Cairo University
Faculty of Engineering

Department of Computer Engineering

ELC 325B – Spring 2023

Digital Communications

Assignment #2

Matched Filters

Submitted to

Dr. Mai

Dr. Hala

Eng. Mohamed Khaled

Submitted by

Name	Sec	BN
Nancy Ayman	2	27
Yara Hisham	2	31

Contents

Part I Requirements: 4

a) s(t) for b_0 = ‘0’, b_1 = ‘1’, and b_2 = ‘1’ 4

b) Matched filter output due to signal only 5

c) Sampling instants markings 6

d) Block diagram of transmitter 6

e) Block diagram of receiver 6

Part II Requirements: 7

1. Probability of error in the three mentioned cases 7

.... 9

2. Python code 13

3. And 4. Output of the receive filter for the three mentioned cases 17

5. BER is decreasing function of $EN0$ 19

6. Matched filter has the lowest BER 19

Figures

Figure 1 : Part I a 4
Figure 2 : Part I b 5
Figure 3 : Part I c 6
Figure 4 : Part I d 6
Figure 5 : Part I e 6
Figure 6: Filters output 17
Figure 7: BER vs E/N0 for each filter. 18
Figure 8: BER vs E/N0 for all filters. 18

Part I Requirements:

a) s(t) for b_0 = '0', b_1 = '1', and b_2 = '1'

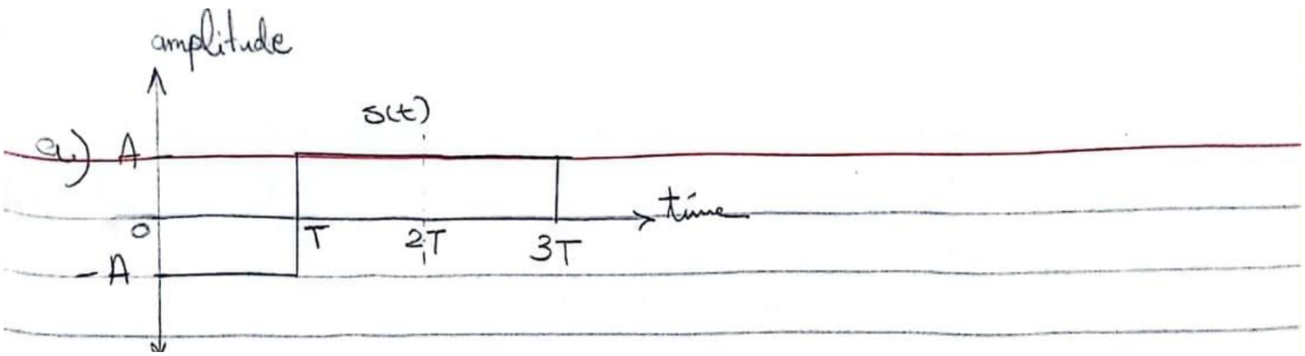
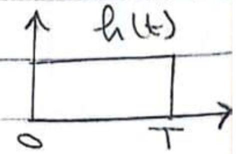


Figure 1 : Part I a

b) Matched filter output due to signal only

b) $\therefore g(t) = A \operatorname{rect}\left(\frac{t - T/2}{T}\right)$

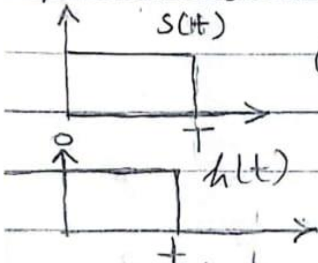
$\because h(t) = g(T-t) \Rightarrow \therefore h(t) = A \operatorname{rect}\left(\frac{T/2 - t}{T}\right)$



$$y(t) = s(t) \circledast h(t) = \int_{-\infty}^{\infty} s(\tau)\, h(t-\tau)\, d\tau$$

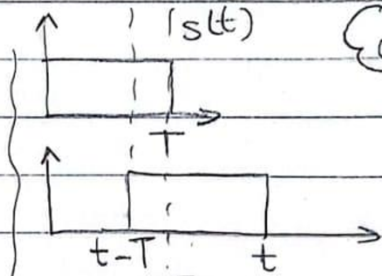
→ 1st Case : '1' was sent

when $0 < t < T$



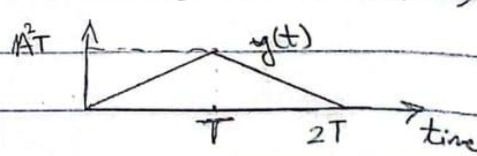
$$y(t) = \int_0^t A^2 d\tau = A^2 t$$

when $T < t < 2T$



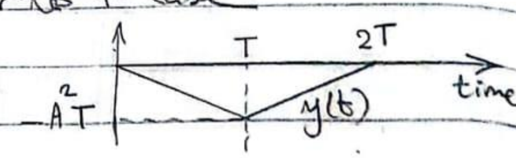
$$y(t) = \int_{t-T}^{T} A^2 d\tau = A^2[T - t + T] = A^2(2T - t)$$

$$\therefore y(t) = \begin{cases} A^2 t, & 0 < t < T \\ A^2(2T - t), & T < t < 2T \end{cases} \Rightarrow$$



→ 2nd Case : '0' was sent → Similar to 1st Case

$$y(t) = \begin{cases} -A^2 t, & 0 < t < T \\ -A^2(2T - t), & T < t < 2T \end{cases} \Rightarrow$$



$\therefore$ Final plot of matched filter:

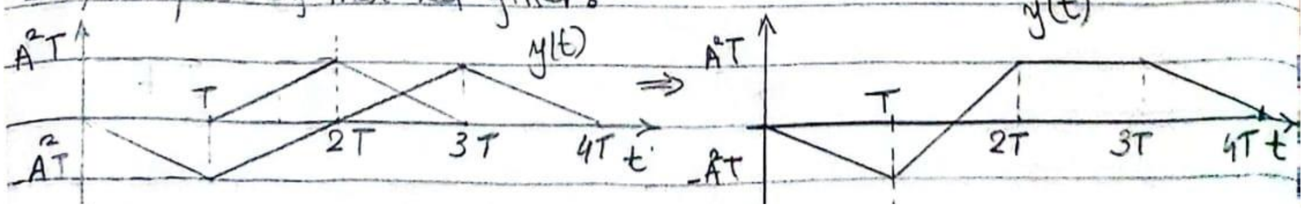


Figure 2 : Part I b

c) Sampling instants markings

(We sampled @ T)

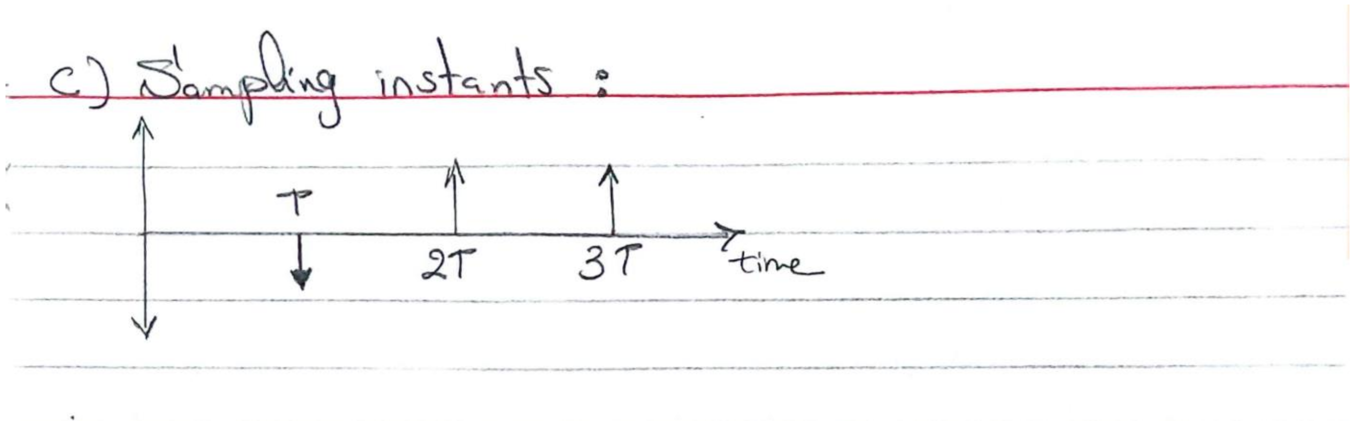


Figure 3 : Part I c

d) Block diagram of transmitter

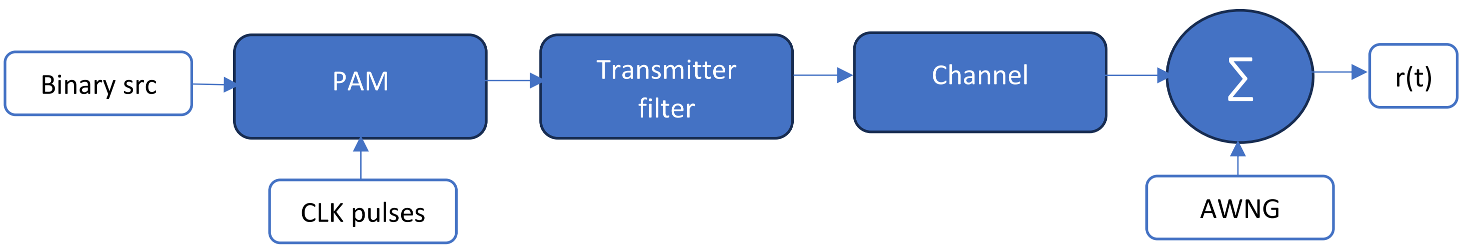


Figure 4 : Part I d

e) Block diagram of receiver

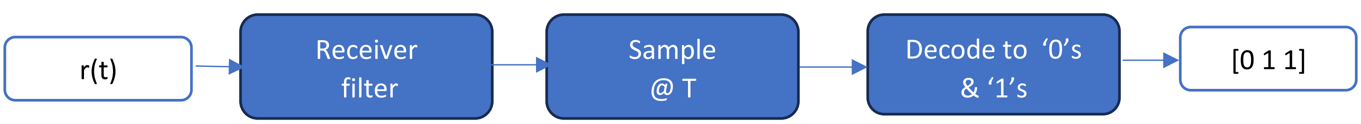


Figure 5 : Part I e

Part II Requirements:

1. Probability of error in the three mentioned cases

a) $h(t)$ is unit energy:

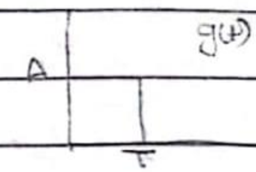
$$g(t) = A\,\mathrm{rect}\left(\frac{t-\frac{T}{2}}{T}\right)$$

$$h(t) = g(T-t) = A\,\mathrm{rect}\left(\frac{T-t-T/2}{T}\right) = A\,\mathrm{rect}\left(\frac{T/2-t}{T}\right)$$

$$r(t) = g(t) + w(t)$$

$$y(t) = r(t) * h(t) = g(t) * h(t) + w(t) * h(t)$$

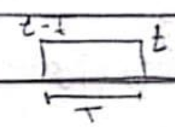
$$= g_o(t) + n(t)$$



for $t < 0$ $\quad g_o(t) = 0$

$0 < t < T$

$$g_o(t) = \int_0^t A^2\,dt = (A^2 t)\Big|_0^t = A^2 t$$



$T < t < 2T$

$$g(t) = \int_{t-T}^{T} A^2\,dt = A^2\left[T - t + T\right] = A^2\left[2T - t\right]$$

$t > 2T$ $\quad g_o(t) = 0$

$$g_o(t) = \begin{cases} A^2 t & 0 < t < T \\ A^2(2T-t) & T < t < 2T \\ 0 & \text{otherwise} \end{cases}$$

$$y(T) = \begin{cases} -A^2T + n(T) & \text{'0'} \\ A^2T + n(T) & \text{'1'} \end{cases}$$

$$\mu_y = E(y(T)) = E(g_o(T)) + E(n(T))$$

$$= E(\pm A^2 T) + E(n(T))$$

$$= \pm A^2 T + E(n(T))$$

$$E(n(T)) = E\left\{\int w(\tau)\, h(T-\tau)\, d\tau\right\}$$

$$= E\left\{\int w(\tau)\, g(\tau)\, d\tau\right\}$$

$$= E\int_0^T \pm A\, w(\tau)\, d\tau = \int_0^T \pm A\, E(w(\tau))\, d\tau = 0$$

$$\therefore \mu_y = \pm A^2 T$$

$$\sigma_y^2 = \mathrm{var}(y(T)) = E\left\{(g_o(T) + n(T) - \mu_y)^2\right\}$$

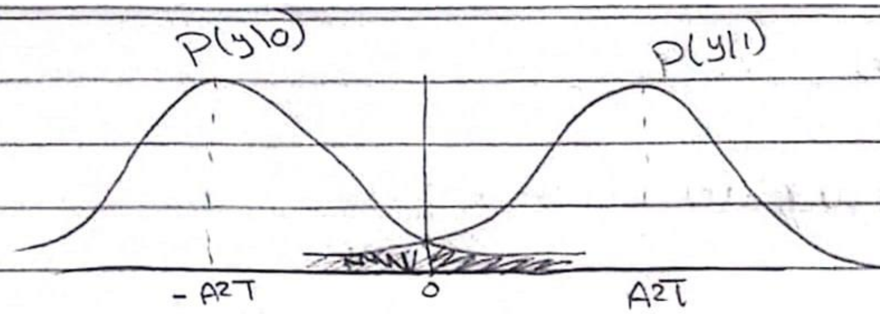
$$\sigma_y^2 = E\left\{(n(T))^2\right\} = \frac{N_0}{2}\int_{-\infty}^{\infty} |H(f)|^2\, df = \frac{N_0}{2}\int_0^T |h(t)|^2\, dt$$

$$\sigma_y^2 = \frac{N_0}{2} A^2 T$$

$$P(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y-\mu_y)^2}{2\sigma^2}\right)$$

$$P(y|0) = \frac{1}{\sqrt{\pi N_0 A^2 T}} \exp\left(\frac{-(y + A^2 T)^2}{N_0 A^2 T}\right)$$

$$P(y|1) = \frac{1}{\sqrt{\pi N_0 A^2 T}} \exp\left(\frac{-(y - A^2 T)^2}{N_0 A^2 T}\right)$$



$P(0) = P(1) = \frac{1}{2}$

$$P(e|0) = \int_0^{\infty} P(y|0)\, dy = \int_0^{\infty} \frac{1}{\sqrt{\pi N_0 A^2 T}} \exp\left(-\frac{(y+A^2T)^2}{N_0 A^2 T}\right) dy$$

$$= \int_{\frac{A^2T}{\sqrt{N_0 A^2 T}}}^{\infty} \frac{1}{\sqrt{\pi N_0 A^2 T}}\, e^{-z^2} \sqrt{N_0 A^2 T}\, dz \qquad \text{Let } z = \frac{y + A^2 T}{\sqrt{N_0 A^2 T}}$$

$$= \int_{\frac{A^2T}{\sqrt{N_0 A^2 T}}}^{\infty} \frac{e^{-z^2}}{\sqrt{\pi}}\, dz \qquad dz = \frac{dy}{\sqrt{N_0 A^2 T}}$$

$$= \frac{1}{2} \operatorname{erfc}\left[\frac{A^2 t}{\sqrt{N_0 A^2 T}}\right]$$

$$\therefore P(e) = P(e|0)P(0) + P(e|1)P(1) = \frac{1}{2}\left(P(e|0) + P(e|1)\right)$$

$$= P(e|0)$$

$$P(e) = \frac{1}{2} \operatorname{erfc}\left[\frac{A^2 T}{\sqrt{N_0 A^2 T}}\right] \quad \boxed{A=1 \quad T=1} = \frac{1}{2} \operatorname{erfc}\left[\frac{1}{\sqrt{N_0}}\right.$$

b) $h(t) = \delta(t)$

$$y(t) = r(t) * h(t) = r(t) * \delta(t)$$

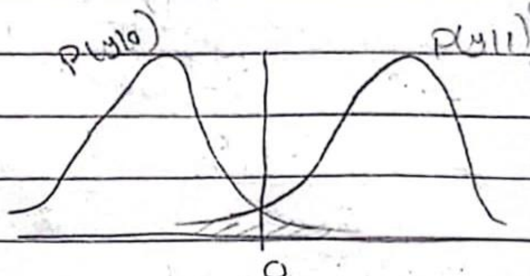
$$y(t) = g(t) + w(t) = \pm A + w(t) \qquad y(t) = \begin{cases} A + w(T) & '1' \\ -A + w(T) & '0' \end{cases}$$

$$E(y(t)) = E(\pm A + w(T)) = \pm A + \underbrace{E(w(T))}_{=0} = \pm A$$

$$\sigma_y^2 = Var(y(t)) = E(g(t) + n(T) - \mu_y)^2 = E(n(T))^2 = \frac{N_0}{2}\int_{-\infty}^{\infty} |H(f)|^2 \, df$$
$$= \frac{N_0}{2}\int_{-\infty}^{\infty} 1^2 \, df = \frac{N_0}{2}$$

$$P(y|'0') = \frac{1}{\sqrt{\pi N_0}} \exp\left(\frac{-(y+A)^2}{N_0}\right)$$

$$P(y|1) = \frac{1}{\sqrt{\pi N_0}} \exp\left(\frac{-(y-A)^2}{N_0}\right)$$



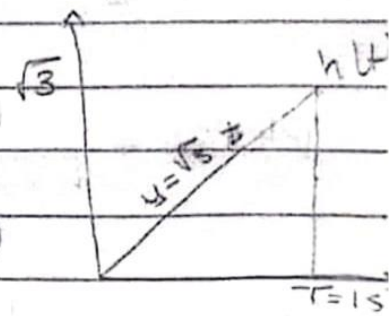
$$P(e|0) = \int_0^{\infty} P(y|0)\, dy = \int_0^{\infty} \frac{1}{\sqrt{\pi N_0}} \exp\left(\frac{-(y+A)^2}{N_0}\right) dy \qquad \text{let } z = \frac{y+A}{\sqrt{N_0}}$$

$$= \int_{\frac{A}{\sqrt{N_0}}}^{\infty} \frac{1}{\sqrt{\pi N_0}} e^{-z^2} \sqrt{N_0}\, dz = \int_{\frac{A}{N_0}}^{\infty} \frac{e^{-z^2}}{\sqrt{\pi}} dz \qquad dz = \frac{dy}{\sqrt{N_0}}$$

$$= \frac{1}{2} \operatorname{erfc}\left(\frac{A}{\sqrt{N_0}}\right) @ A = 1 \quad P(e|0) = \frac{1}{2}\operatorname{erfc}\left(\frac{1}{\sqrt{N_0}}\right)$$

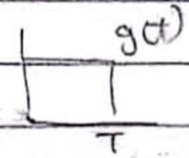
$P(e) = P(e|0)\,P(0) + P(e|1)\,P(1)$ $\quad \because P(0) = P(1) = 1/2$

$P(e) = P(e|0) = \frac{1}{2}\,\mathrm{erfc}\left[\frac{1}{\sqrt{N_0}}\right)$ $\underline{A=1}$ from Symmetry



c) $y(t) = r(t) * h(t) = g_0(t) + n(t)$

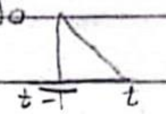
$g_0(t) = g(t) * h(t)$



for $t<0$ $\quad g_0(t) = 0$

for $0<t<T$

$$g_0(t) = \int_0^t \sqrt{3}\,A\tau\,d\tau = \sqrt{3}A\left[\frac{t^2}{2}\right]_0^t = \frac{\sqrt{3}}{2}At^2$$



$$y(t) = \begin{cases} \frac{\sqrt{3}}{2}At^2 + n(t) & \text{'1'} \\ -\frac{\sqrt{3}}{2}AT^2 \; n(T) & \text{'0'} \end{cases}$$

$$\mu_y = E(y(T)) = E(g_0(T)) + E(n(T)) = \pm\frac{\sqrt{3}}{2}AT^2 + E(n(T))$$

$$E(n(T)) = E\left(\int w(\tau)\,h(T-\tau)\,d\tau\right) = E\left(\int w(\tau)\,g(\tau)\,d\tau\right)$$
$$= E\left(\int_0^T \pm A\,w(t)\,dt\right) = \int_0^T \pm A\,E(w(t))\,dt = 0$$

$$\mu_y = \pm\frac{\sqrt{3}}{2}AT^2$$

Date:......................

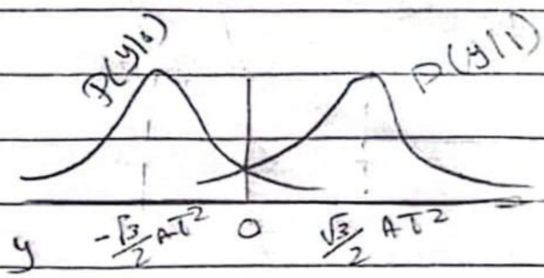
$$\sigma_y^2 = Var(y(t)) = E(n(T)^2) = \frac{N_0}{2}\int_{-\infty}^{\infty} |H(f)|^2 df = \frac{N_0}{2}\int_{-\infty}^{\infty} |H(t)|^2 dt$$

$$= \frac{N_0}{3}\int_0^T 3t^2\, dt = \frac{N_0}{2}\left[3\frac{t^3}{3}\right]_0^T = \frac{N_0 T^3}{2}$$

$$P(y|0) = \frac{1}{\sqrt{\pi N_0 T^3}} \exp\left(\frac{-(y + \frac{\sqrt{3}}{2}AT^2)}{N_0 T^3}\right)$$

$$P(y|1) = \frac{1}{\sqrt{\pi N_0 T^3}} \exp\left(\frac{-(y - \frac{\sqrt{3}}{2}AT^2)}{N_0 T^3}\right)$$

$$P(e|0) = \int_0^{\infty} p(y|0)\, dy$$



$$= \int_0^{\infty} \frac{1}{\sqrt{\pi N_0 T^3}} \exp\left(\frac{-(y + \frac{\sqrt{3}}{2}AT^2)}{N_0 T^3}\right) dy$$

$$= \int_{\frac{\frac{\sqrt{3}}{2}AT^2}{\sqrt{N_0 T^3}}} \frac{1}{\sqrt{\pi N_0 T^3}} e^{-z^2} \sqrt{N_0 T^3}\, dz = \int_{\frac{\frac{\sqrt{3}}{2}AT^2}{\sqrt{N_0 T^3}}} \frac{e^{-z^2}}{\sqrt{\pi}}\, dz$$

Let $z = \frac{y + \frac{\sqrt{3}}{2}At^2}{\sqrt{N_0 T^3}}$

$dz = \frac{dy}{\sqrt{N_0 T^3}}$

$$= \frac{1}{2}\, erfc\left(\frac{\sqrt{3}/2\, AT^3}{\sqrt{N_0 T^3}}\right)$$

assume $P(0) = P(1) = 0.5$
due to Symmetry

$$P(e) = P(e|0) = \frac{1}{2}\, erfc\left(\frac{\sqrt{3}/2}{\sqrt{N_0}}\right) \text{ @ } A=1, T=1$$

2. Python code

```
import numpy as np
import matplotlib.pyplot as plt
import math

BITS_NUM = 5
SAMPLES_PER_BIT_NUM = 5
E=1
def generate_random_bits(size = BITS_NUM):
    return np.random.choice(a=[0,1] , size= size)

def generate_signal(random_bits,samples_per_bit_num,duration):
    signal = np.ones((len(random_bits),samples_per_bit_num))
    for i in range (len(random_bits)):
        if random_bits[i] == 1:
            signal[i] = np.ones(int(samples_per_bit_num * duration))
        elif random_bits[i] == 0:
            signal[i] = np.full(int(samples_per_bit_num * duration), -1)
    return signal/math.sqrt(samples_per_bit_num)

def generate_gaussian_noise (sigma, size):
    return np.random.normal(loc = 0 ,scale=sigma, size=size)

def add_noise(signal,sigma,samples_per_bit_num = SAMPLES_PER_BIT_NUM):
    noise = generate_gaussian_noise(sigma = sigma,
size=len(signal)*samples_per_bit_num)
    noisy_signal=np.copy(signal)
    for i in range(len(signal)):
        noisy_signal[i, :] +=noise[i*samples_per_bit_num:(i+1)*samples_per_bit_num]
    return noisy_signal

def calc_convolution(noisy_signal, filter):
    convolved = None
    convolved_sampled=np.zeros(noisy_signal.shape[0])
    if(filter is None):
        convolved = noisy_signal.flatten()
    else:
        convolved = np.convolve(noisy_signal.flatten(), filter)

    for i in range(noisy_signal.shape[0]):
        convolved_sampled[i] =convolved[(noisy_signal.shape[1] - 1) +
noisy_signal.shape[1] *i]

    convolved_sampled = (convolved_sampled > 0).astype(int)
    return convolved,convolved_sampled
```

```
def calc_sim_error(expected, received):
    return (np.sum(received != expected)) / len(expected)

def calc_theo_error(N,filter_type):
    if(filter_type==0 ): # matched
        return (0.5 * math.erfc(1/(N ** 0.5)))

    if(filter_type==1): #ramp
        return 0.5 * math.erfc((3 /N) ** 0.5 / 2)

    else: # none
        return (0.5 * math.erfc((1/math.sqrt(SAMPLES_PER_BIT_NUM))/(N ** 0.5)))



def plot_filter_out(ax,noisy_signal,random_bits,filter,filter_type,N):
    convolved,convolved_sampled = calc_convolution(noisy_signal=noisy_signal , filter=
filter)
    sim_error = calc_sim_error(random_bits,convolved_sampled)
    print("Prob of error for filter type ",filter_type," is ",sim_error)
    ax[filter_type].plot(range(0, convolved.flatten().shape[0]), convolved.flatten(),
label = "bit value")
    for i in range(SAMPLES_PER_BIT_NUM - 1, convolved.flatten().shape[0],
SAMPLES_PER_BIT_NUM):
        ax[filter_type].stem([i], [convolved.flatten()[i]], linefmt='magenta')
    ax[filter_type].set_xlabel('Time (s)')
    ax[filter_type].set_ylabel('Amplitude')
    if filter_type == 0:
        ax[filter_type].set_title(f'Matched Filter Output')
    elif filter_type == 1:
        ax[filter_type].set_title(f'Ramp Filter Output')
    else:
        ax[filter_type].set_title(f'No Filter Output')
    ax[filter_type].grid(True)
#testing
random_bits = np.array([0,1,0,0,0])
print("Signal ",random_bits)

E_over_N=10**(20/10)
N=E/E_over_N


signal=generate_signal(random_bits,SAMPLES_PER_BIT_NUM,1)
noisy_signal=add_noise(signal,np.sqrt(N/2),SAMPLES_PER_BIT_NUM)
fig, ax = plt.subplots(1, 3, figsize=(15, 5))



#0 --> Matched
```

```python
filter_matched=np.ones(SAMPLES_PER_BIT_NUM)
plot_filter_out (ax,noisy_signal , random_bits , filter_matched,0,N )


#1 --> Ramp
filter_ramp= np.linspace(0, 3**0.5, SAMPLES_PER_BIT_NUM)
plot_filter_out (ax,noisy_signal , random_bits , filter_ramp,1,N )

#2 --> None
plot_filter_out (ax,noisy_signal , random_bits , None,2,N )

plt.show()

#BER part

random_bits = generate_random_bits( size= 10**5)

matched_sim_BER,matched_theo_BER= [] , []
ramp_sim_BER,ramp_theo_BER= [] , []
none_sim_BER,none_theo_BER= [] , []

# Define a dictionary to store lists for each filter type
sim_BER_dict = {0: matched_sim_BER, 1: ramp_sim_BER, 2: none_sim_BER}
theo_BER_dict = {0: matched_theo_BER, 1: ramp_theo_BER, 2: none_theo_BER}
#BER part

random_bits = generate_random_bits( size= 10**5)

matched_sim_BER,matched_theo_BER= [] , []
ramp_sim_BER,ramp_theo_BER= [] , []
none_sim_BER,none_theo_BER= [] , []

# Define a dictionary to store lists for each filter type
sim_BER_dict = {0: matched_sim_BER, 1: ramp_sim_BER, 2: none_sim_BER}
theo_BER_dict = {0: matched_theo_BER, 1: ramp_theo_BER, 2: none_theo_BER}
def calc_filter_error (random_bits , convolved_sampled ,filter_type,N):

    # filter_type represents the type of filter (0 for matched, 1 for ramp, 2 for none)
    sim_error = calc_sim_error(random_bits, convolved_sampled)
    theo_error = calc_theo_error(N, filter_type)

    # Append the errors to the corresponding lists based on the filter type
    sim_BER_dict[filter_type].append(sim_error)
    theo_BER_dict[filter_type].append(theo_error)
for E_over_N_db in range (-10,21,1):
    E_over_N=10**(E_over_N_db/10)
    N=E/E_over_N
```

```
    signal=generate_signal(random_bits,SAMPLES_PER_BIT_NUM,1)
    noisy_signal=add_noise(signal,np.sqrt(N/2),SAMPLES_PER_BIT_NUM)

    #0 --> Matched
    convolved,convolved_sampled=calc_convolution(noisy_signal,filter_matched)
    calc_filter_error(random_bits,convolved_sampled,0,N)

    #1 --> Ramp

    convolved,convolved_sampled = calc_convolution(noisy_signal,filter_ramp)
    calc_filter_error(random_bits,convolved_sampled,1,N)

    #2 --> None
    convolved,convolved_sampled=calc_convolution(noisy_signal,None)
    calc_filter_error(random_bits,convolved_sampled,2,N)

sim_BER_label_dict = {0: 'matched_sim_BER', 1: 'matched_sim_BER', 2: 'matched_sim_BER'}
theo_BER_label_dict = {0: 'matched_sim_BER', 1: 'matched_sim_BER', 2:
'matched_sim_BER'}

def plot_filter_BER(ax,filter_type):
    ax[filter_type].semilogy(range(-10,21), sim_BER_dict[filter_type]
,label=sim_BER_label_dict[filter_type])
    ax[filter_type].semilogy(range(-10,21), theo_BER_dict[filter_type]
,label=theo_BER_label_dict[filter_type])
    ax[filter_type].legend()
    ax[filter_type].set_xlabel('E/N0 (dB)') # X-axis label
    ax[filter_type].set_ylabel('BER')
    if(filter_type == 0 ):
        ax[filter_type].set_title('BER vs. E/N0 (Matched Filter)')
    if(filter_type == 1 ):
        ax[filter_type].set_title('BER vs. E/N0 (Ramp Filter)')
    else:
        ax[filter_type].set_title('BER vs. E/N0 (None Filter)')

    ax[filter_type].set_ylim(10**(-4))
    ax[filter_type].grid(True)



fig, ax = plt.subplots(1, 3, figsize=(15, 5))
plot_filter_BER(ax,0)
plot_filter_BER(ax,1)
plot_filter_BER(ax,2)
plt.show()

#plot all
plt.semilogy(range(-10,21), matched_sim_BER ,label='matched_sim_BER')
```

```python
plt.semilogy(range(-10,21), matched_theo_BER,label='matched_theo_BER')
plt.semilogy(range(-10,21), ramp_sim_BER ,label='ramp_sim_BER')
plt.semilogy(range(-10,21), ramp_theo_BER,label='ramp_theo_BER')
plt.semilogy(range(-10,21), none_sim_BER ,label='none_sim_BER')
plt.semilogy(range(-10,21), none_theo_BER,label='none_theo_BER')
plt.legend()
plt.xlabel('E/N0 (dB)') # X-axis label
plt.ylabel('BER')
plt.title('BER vs. E/N0 for different Filters ')
plt.ylim(10**(-4))
plt.grid(True)
plt.tight_layout()
plt.show()
```

3. And 4. Output of the receive filter for the three mentioned cases

Original Signal = [0 1 0 0 0]

With probability of error of 0 for all filters

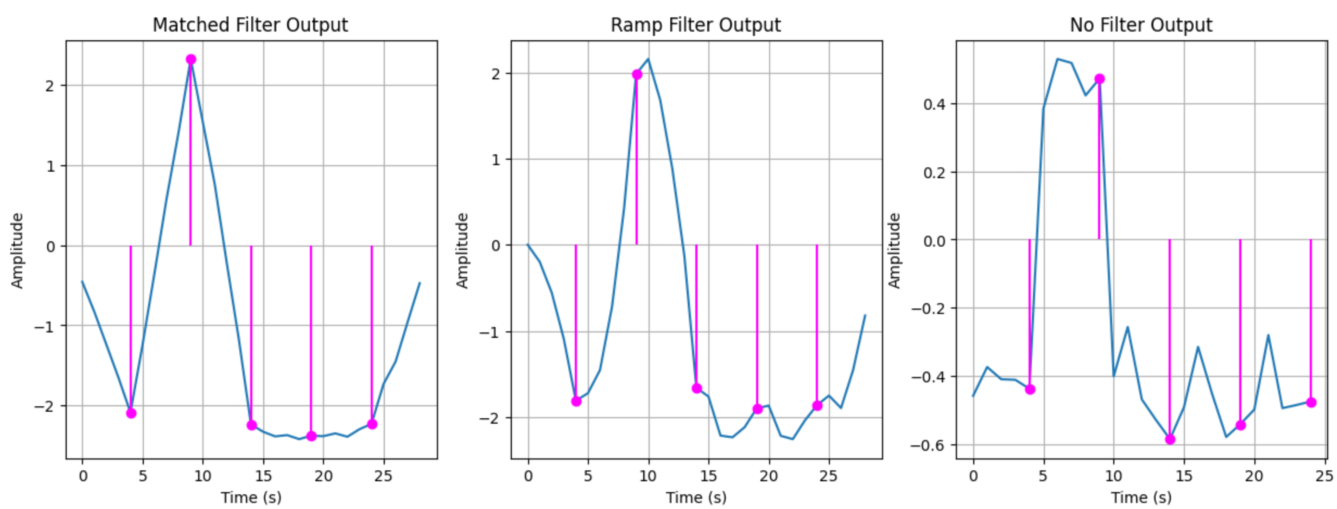


Figure 6: Filters output

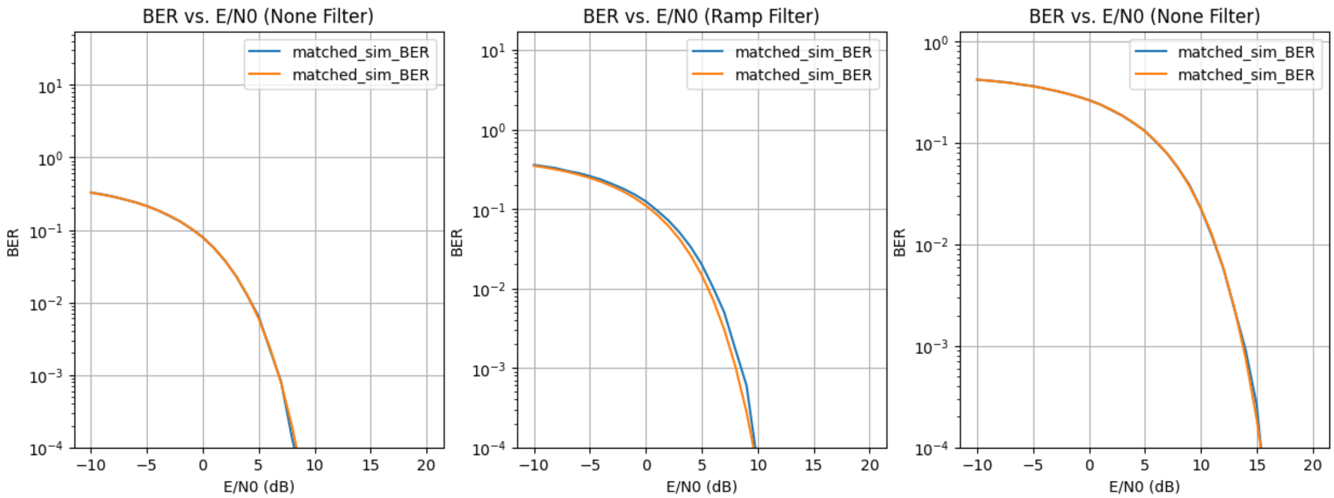


Figure 7: BER vs E/N0 for each filter.

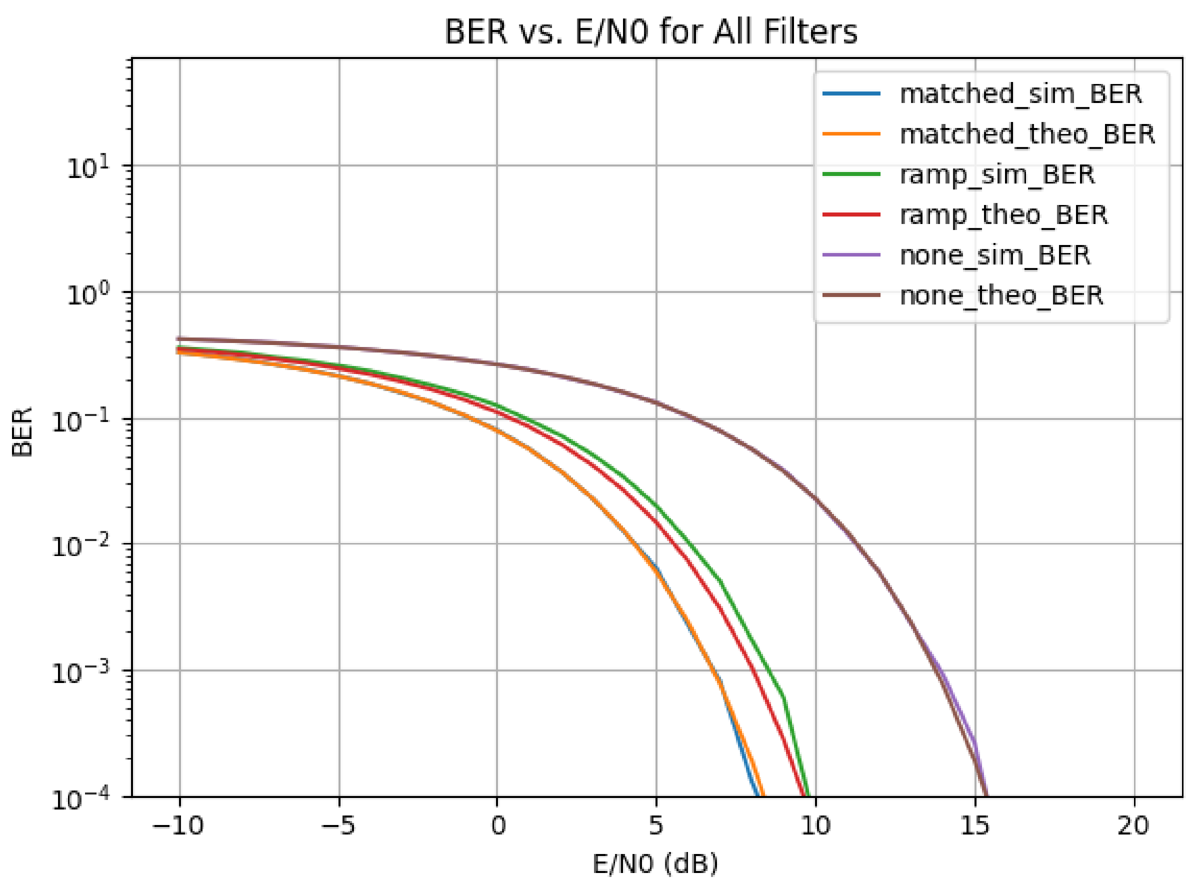


Figure 8: BER vs E/N0 for all filters.

5. BER is decreasing function of $\frac{E}{N_0}$

Increasing the E/No ratio reduces the bit error rate (BER). This entails boosting E/No, decreasing noise (N0), narrowing the Gaussian noise distribution, and enhancing SNR, thus minimizing noise interference, facilitating signal detection, and leading to an exponential decrease in BER as E/No rises.

6. Matched filter has the lowest BER

Because it’s designed to match the transmitted signal's characteristics. This allows the filter to provide the best detection strategy by comparing the received signal with a copy of the transmitted one. As a result, this reduces errors and maximizes the signal