# Signal and System Homework № 2

Atahan Özer 09/05/2020

## Problem 1

i. (a) **Linearity**

$x_1(t)\Rightarrow ax_1(t)$ and $x_2(t)\Rightarrow bx_2(t)$ summed and fed into the system

$ax_1(t) + bx_2(t) \Rightarrow y = 5(ax_1(t) + bx_2(t))^2 + 2(ax_1(t) + bx_2(t)) + 4$

In order to check linearity ,this time two input first fed into system and then summed .

$x_1(t)\Rightarrow y_1 = 5(ax_1(t))^2 + 2ax_1(t) + 4$ and $x_2(t)\Rightarrow y_2 = 5(ax_2(t))^2 + 2ax_2(t) + 4$

Since $y_1 + y_2$ not equal to $y$ this system is not linear.

(b) **Time invariance**

First fed into the system then shifted by $t_0$.

$x_1(t)\Rightarrow y_1(t) = 5(ax_1(t))^2 + 2ax_1(t) + 4 \Rightarrow y(t\text{-}t_0) = 5(ax_1(t - t_0))^2 + 2ax_1(t_0) + 4$

First shifted by $t_0$ then fed into system

$x_1(t)\Rightarrow x_1(t - t_0) \Rightarrow y_1(t - t_0) = 5(ax_1(t - t_0))^2 + 2ax_1(t_0) + 4$

Since $y = y_1$ this system is time invariant.

(c) **Causality**

This system does not require any future inputs therefore it is casual

(d) **Stability**

Since this system is not a LTI stability can not be checked by using impulse response ; however , the function of the system does not have any discontinuity or jump therefore for every bounded input , system will produce an bounded output.This satisfies the general stability condition.

ii. (a) **Linearity**

For this system it will be sufficient to investigate only when t>0 because system is zero otherwise. $x_1(t)\Rightarrow ax_1(t)$ and $x_2(t)\Rightarrow bx_2(t)$ summed and fed into the system

$ax_1(t) + bx_2(t) \Rightarrow y = ax_1(t) + bx_2(t) - ax_1(t - 4) - bx_2(t - 4)$ In order to check linearity ,this time two input first fed into system and then summed .

$x_1[n]\Rightarrow y_1 = x_1(t) - x_1(t - 4)$ and $x_2[n]\Rightarrow y_2 = x_2(t) - x_2(t - 4)$

$ay_1 + b_y2 = ax_1(t) + bx_2(t) - ax_1(t - 4) - bx_2(t - 4)$

Since $y_1 + y_2 = y$ this system is linear .

(b) **Time invariance**

First fed into the system then shifted by $t_0$.

$x_1(t) \Rightarrow$ y(t)= x(t)-x(t-4) $\Rightarrow$ y(t-$t_0$)= $x(t-t_0) - x(t-4-t_0)$

First shifted by $t_0$ then fed into system

$x_1(t) \Rightarrow$ x$_1(t-t_0) \Rightarrow$ y$_1(t-t_0) = x(t-t_0) - x(t-4-t_0)$

Since $y = y_1$ this system is time invariant.

(c) **Causality**

This system does not require any future inputs therefore it is casual.

(d) **Stability**

Since this system is LTI impulse response can be used for stability check. Impulse response of the sytem is given below.

$$h(t) = \begin{cases} 0, & t < 0 \\ \delta(t) - \delta(t-4), & t \geq 0 \end{cases}$$

$\int_{-\infty}^{\infty} |h(\tau)| d\tau$ = 0 + $\int_0^{\infty} \delta(t) - \delta(t-4) d\tau$ = $u(t) - u(t-4)$

Since the equation above less than infinity this system is stable .

iii (a) **Linearity**

$x_1[n] \Rightarrow$ ax$_1[n]$ and $x_2[n] \Rightarrow$ bx$_2[n]$ summed and fed into the system.

$ax_1[n] + bx_2[n] \Rightarrow$ y=$\sum_{n+2}^{m=n-2} ax_1[m] + bx_2[m] - 2|(ax_1[n] + bx_2[n])|$

In order to check linearity ,this time two input first fed into system and then summed.

$x_1[n] \Rightarrow$ y$_1 = \sum_{n+2}^{m=n-2} x_1[m] - 2|x_1[n]|$ and $x_2[n] \Rightarrow$ y$_2 = \sum_{n+2}^{m=n-2} x_2[m] - 2|x_2[n]|$

$ay_1 + by_2$ = $\sum_{n+2}^{m=n-2} ax_1[m] - 2a|x_1[n]| + \sum_{n+2}^{m=n-2} bx_2[m] - 2b|x_2[n]|$

Since $y_1 + y_2$ not equal to $y$ this system is not linear .

(b) **Time invariance**

First fed into the system then shifted by $n_0$.

$x[n] \Rightarrow$ y(t)= $\sum_{n+2}^{m=n-2} x[m] - 2|x[n]| \Rightarrow$ y(t-$t_0$)= $\sum_{n+2}^{m=n-2} x[m-n_0] - 2|x[n-n_0]|$

First shifted by $t_0$ then fed into system

$x(t) \Rightarrow$ x(t-$t_0$) $\Rightarrow$ y$_1(t-t_0) = \sum_{n+2}^{m=n-2} x[m-n_0] - 2|x[n-n_0]|$

Since $y = y_1$ this system is time invariant.

(c) **Causality**

In order to calculate y[n] system requires x[n+2] ; therefore ,this system is not casual .

(d) **Stability** If system is fed with a finite input, the output will be finite therefore system is stable.

iv (a) **Linearity and Time variance**

This system can be represented as an IIR filter.IIR filters are LTI whenever initial rest condition is satisfied.

$y[n] = \sum_{l=1}^{4} a_l y[n-l] + \sum_{k=0}^{2} b_k x[n-k]$

(b) **Casuality**

With the initial rest condition system does not require any future inputs ,therefore it is causal

## Problem 2

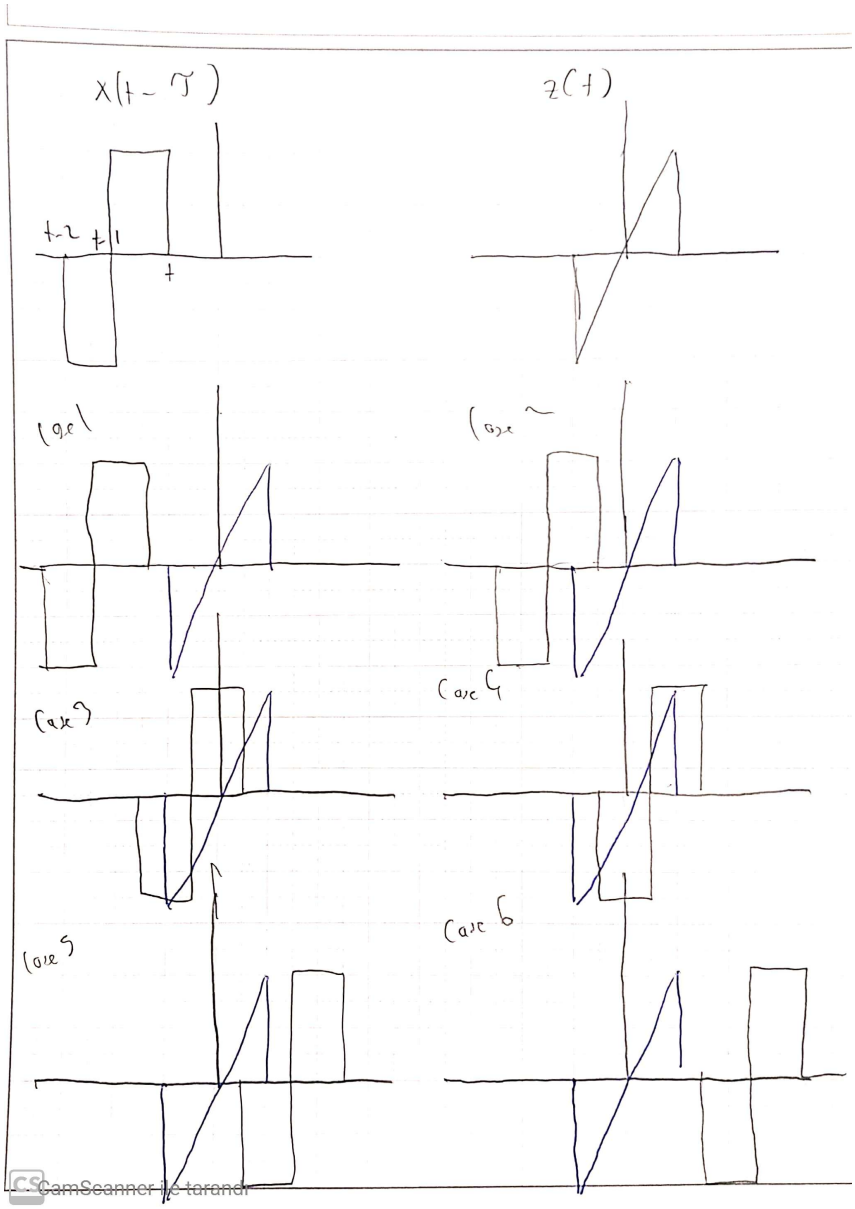


Figure 1: i

(a) i In order to calculate the result of the convolution 6 case should be computed.

-First case

When t<-1 , the output is y = 0 since there is no overlap.

-Second case

When -1<t<0 ,the output will be

y(t)= $\int_{-1}^{t} 1(\tau) = \frac{t^2-1}{2}$

-Third case

When 0<t<1 ,the output will be

y(t)= $\int_{t-1}^{t} \tau + \int_{-1}^{t-1} -\tau = \frac{-t^2+4t-1}{2}$

-Fourth case

When 1<t<2 ,the output will be

y(t)= $\int_{t-1}^{1} \tau + \int_{t-2}^{t-1} -\tau = \frac{-t^2+3}{2}$

-Fifth case

When 2<t<3 ,the output will be

y(t)= $\int_{t-2}^{1} -\tau = \frac{t^2-4t+3}{2}$

-Sixth case

When 3<t ,the output will be y(t)=0 since there is no overlap.

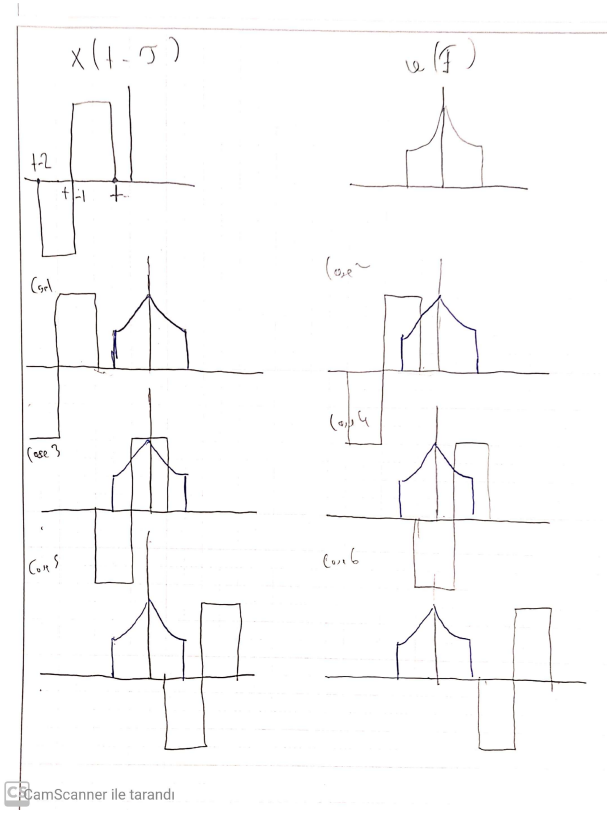



Figure 2: ii

ii In order to calculate the result of the convolution 6 case should be computed.

-First case

When t<-1 , the output is y = 0 since there is no overlap.

-Second case

When -1<t<0 ,the output will be

y(t)= $\int_{-1}^{t} e^{-2\tau} = \frac{e^{2t}-e^{-2}}{2}$

-Third case

When 0<t<1 ,the output will be

y(t)= $\int_{0}^{t} e^{-2\tau} + \int_{t-1}^{0} e^{-2\tau} + \int_{-1}^{t-1} -e^{-2\tau} = \frac{-2e^{2t-2}+e^{-2}+2-e^{-2t}}{2}$

-Fourth case

When 1<t<2 ,the output will be

y(t)= $\int_{t-1}^{1} e^{-2\tau} + \int_{0}^{t-1} -e^{-2\tau} + \int_{t-2}^{0} -e^{2\tau} = \frac{2e^{-2t+4}-2e^{2}-1+e^{2t-2}}{2e^{2}}$

-Fifth case

When 2<t<3 ,the output will be

y(t)= $\int_{t-2}^{1} -e^{-2\tau} d\tau = \frac{-e^{-2t+4}+e^{-2}}{2}$

-Sixth case

When 3<t ,the output will be y(t)=0 since there is no overlap.

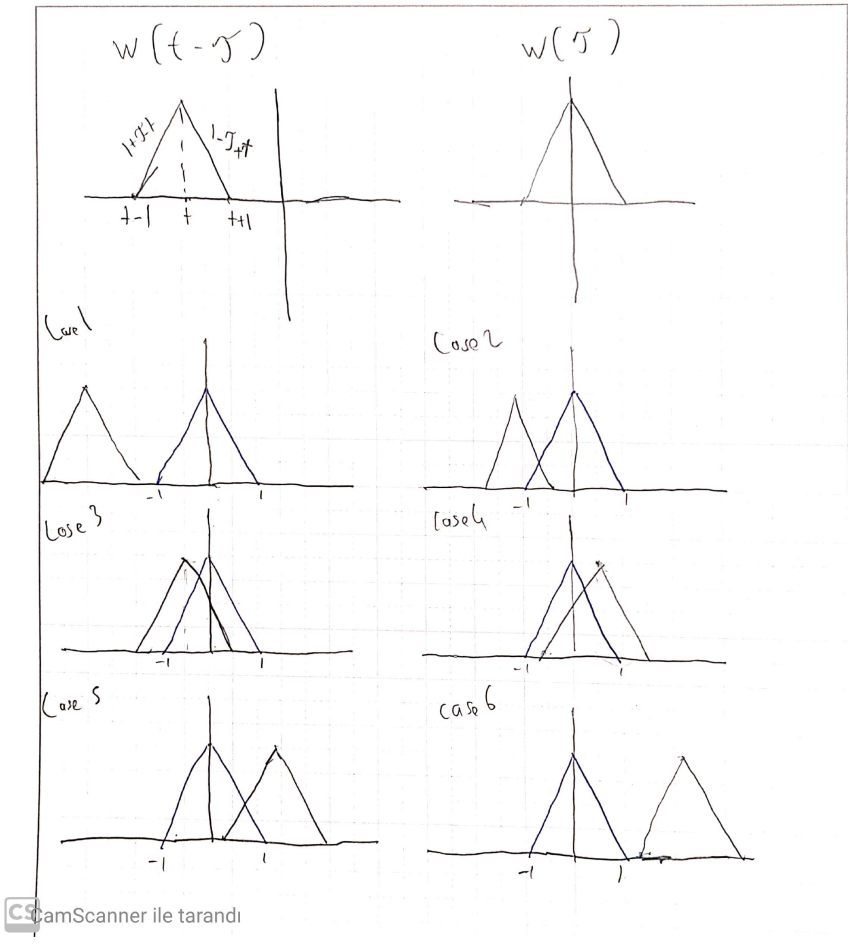


Figure 3: ii

iii In order to calculate the result of the convolution 6 case should be computed.

-First case

When t<-2 , the output is y = 0

-Second case

When -2<t<-1 ,the output will be

y(t)= $\int_{-1}^{t+1}(1+\tau)(1-\tau+t) = (1/6)(t+2)^3$

-Third case

When -1<t<0 ,the output will be

y(t)= $\int_{0}^{t+1}(1-\tau)(1-\tau+t) + \int_{t}^{0}(1+\tau)(1-\tau+t) + \int_{t}^{-1}(1+\tau)(1+\tau-t) = \frac{-3t^3-6t^2+4}{6}$

-Fourth case

When 0<t<1 ,the output will be

y(t)= $\int_{t}^{1}(1-\tau)(1-\tau+t) + \int_{0}^{t}(1-\tau)(1+\tau-t) + \int_{t-1}^{0}(1+\tau)(1+\tau-t) = \frac{3t^3-6t^2+4}{6}$

-Fifth case

When 1<t<2 ,the output will be

y(t)= $\int_{t-1}^{1}(1-\tau)(1+\tau-t) = \frac{-(t-2)^3}{6}$

-Sixth case

When 2<t ,the output will be y(t)=0 since there is no overlap.

(b) Graphics are plotted by hand ;however , due to long filter size computations made by computer using the table method.

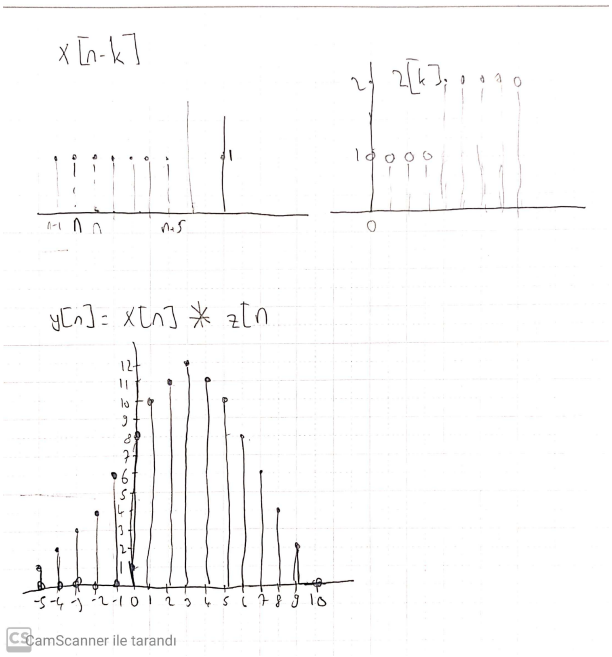


Figure 4: i

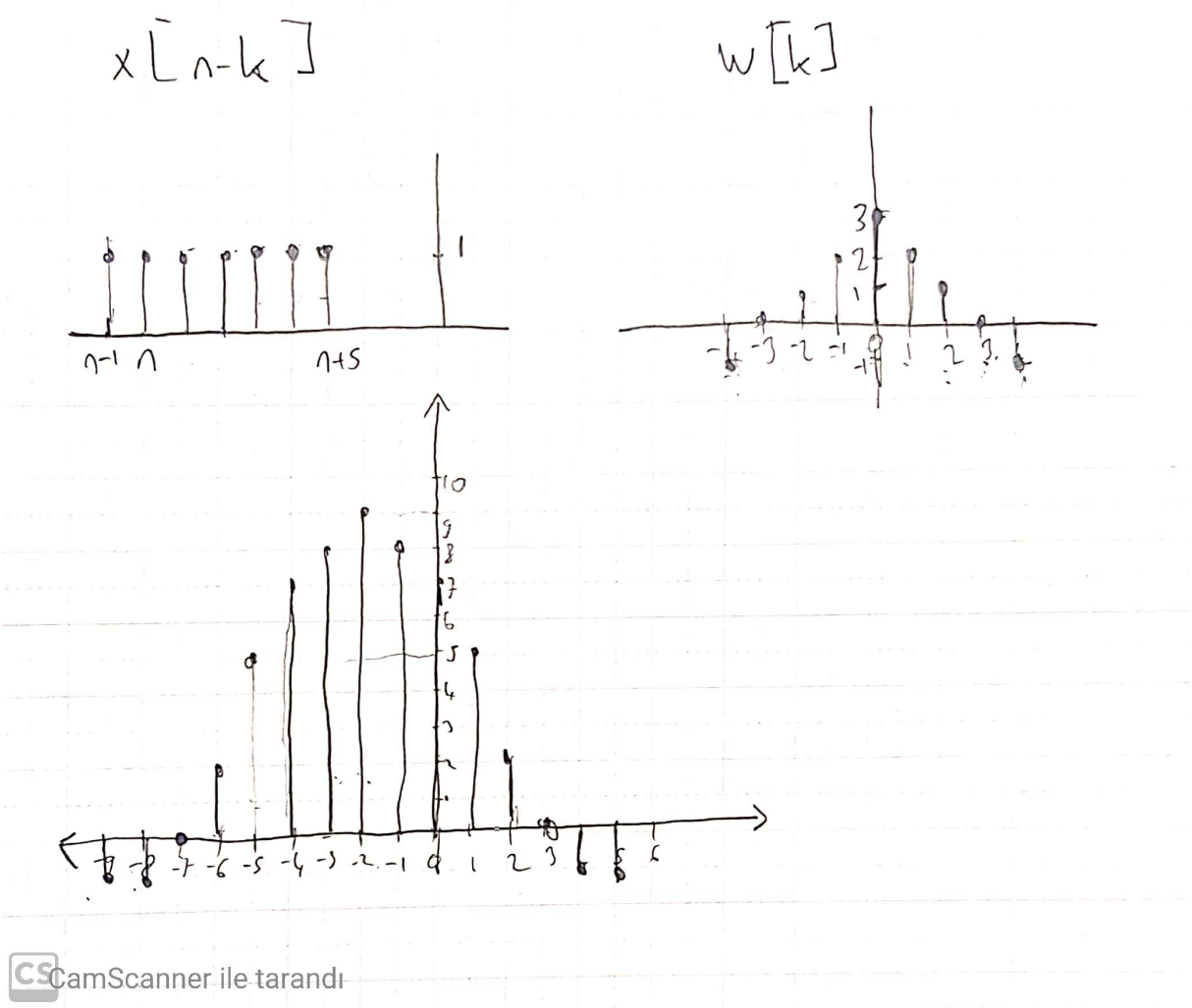




Figure 5: ii

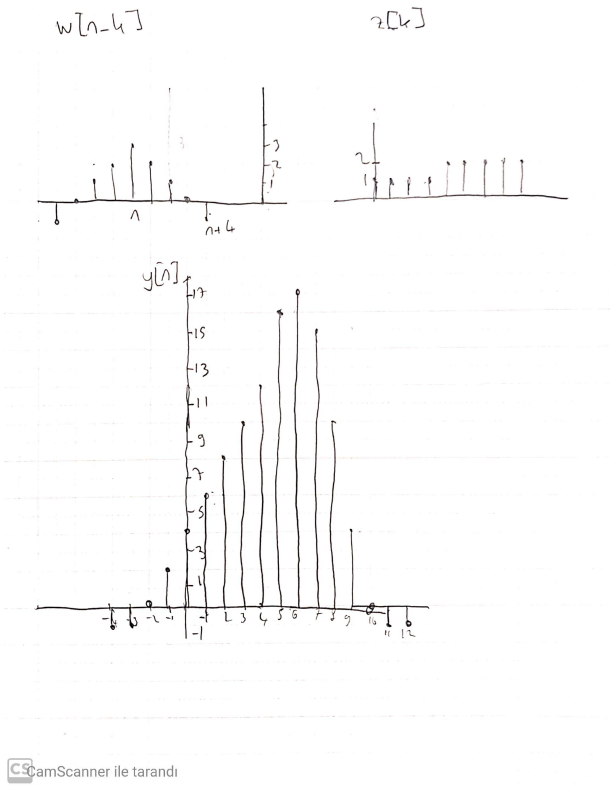


Figure 6: iii

## Problem 3

(i) $y[n] = ((u[n+10] - u[n]) - (u[n] + u[n-1])) * u[n-2]$
$y[n] = (\sum_{-10}^{-1} \delta(n+k) - \sum_{0}^{3} \delta(n-k)) * u[n-2]$
$y[n] = \sum_{-10}^{-1} u[n+k-2] - \sum_{0}^{3} u[n-k-2]$

(ii) I catched the mod 4 but I could not express it in terms of maths.

(iii) $y(t) = 2\delta(t+1) * u(t-1) + \delta(t-5) * u(t-1)$
$y(t) = 2u(t) + u(t-6)$

(iv) When t<0 u(t)*h(t) will be

y(t) = $\int_{-\infty}^{t} e^{2\tau} = \frac{e^{2t}}{2}$

When $t \geq 0$ u(t)*h(t) will be

y(t) = $\int_{-\infty}^{0} e^{2\tau} + \int_{0}^{t} e^{-3\tau} = 1/2 + \frac{1-e^{3t}}{3}$

## Problem 4

(a) Difference equation can be written by using h[n].

$y[n] = x[n] - x[n+2] + -7x[n-3]$

By using difference equation y[n] can be calculated for the given input.

$y[n] = x[n] - x[n+2] + -7x[n-3]$

$x[n] = \delta[n] - \delta[n-5] + 4\delta[n+3]$

$x[n+2] = \delta[n+2] - \delta[n-3] + 4\delta[n+5]$

$x[n-3] = \delta[n-3] - \delta[n-8] + 4\delta[n]$

$y[n] = -4\delta[n+5] + 4\delta[n+3] - \delta[n+2] - 27\delta[n] - 6\delta[n-3] - \delta[n-5] + 7\delta[n-8]$

(b)

```
h_y =(1,-1,-7) # given inputs
h_x =(0,-2,3)
x_y=(1,-1,4)
x_x=(0,5,-3)
y_nx =[-5,-3,-2,0,3,5,8]

def indexer (x,y):# indexing the given inputs between-6 and 8
    indexlist=np.zeros(15)
    for j in range(len(x)):
        indexlist[x[j]+6]=y[j]
    return indexlist

x_n= indexer(x_x,x_y)
y_n= indexer(h_x,h_y)

y_y=  np.convolve(x_n,y_n)#convolution part
y_x= np.arange(-6,9)

y_y =y_y[6:]# this part is required to match y_x and y_y values
y_y =y_y[:15]


plt.figure(figsize=(15, 4))# plotting part
plt.subplot(1,3,1)
markerline, stemlines, baseline = plt.stem(h_x, h_y, '-.')
plt.setp(baseline, 'color', 'r', 'linewidth', 2)
plt.title('h[n]')
plt.subplot(1,3,2)
markerline, stemlines, baseline = plt.stem(x_x, x_y, '-.')
plt.setp(baseline, 'color', 'r', 'linewidth', 2)
plt.title('x[n]')
plt.subplot(1,3,3)
```

```
markerline, stemlines, baseline = plt.stem(y_x, y_y, '-.')
plt.setp(baseline, 'color', 'r', 'linewidth', 2)
plt.title('y[n]')
```

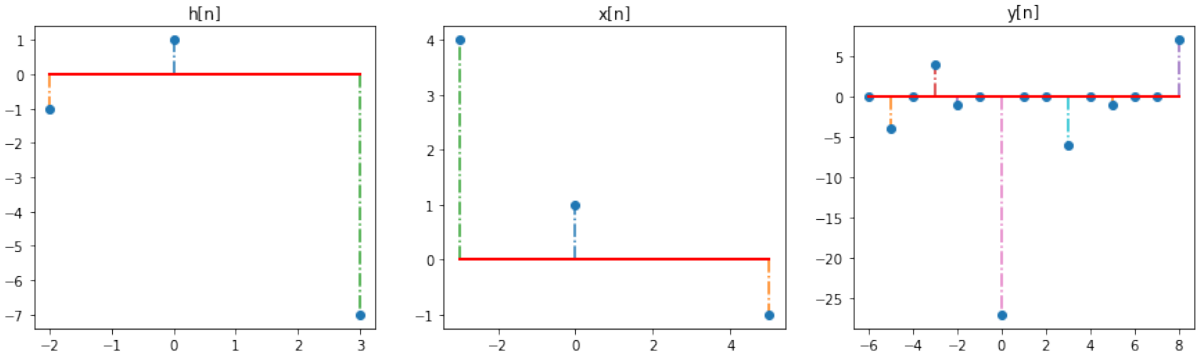


Figure 7: Plots

## Problem 5

(a) In this part forward running average filter and forward median filter implemented.In order to get the same size with the wav file ,the output of the convolution is truncated.

Average filter

```
def average_filter (data ,n):
    channel_1 = data[:,0]
    channel_2 = data[:,1]
    channel1_output = np.empty([873693])
    channel2_output = np.empty([873693])

    for i in range(len(channel_1)):
        channel1_output[i] = np.sum(channel_1[i:i+n])/n
    for i in range(len(channel_2)):
        channel2_output[i] = np.sum(channel_2[i:i+n])/n

    output  = np.c_[channel1_output,channel2_output]
    return output
```

Median Filter

```
def median_filter (data ,n):
    channel_1 = data[:,0]
    channel_2 = data[:,1]
    channel1_output = np.empty(len(channel_1))
    channel2_output = np.empty(len(channel_1))
    for i in range(len(channel_1)):
        channel1_output[i] = np.median(channel_1[i:i+n])
    for i in range(len(channel_2)):
        channel2_output[i] = np.median(channel_1[i:i+n])
    output  = np.c_[channel1_output,channel2_output]
    return output
```

-MSE

```
def mse(a,b):
    MSE = np.square(np.subtract(a,b)).mean()
    return MSE
```

(b) It is observed that running average filter gives better results in this data against median filter

;however, applying this filters in time domain does not provide any improvement on noisy data.In order to take better results frequency domain should be used. Average filter and median filters give 38749 when n = 2 but as n increases averager filter give better results.

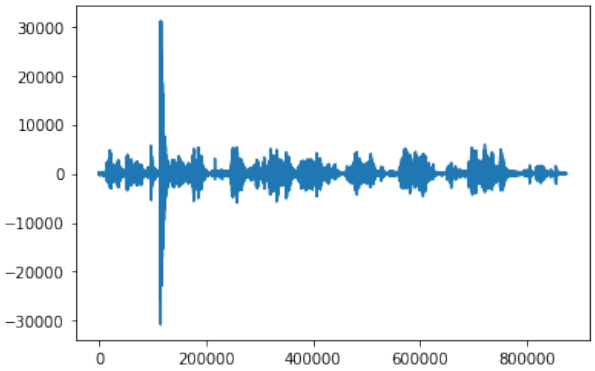


Figure 8: Clean data in time domain

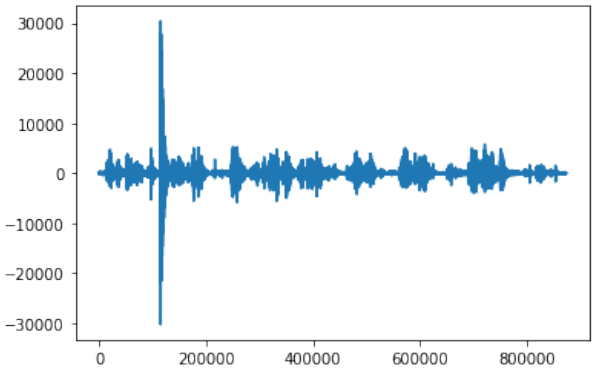


Figure 9: Averager filter applied data in time domain n=5

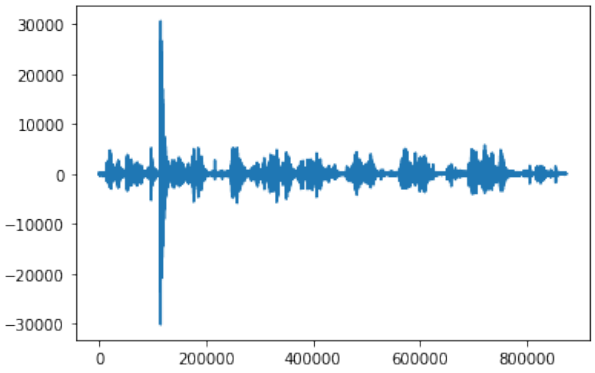


Figure 10: Median filter applied data in time domain n=5

(c) Running average filter is linear filter because its output is linear combination of the neighbouring points, on the other side median filter is not a linear filter because its output is not a linear function of its input.

(d) Bonus problem is explained in the jupyter notebook.

## Problem 6

Using the convolution properties general h[n] can be found such as :

$h[n] = h_1[n] * (h_2[n] + h_3[n] + h_4[n])$

$h[n] = \alpha^n u[n](u[n+3] + \delta[n+1])$

$h[n] = \alpha^n u[n] * u[n+3] + \alpha^n u[n] * \delta[n+1]$

When $n \geq 0$

$\alpha^n u[n] * u[n+3] = \sum_0^{n+3} a^k = \frac{1-a^{n+1}}{1-a} + a^{n+1} + a^{n+2} + a^{n+3}$

when $0 \leq n < 1$

$y[n] = \frac{1-a^{n+1}}{1-a} + a^{n+1} + a^{n+2} + a^{n+3}$

when $1 \leq n$

$y[n] = \frac{1-a^{n+1}}{1-a} + 2a^{n+1} + a^{n+2} + a^{n+3}$

The same h[n] can be found also without using convolutiıon properties but with forward calculation.

$h[n] = (h_1[n] * h_2[n]) + (h_1[n] * h_3[n]) + (h_1[n] * h_4[n])$

$h[n] = (\alpha^n u[n] * u[n]) + (\alpha^n u[n] * (u[n+3] - u[n])) + (\alpha^n u[n] * \delta[n+1])$

$h[n] = \alpha^n u[n] * u[n+3] + \alpha^n u[n] * \delta[n+1]$

From the step above , calculations are same with the with the first part .