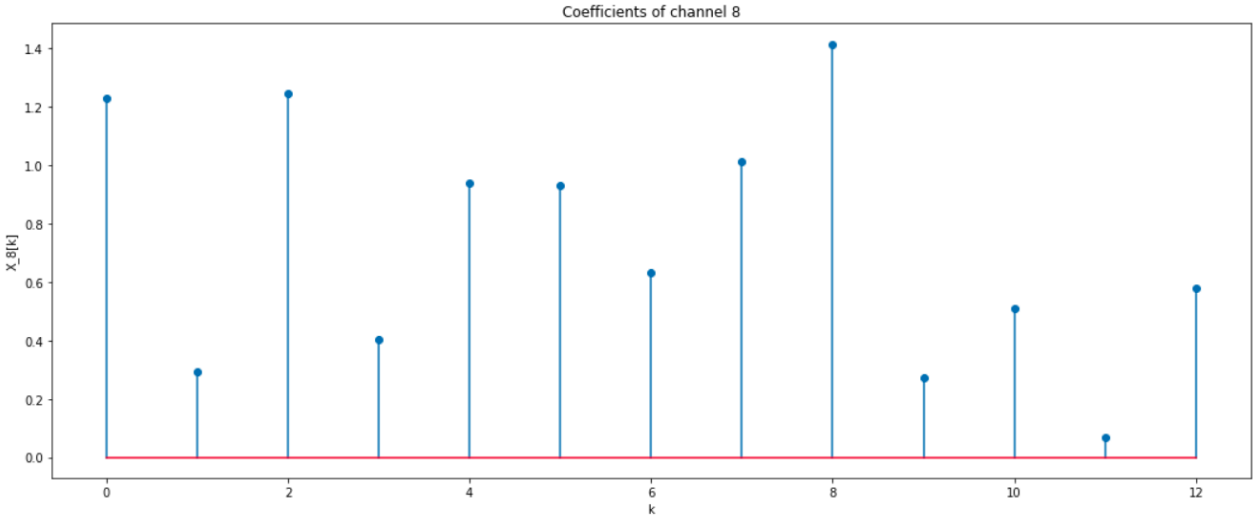

Channel output for channel 6
Re{ch6_out[n]}
n

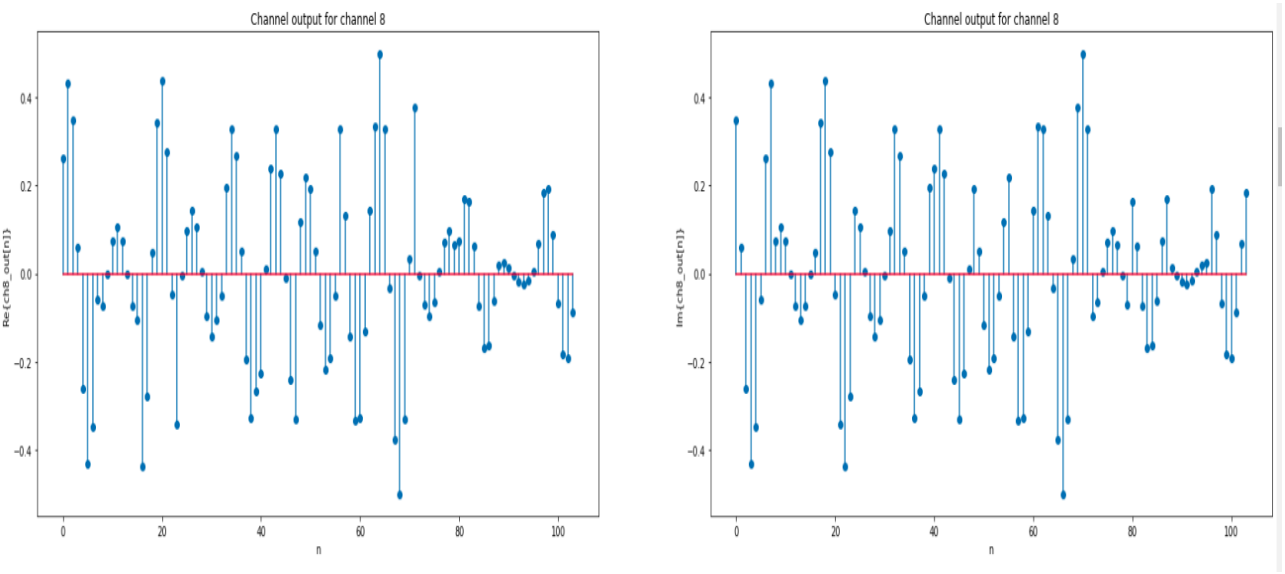

Channel output for channel 6
Im{ch6_out[n]}
n


Coefficients of channel 7
X_7[k]
k

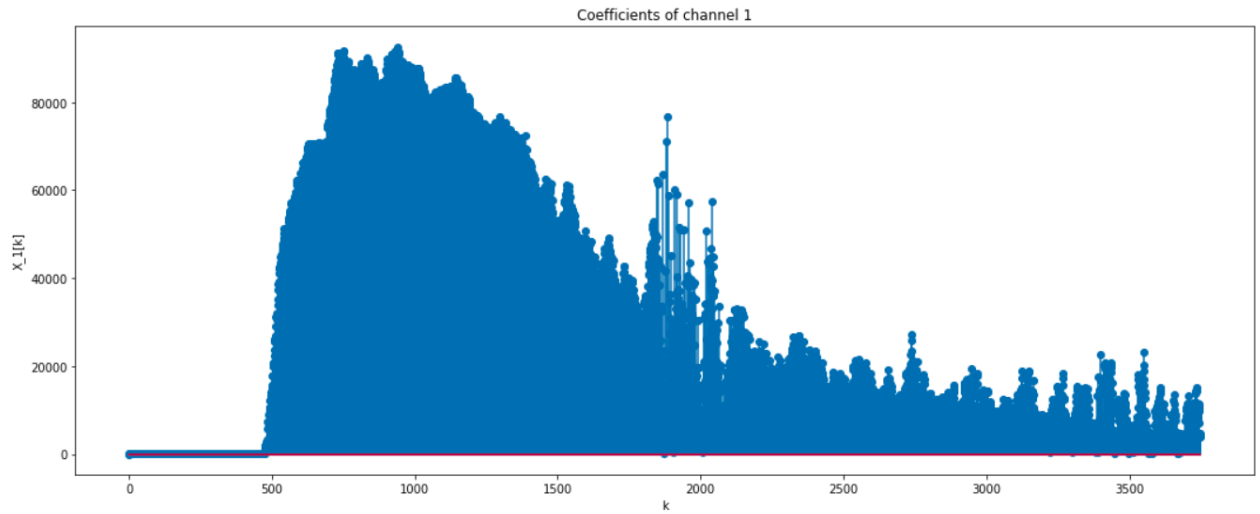

Channel output for channel 7
Re{ch7_out[n]}
n

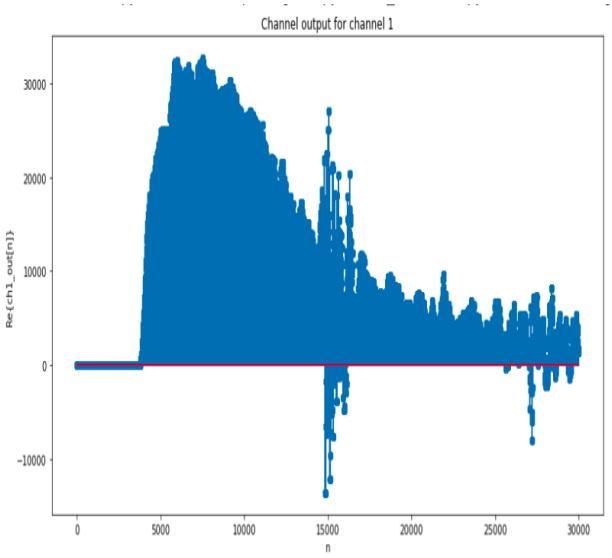

Channel output for channel 7
Im{ch7_out[n]}
n

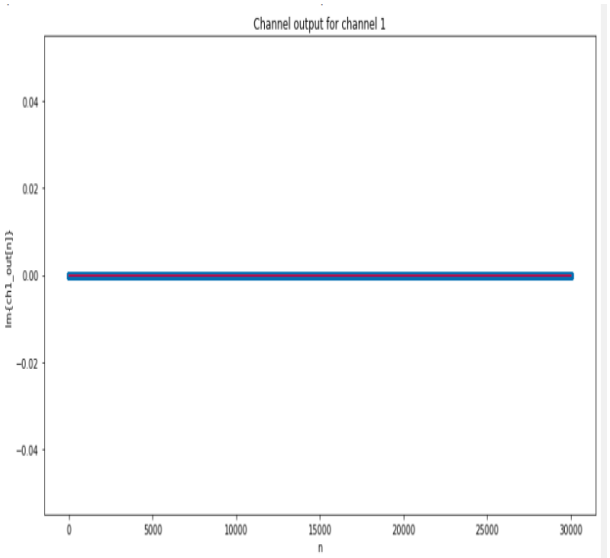

Coefficients of channel 8
X_8[k]
k

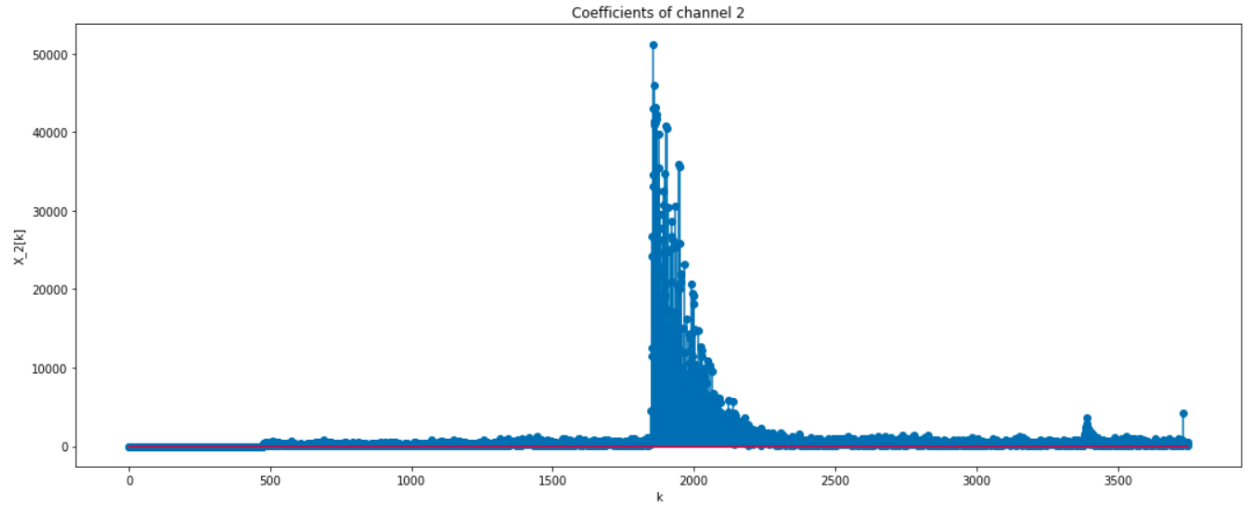

Channel output for channel 8
Re{ch8_out[n]}
n
Channel output for channel 8
Im{ch8_out[n]}
n

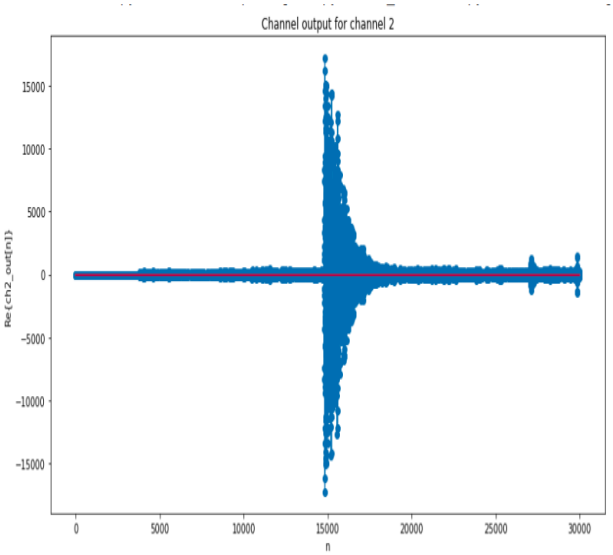
Part (c) : x3[n]

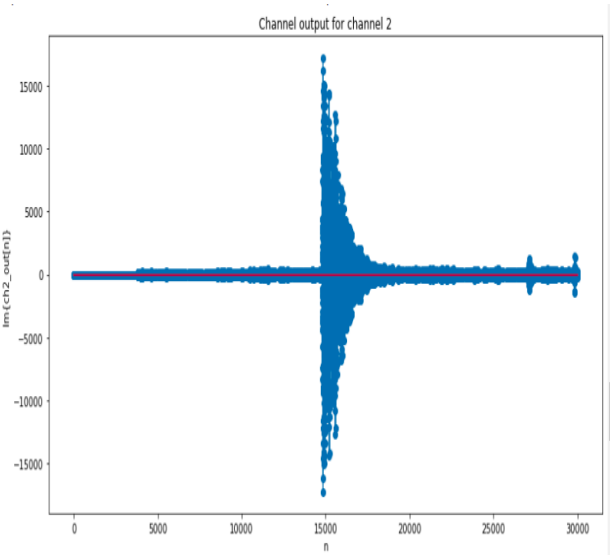

Coefficients of channel 1
X_1[k]
k

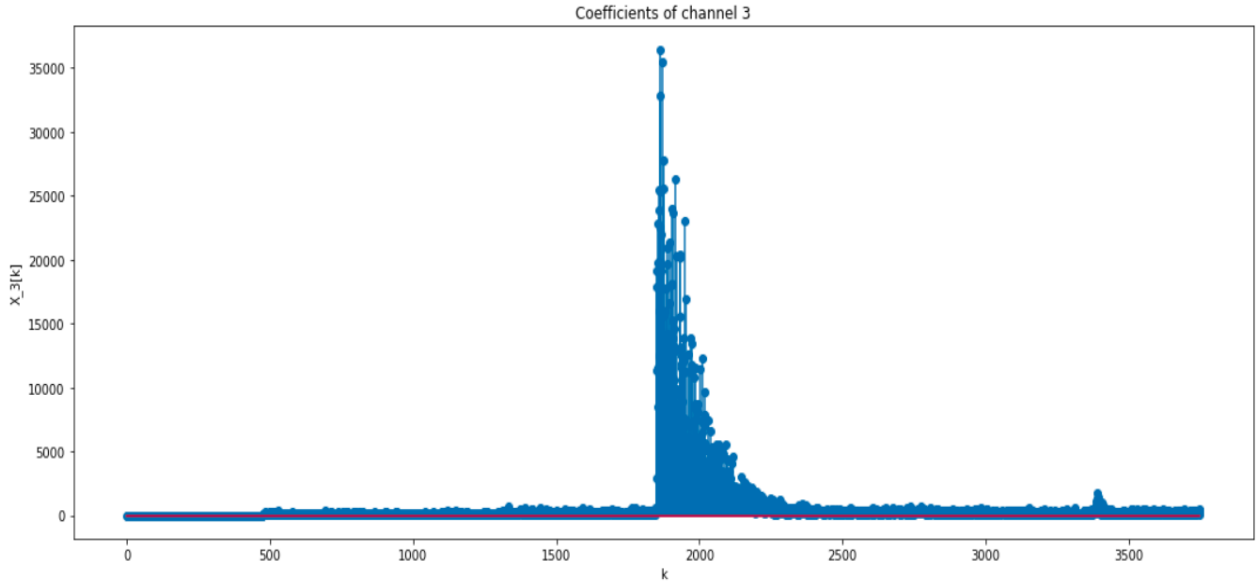

Channel output for channel 1
Re{ch1_out[n]}
n

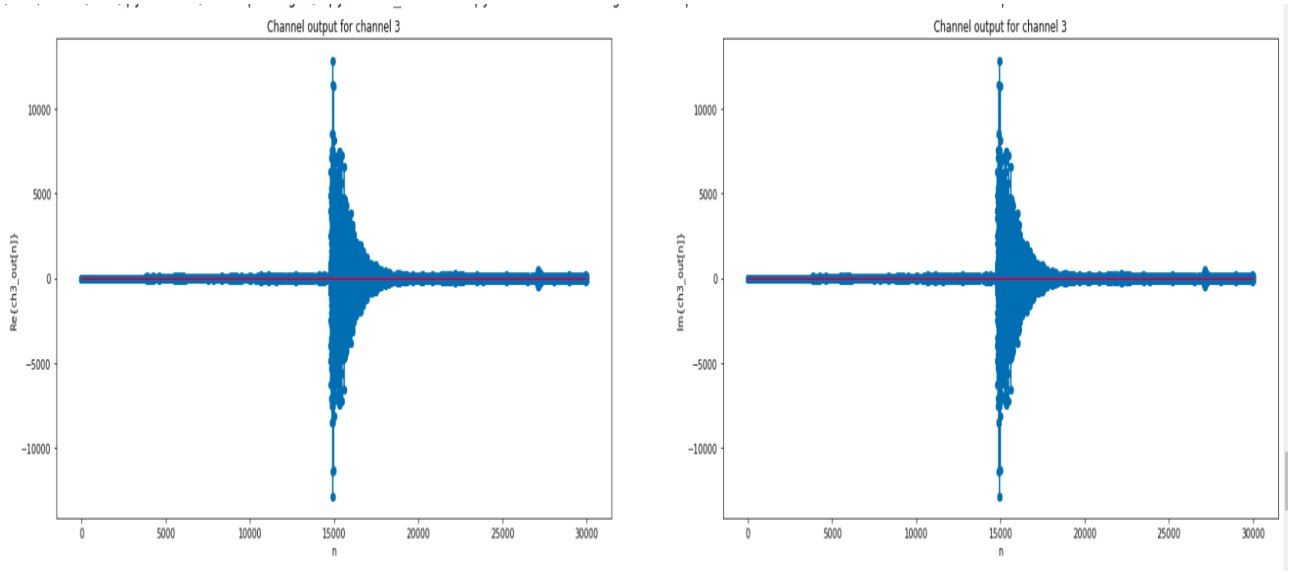

Channel output for channel 1
Im{ch1_out[n]}
n

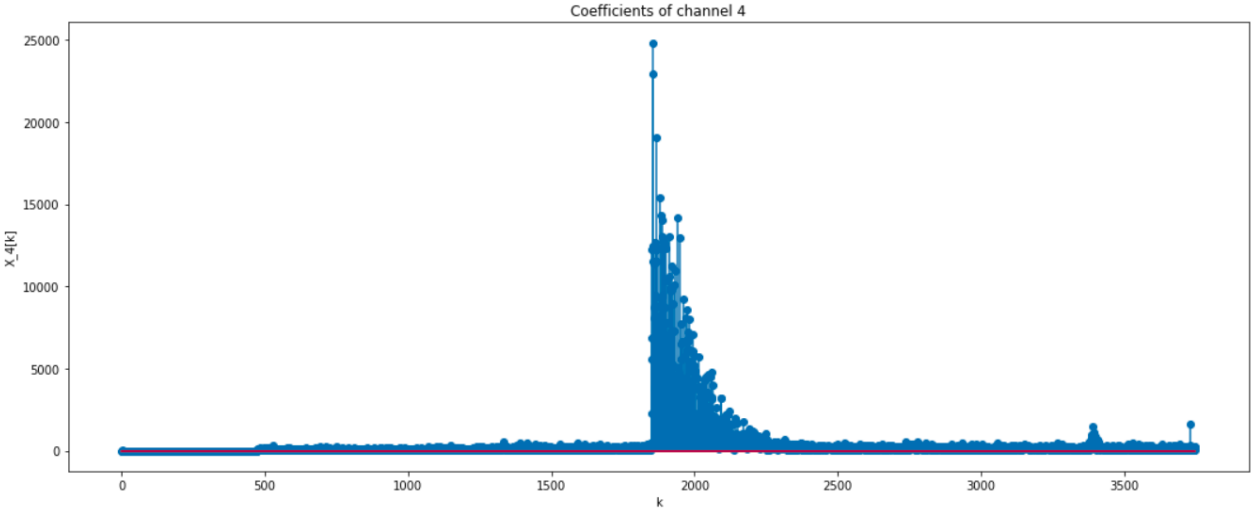

Coefficients of channel 2
X_2[k]
k

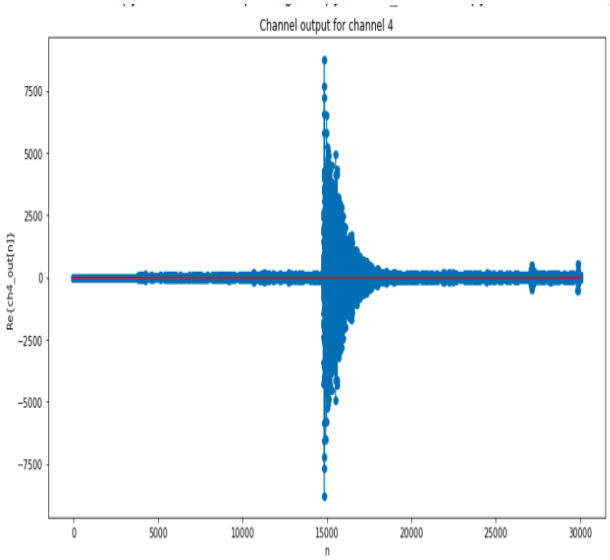

Channel output for channel 2
Re{ch2_out[n]}
n

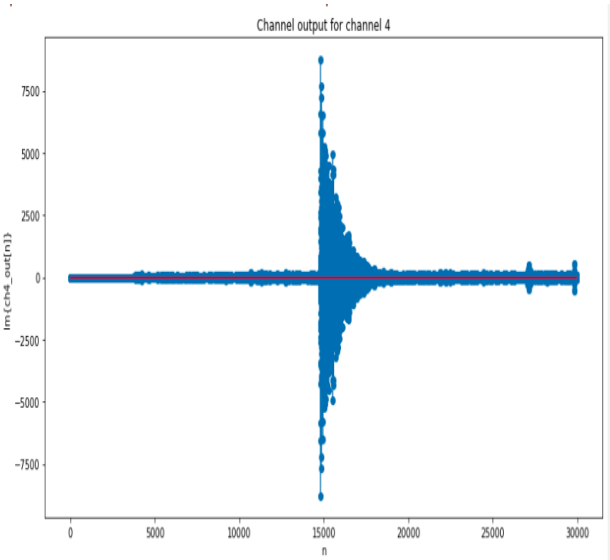

Channel output for channel 2
Im{ch2_out[n]}
n

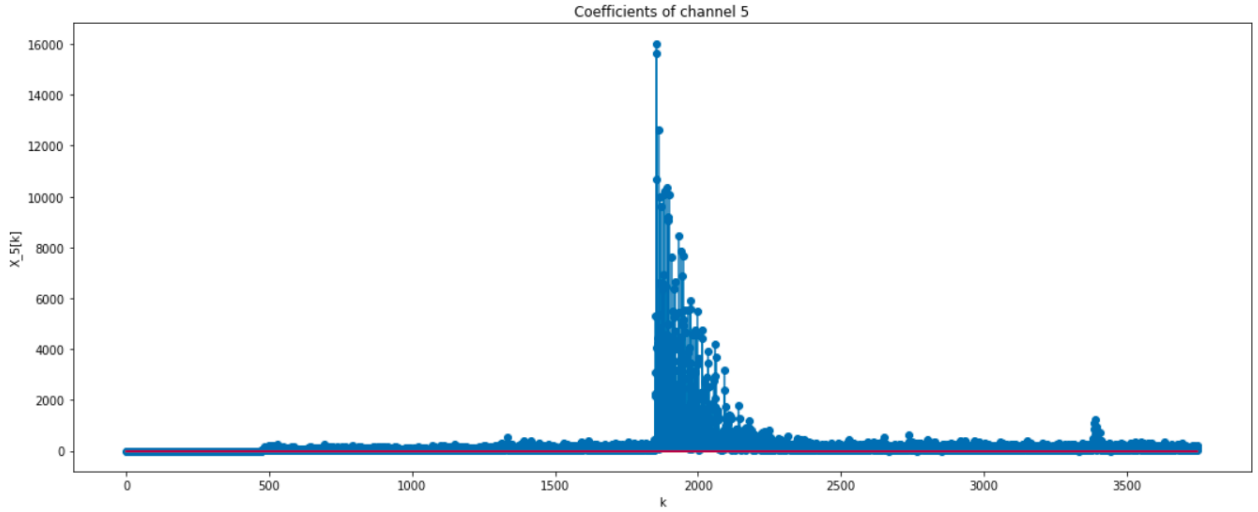

Coefficients of channel 3
X_3[k]
k

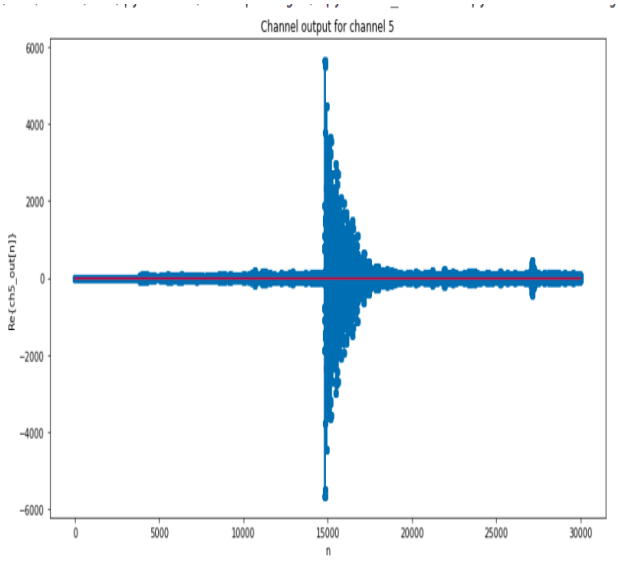

Channel output for channel 3
Re{ch3_out[n]}
n
Channel output for channel 3
Im{ch3_out[n]}
n

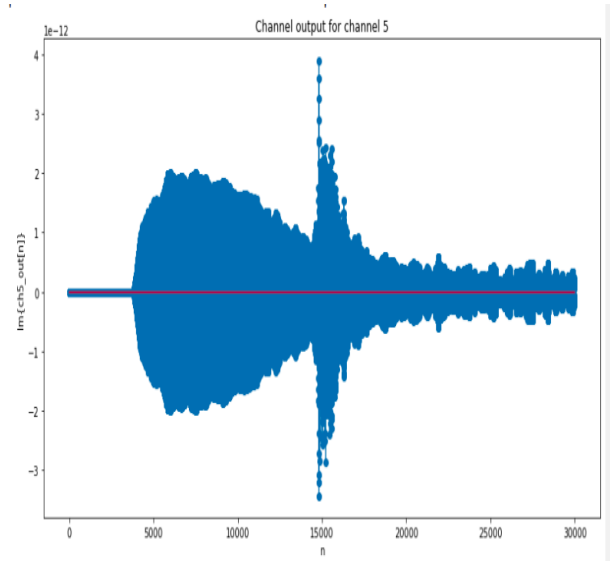

Coefficients of channel 4
X_4[k]
k

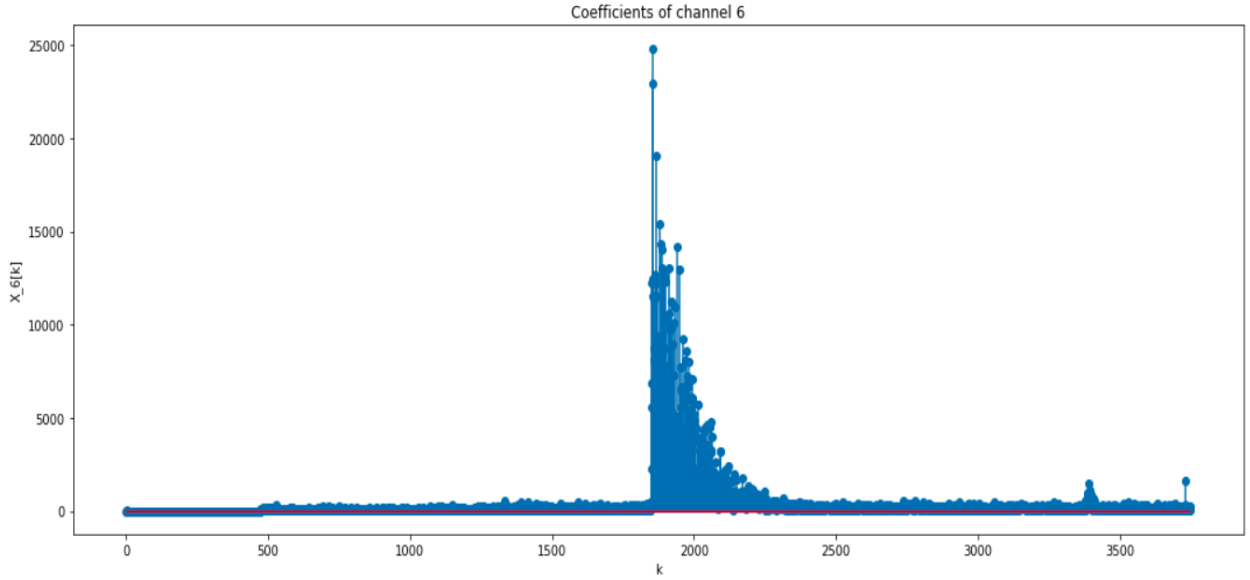

Channel output for channel 4
Re{ch4_out[n]}
n

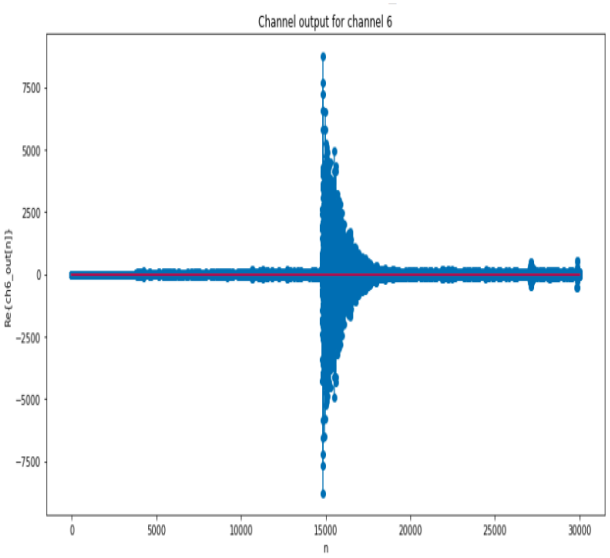

Channel output for channel 4
Im{ch4_out[n]}
n

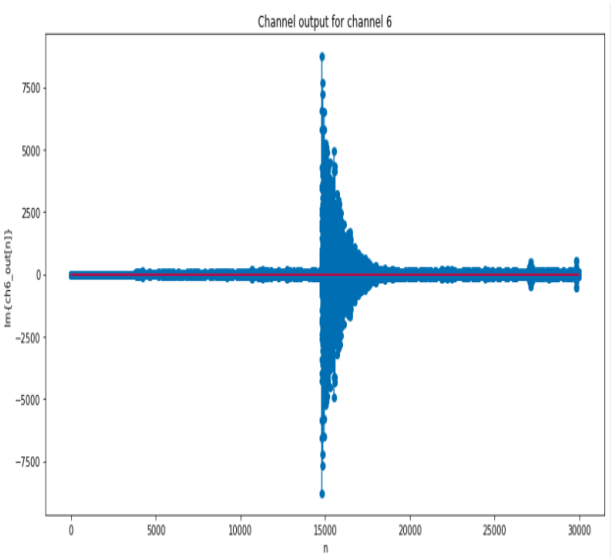

Coefficients of channel 5
X_5[k]
k

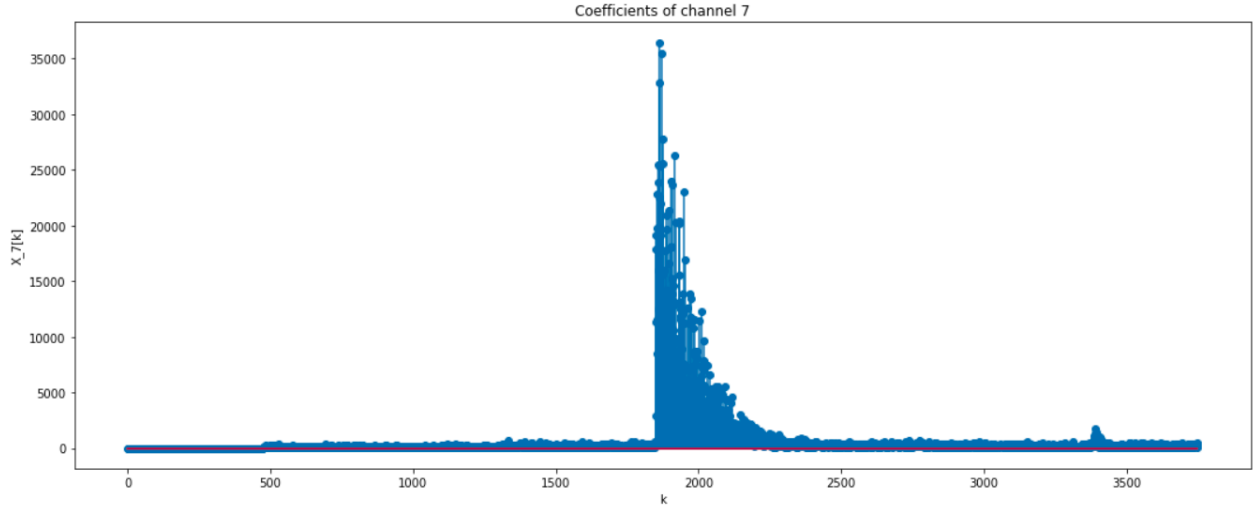

Channel output for channel 5
Re{ch5_out[n]}
n

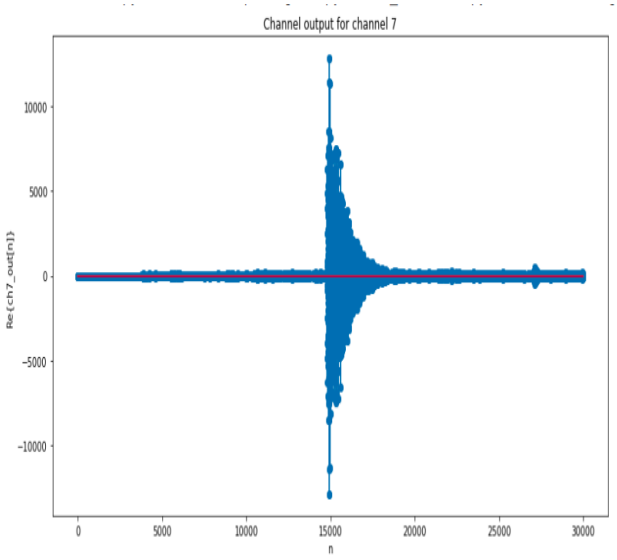

Channel output for channel 5
1e−12
Im{ch5_out[n]}
n

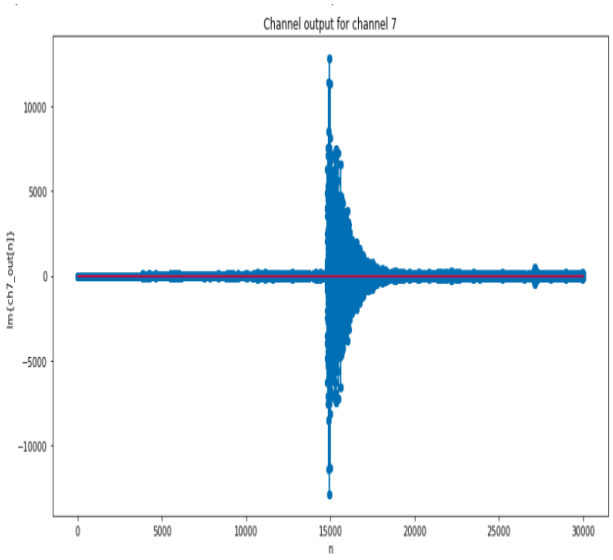

Coefficients of channel 6
X_6[k]
k

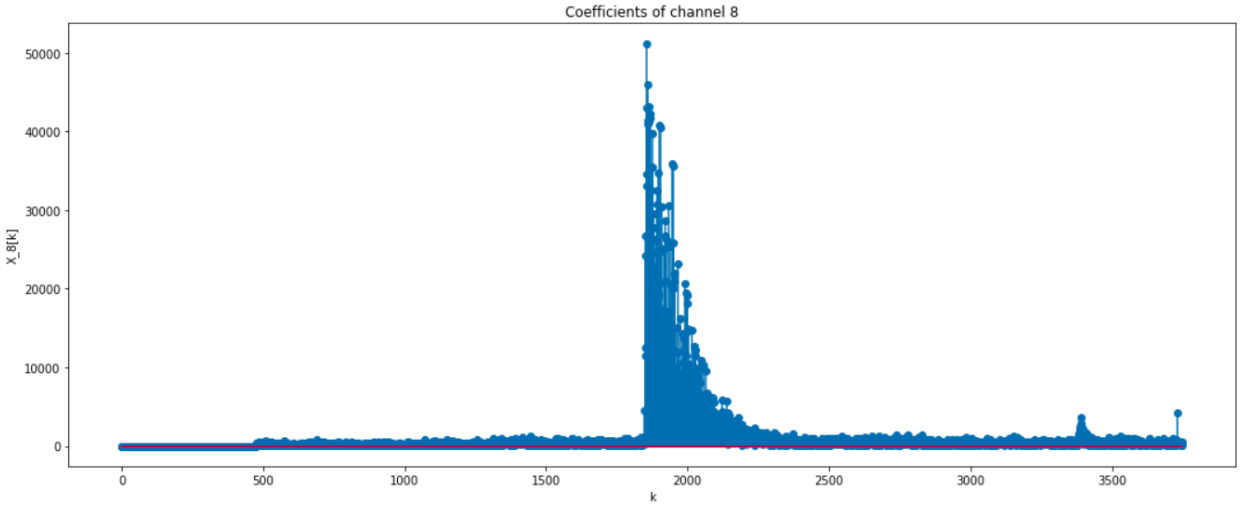

Channel output for channel 6
Re{ch6_out[n]}
n

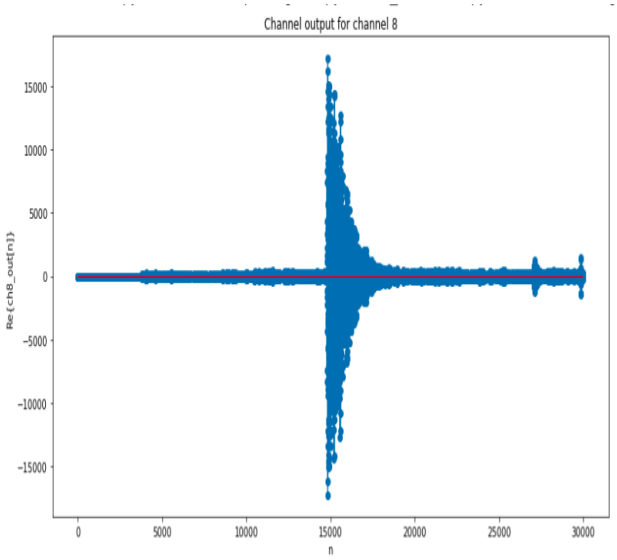

Channel output for channel 6
Im{ch6_out[n]}
n

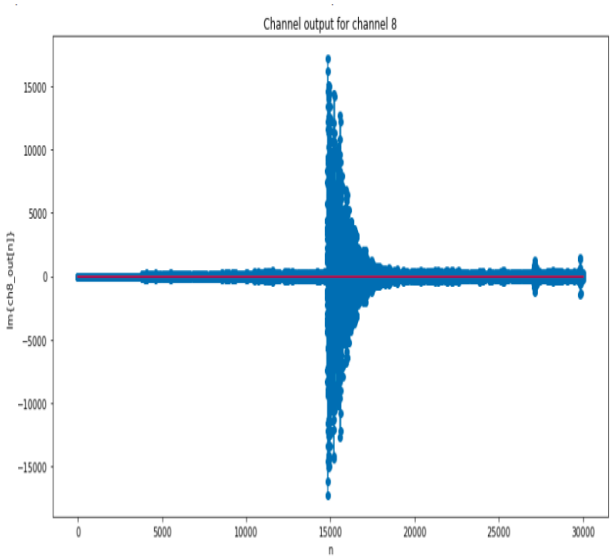

Coefficients of channel 7
X_7[k]
k


Channel output for channel 7
Re{ch7_out[n]}
n


Channel output for channel 7
Im{ch7_out[n]}
n


Coefficients of channel 8
X_8[k]
k


Channel output for channel 8
Re{ch8_out[n]}
n


Channel output for channel 8
Im{ch8_out[n]}
n

Inferences:

- In our eight channel filter bank, we have eight basis signals each of which correspond to a different time period and frequency. Each $X_l[k]$ tells us how much of the basis signal $h_l[n]$ is present in eight successive samples of our original signal starting from time index 8k.

- As we have seen from question 1, the signal x1[n] consists of two different sinusoids with the first 32 samples being a lower frequency sinusoid while the second 32 samples has a higher frequency sinusoid. Thus, if we look at any of the individual channel outputs, either the first four or the last four coefficients are found to dominate depending on whether the channel in consideration is high pass or low pass. It is observed that for channels 1, 2 and 8, the first four coefficients dominate telling us that the lower frequency sinusoid has greater weights of these complex exponentials whereas for channels 3, 4, 5, 6 and 7, the second four coefficients are found to dominate which tells us that the signal has a higher proportion of these frequencies for samples 32 to 63. Thus, the coefficients tell us what kind of frequency is present in the signal at different time intervals. Looking at the channel outputs, we can observe that channels 1, 2 and 8 being low pass channels pass the lower frequency sinusoid i.e. the first 32 samples whereas the remaining channels pass the higher frequency sinusoids. The channels 4, 5 and 6 almost completely suppress the lower frequency component whereas the remaining samples only partially suppress the lower frequency sinusoid. The channel outputs have a real and imaginary part which correspond to a cosine or a sine of a that particular frequency which is weighted by the channel coefficients for 8 samples each. The sinusoids are phase shifted since the coefficients may be complex.

- The random Gaussian signal that we have generated in this case has a channel response similar to question 2 where high pass channels have dominate magnitudes towards the middle of the time axis while low pass channels have dominant coefficients towards the ends. The channle outputs also reflect the same behaviour where high pass channels attenuate the signal values near the ends while low pass channels attenuate the signal values towards the middle.

- Looking at the coefficients of x3[n], most of the information about the signal is obtained from channel 1 itself. The remaining channels have very similar coefficients somewhere near the time index of n = 15000. This is expected since as we discussed earlier, near n = 15000 the signal behaves like an impulse which has a uniform frequency distribution and thus the frequency-selective channels will all have equal magnitudes at the different frequencies. The fact that most of the information comes from channel 1 which is a dc channel is justified by the fact that the DTFT of the signal that we observed in question 1 had a huge peak near the zero or dc frequency. The channel outputs are also very similar with channel 1 giving most of the original signal while the other channels give an approximate impulse function with varying magnitudes.