Cairo University

Faculty of Engineering
Cairo University

Assignment2

Team Members:

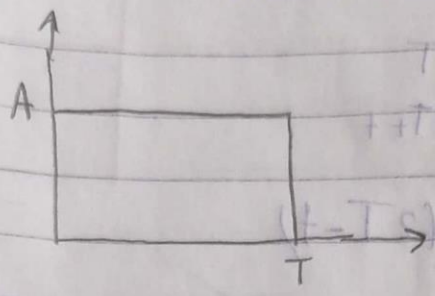
Name	Sec	BN
Zeinab Moawad Fayez Hassan	1	29
Basma Hatem Farid	1	17

Sumbitted To:

Eng: Mohamed Khaled

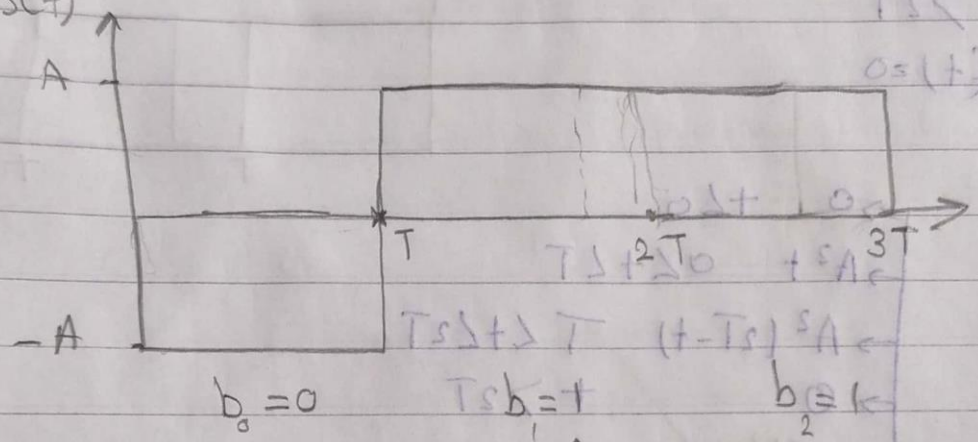
Assignment 2:

1. $g(t) =$



$$g(t) = \left(\frac{t - T/2}{T}\right)$$

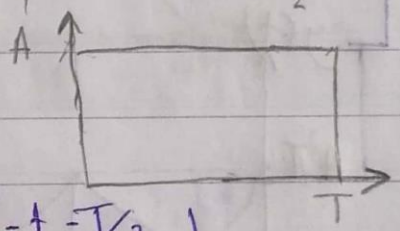
a) $s(t)$



$b_0 = 0$ $\quad b_1 = 1$ $\quad b_2 = 1$

b)

$$h(t) = g(T - t) =$$



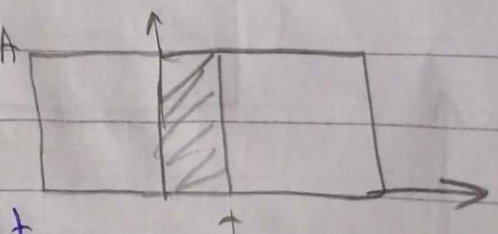
$$h(t) = A\,\text{rect}\left(\frac{T - t - T/2}{T}\right)$$

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

$$y_0(t) = s(t) * h(t)$$

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

Case 1: $t < 0 \longrightarrow y(t) = 0$



Case 2: $0 < t < T$

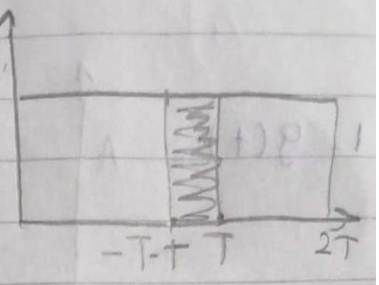
$$y(t) = \int_0^t A^2\, d\tau = (A^2 \tau)\Big|_0^t$$

$$= A^2 t$$

Case (3): $T < t < 2T$

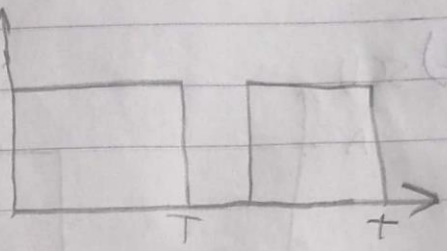
$$y(t) = \int_{-T+t}^{T} A^2 \, dt = (A^2 t)_{-T+t}^{T}$$

$$= A^2 (T + T - t) = A^2 (2T - t)$$

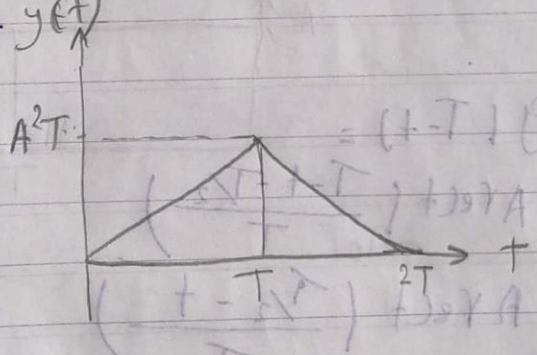


Case (4) $t > 2T$

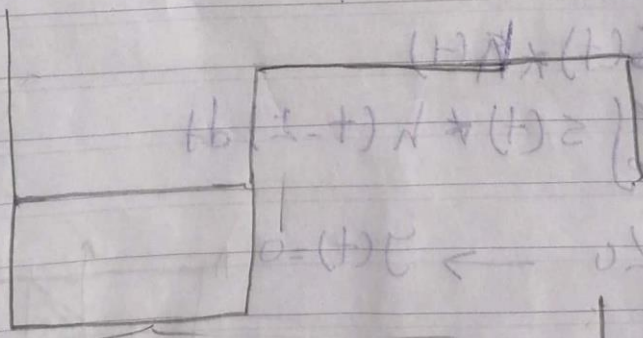
$y(t) = 0$



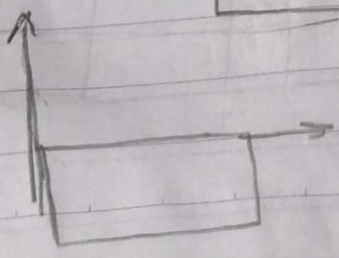
$$y(t) = \begin{cases} 0 & t < 0 \\ A^2 t & 0 < t < T \\ A^2 (2T - t) & T < t < 2T \\ 0 & t > 2T \end{cases}$$

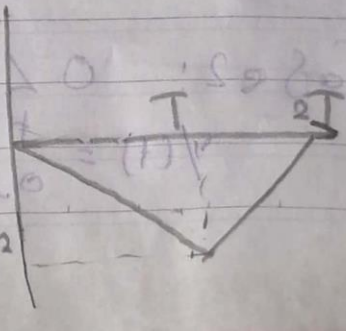


$g(t)$



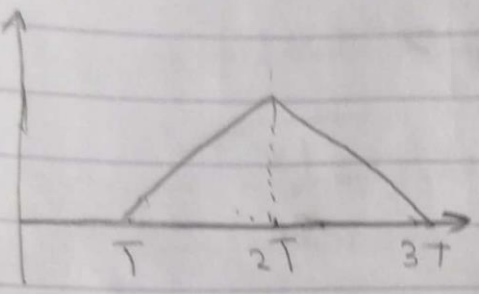
1)





$$y_1(t) = \begin{cases} -A^2 t & 0 \le t \le T \\ -A^2(2T-t) & T \le t \le 2T \\ 0 \end{cases}$$

$y_2(t)$

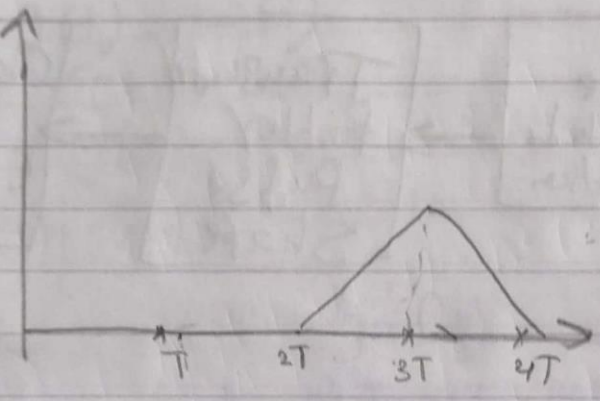


$y(t)$

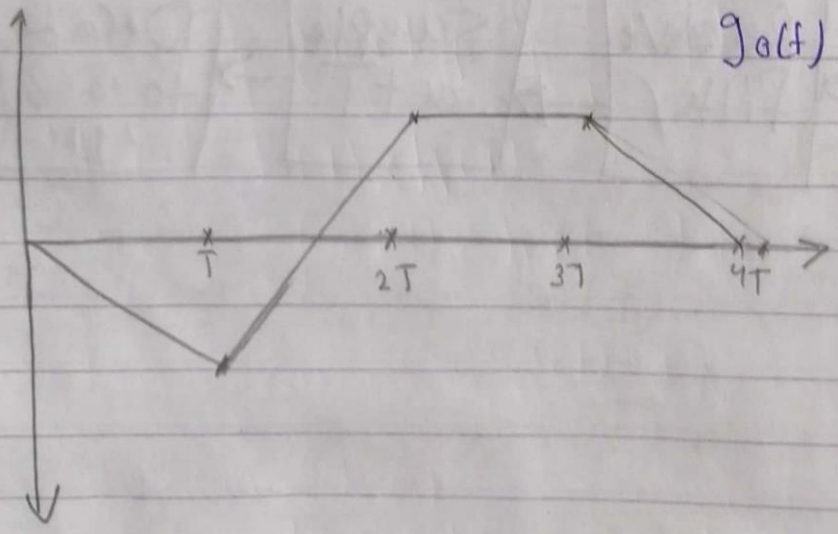
$A^2(t-1)$ $1 \le t \le 2$

$A^2($

$y_3(t)$



$$g_0(t) = y_1(t) + y_2(t) + y_3(t)$$



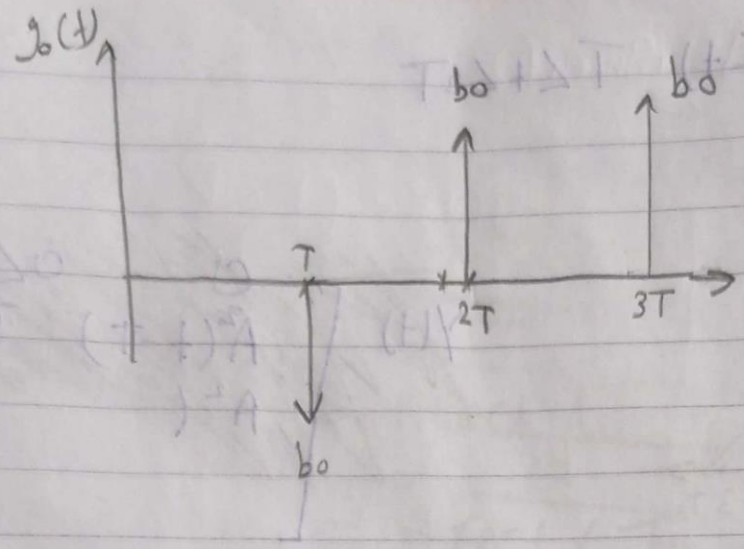
$g_0(t)$ → $= A^2 t$ $0 \le t \le T$

→ $A^2(2t-3T)$ $T < t \le 2T$

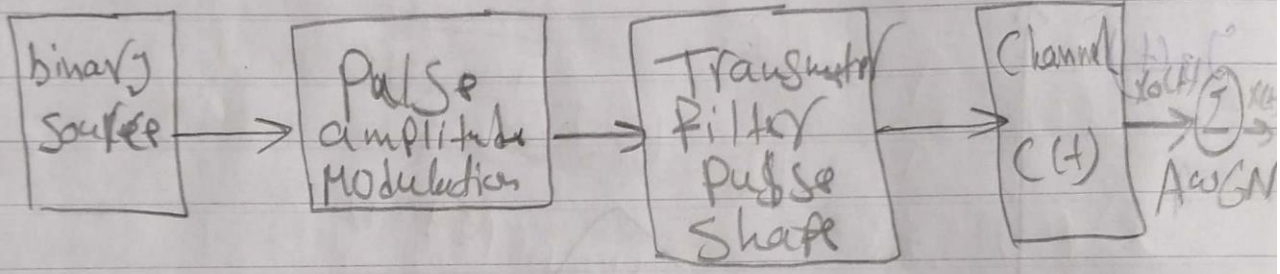
→ $A^2 T$ $2T \le t < 3T$

→ $A^2(4T-t)$ $3T \le t < 4T$

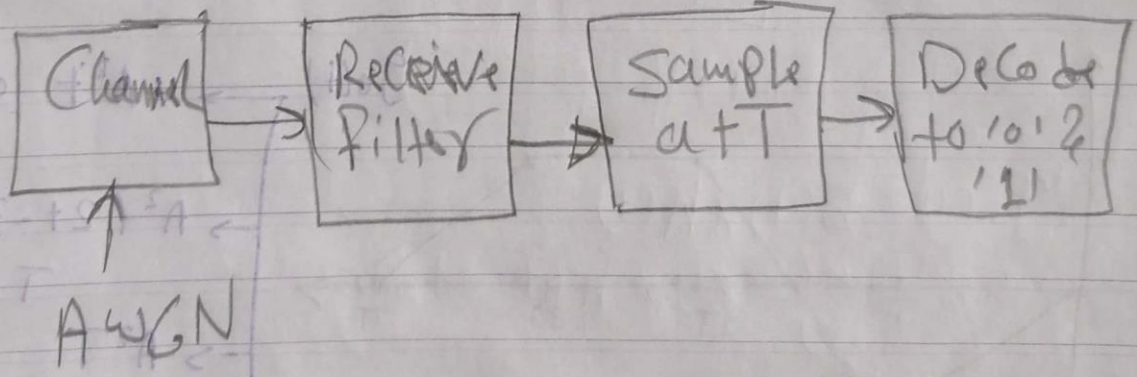
c)


$g_0(t)$
b_0
b_0
T
2T
3T
b_0

d)


binary source
Pulse amplitude Modulation
Transmit Filter pulse Shape
Channel
C(t)
AWGN

e)


Channel
Receive Filter
Sample at T
Decode to '0' & '1'
AWGN

Part (2):

a) $g(t) = \begin{cases} -A \\ A \end{cases}$ $0 < t < T$

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

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

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

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

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

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

$= g_0(t) + n(t)$

From Part 1: $g_0(t) = \begin{cases} 0 \\ A^2 T \end{cases}$ $0 < t < T$

$y(T) = \begin{cases} -A^2 T + n(T) & (0) \\ A^2 T + n(T) & (1) \end{cases}$ $\begin{cases} 0 \\ A^2(2T - t) \end{cases}$ $T < t < 2T$

$M_y = E(y(T)) = E(g_0(T)) + E(n(T))$

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

$E(n(T)) = E\left(\int w(t)\, n(T-t)\, dt\right)$

$= \int_0^T \pm A\, E(w(2))\, d2 = 0$

- $M_y = \pm A^2T$

$$M_y = \begin{cases} -A^2T & '0' \\ A^2T & '1' \end{cases}$$

$$\sigma_y^2 = \text{Var}(Y(T)) = E[g_0(T) + n(T)]$$

$$\text{Var}(n(T)) = E(n^2(t)) - E(n(t))^2 \quad \text{(zero)}$$

$$\sigma_y^2 = E(n^2(t)) = \int_{-\infty}^{\infty} S_n(f)\,df$$

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

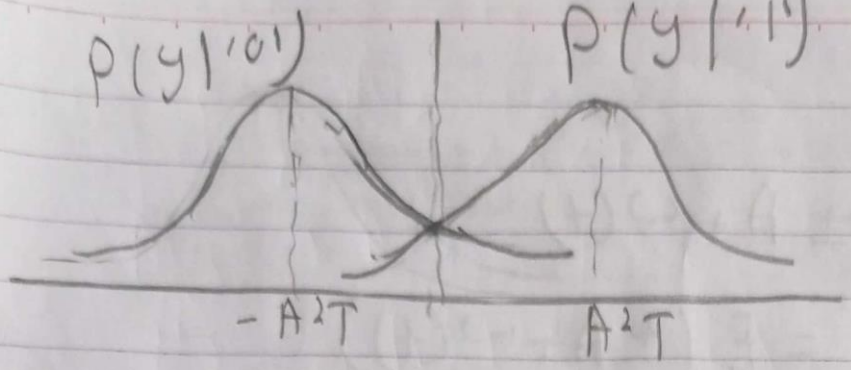
$$\sigma_y^2 = \frac{N_0}{2} \times A^2T = \frac{N_0A^2T}{2}$$

$$P(y) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\left(-\frac{(y-M_y)^2}{2\sigma^2}\right)}$$

$$P(y|'0') = \frac{1}{\sqrt{2\pi \times \frac{N_0A^2T}{2}}} \times e^{\left[-\frac{(y+A^2T)^2}{2\times \frac{N_0A^2T}{2}}\right]}$$

$$P(y|'0') = \frac{1}{\sqrt{\pi N_0A^2T}} \times e^{\left[\frac{-(y+A^2T)^2}{N_0A^2T}\right]}$$

$$P(y|'1') = \frac{1}{\sqrt{\pi N_0A^2T}} \times e^{\left(\frac{-(y-A^2T)^2}{N_0A^2T}\right)}$$



(d

$$P(e|'0') = \int_0^{\infty} \frac{1}{\sqrt{\pi N_0 A^2 T}} e^{\frac{-(y+A^2T)^2}{N_0 A^2 T}} dy$$

$$z = \frac{y+A^2T}{\sqrt{N_0 A^2 T}} \qquad dz = \frac{dy}{\sqrt{N_0 A^2 T}}$$

$$P(e|'0') = \int_{\frac{A^2T}{\sqrt{N_0 A^2 T}}}^{\infty} \frac{1}{\sqrt{\pi}\sqrt{N_0 A^2 T}} \times e^{-z^2} \times \sqrt{N_0 A^2 T}\, dz$$

$$P(e|'0') = \int_{\frac{A^2T}{\sqrt{N_0 A^2 T}}}^{\infty} \frac{1}{\sqrt{\pi}} e^{-z^2} dz = \frac{1}{2} \operatorname{erfc}\left(\frac{A^2T}{\sqrt{N_0 A^2 T}}\right)$$

$$P(e) = P(e|'1') \times P('1') + P(e|'0') P('0')$$

$\lambda = 0 \longrightarrow P(e|'1') = P(e|'0')$

$\hookrightarrow P('0') = P('1') = 0.5$

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

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

b)

$$y(t) = \pm A + \omega(t)$$

$$E(y(t)) = E(\pm A + \omega(t))$$

$$= \pm A + E(\omega(t))$$

↳ zero

$$\mu_y = \begin{cases} -A & \text{'0'} \\ A & \text{'1'} \end{cases}$$

$$\sigma_y^2 = \text{var}(y(T)) = \text{Var}(\pm A + \omega(T))$$

$$= \text{Var}(\omega(T))$$

$$\sigma_y^2 = \text{Var}(\omega(T)) = \frac{N_0}{2}$$

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

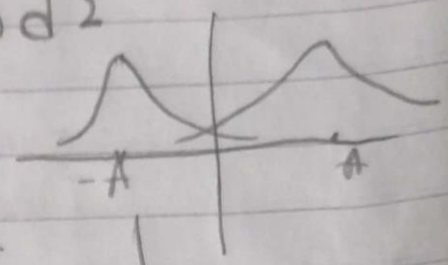
$$P(y|\text{'1'}) = \frac{1}{\sqrt{N_0 \pi}} e^{\frac{-(y-A)^2}{N_0}}$$

$$P(\text{'0'}) = P(\text{'1'}) \longrightarrow \lambda = 0$$

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

$$= \int_0^\infty \frac{1}{\sqrt{N_0 \pi}} e^{\frac{-(y+A)^2}{N_0}} dy$$

$$P(P|'0') = \int_{\frac{A}{\sqrt{N_0}}}^{\infty} \frac{1}{\sqrt{\pi}\sqrt{N_0}} e^{-z^2} \times \sqrt{N_0}\, dz$$



$$= \frac{1}{2} erfc\left(\frac{A}{\sqrt{N_0}}\right)$$

$$P(P) = P(P|'0') \times P('0') + P(P|'1') P('1')$$

$$= 2 \times 0.5 \times P(P|'0')$$

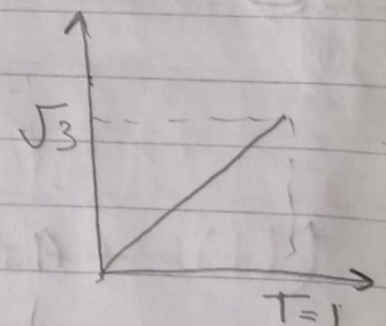
$$P(P|'0') = \frac{1}{2} erfc\left(\frac{A}{\sqrt{N_0}}\right) = \frac{1}{2} erfc\left(\frac{1}{\sqrt{N_0}}\right)$$

c)

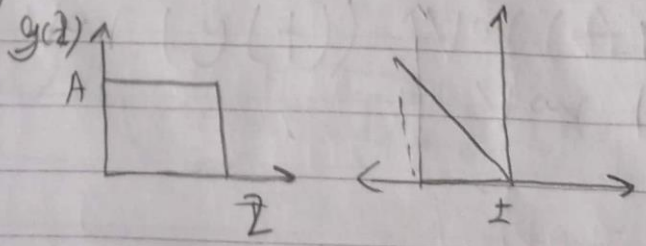
$$Y(t) = r(t) \oplus h(t) = g_0(t) + n(t)$$

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

$$g_0(t) = \int_{-\infty}^{\infty} g(z)\, h(t-z)\, dz$$



Case 1: $t < 0$



$$g_0(t) = 0$$

Case2: for $0 < t < T$

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

$$= \frac{\sqrt{3}}{2} A t^2$$

$$y(t) = \begin{cases} -\frac{\sqrt{3}}{2} A t^2 + n(t) & ,0 \\ +\frac{\sqrt{3}}{2} A t^2 + n(t) & ,1 \end{cases}$$

$$M_y = E(y(T)) = E(g_0(T)) + E(n(T))$$

$$M_y = E\left(\pm \frac{\sqrt{3}}{2} A T^2 + n(T)\right) = \pm \frac{\sqrt{3}}{2} A T^2 + E(n(T))$$

$$E(n(T)) = E\left(\int \omega(\lambda) g(\lambda) d\lambda\right)$$

$$= E\left(\int_0^T \pm A \omega(\lambda) d\lambda\right) = \int_0^T \pm A E(\omega(\lambda)) d\lambda$$

$$= 0$$

$$M_y = \begin{cases} -\frac{\sqrt{3}}{2} A T^2 & ,0 \\ \frac{\sqrt{3}}{2} A T^2 & ,1 \end{cases}$$

$$\sigma_y^2 = \text{var}(y(T)) = E(g_0(T) + n(T))$$

$$= E\left(\pm \frac{\sqrt{3}}{2} A T^2 + n(T)\right)$$

$$\sigma_y^2 = \text{Var}(n(T)) = E(n^2(T)) + E(n(T))^2$$

$$\sigma_y^2 = E(n^2(T)) = \int_{-\infty}^{\infty} S_n(f) df$$

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

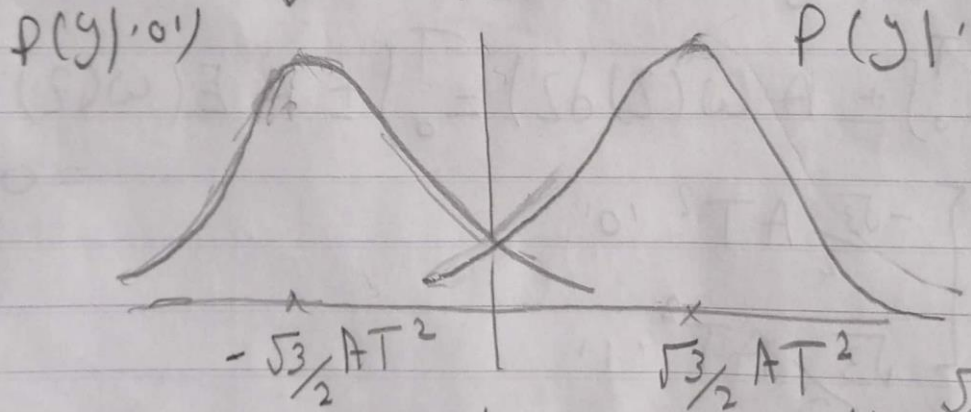
$$\sigma_y^2 = \frac{N_0}{2}\int_0^T 3+t^2\, dt = \frac{N_0}{2}\left[3+\frac{t^3}{3}\right]_0^T = \frac{N_0}{2}T^3$$

$$P(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-N)^2}{2\sigma^2}}$$

$$P(y|'0') = \frac{1}{\sqrt{2\pi \times \frac{N_0}{2} T^3}} e^{-\frac{(y+\frac{\sqrt{3}}{2}AT^2)^2}{N_0 T^3}}$$

$$= \frac{1}{\sqrt{\pi N_0 T^3}} e^{-\frac{(y+\frac{\sqrt{3}}{2}AT^2)^2}{N_0 T^3}}$$

$$P(y|'1') = \frac{1}{\sqrt{\pi N_0 T^3}} e^{-\frac{(y+\frac{\sqrt{3}}{2}AT^2)^2}{N_0 T^3}}$$



$$P(e|'0') = \int_0^{\infty} \frac{1}{\sqrt{\pi N_0 T^3}} e^{-\frac{(y+\frac{\sqrt{3}}{2}AT^2)^2}{N_0 T^3}}$$

$$Z = \frac{y+\frac{\sqrt{3}}{2}AT^2}{\sqrt{N_0 T^3}}$$

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

$$P(R|'0') = \int_{\frac{\sqrt{3}/2\,AT^2}{\sqrt{N_0T^3}}}^{\infty} \frac{1}{\sqrt{\pi}\sqrt{N_0T^3}} e^{-z^2} \times \sqrt{N_0T^3}\, dz$$

$$= \frac{1}{2}\,\mathrm{erfc}\left(\frac{\sqrt{3}/2\,AT^2}{\sqrt{N_0T^3}}\right)$$

$P('0') = P('1') \rightarrow \lambda = 0$

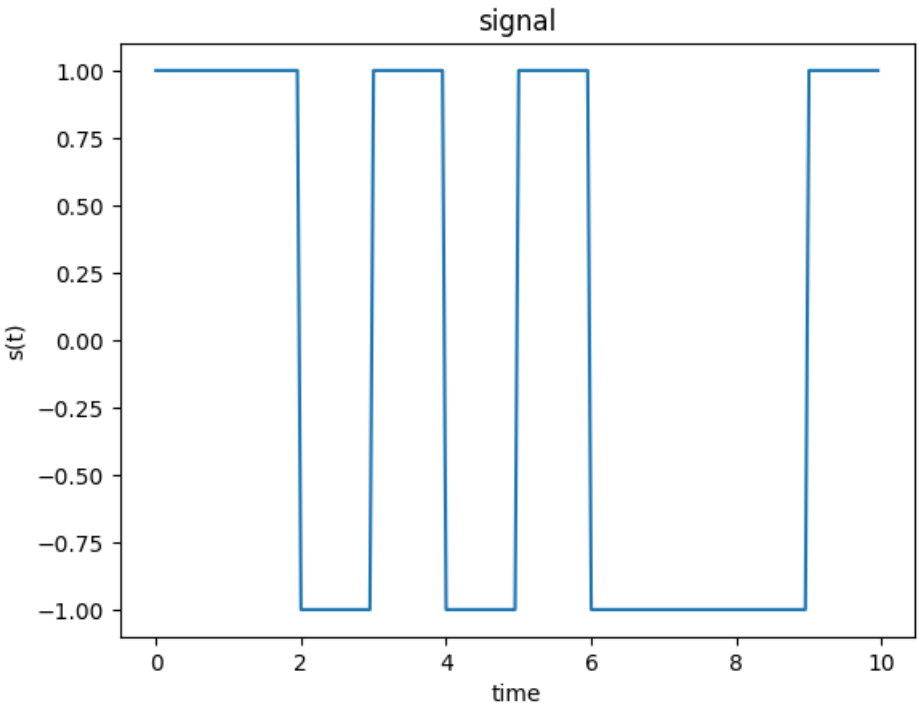
$$P(R) = P(R|'0') = \frac{1}{2}\,\mathrm{erfc}\left(\frac{\sqrt{3}/2\,AT^2}{\sqrt{N_0T^3}}\right)$$

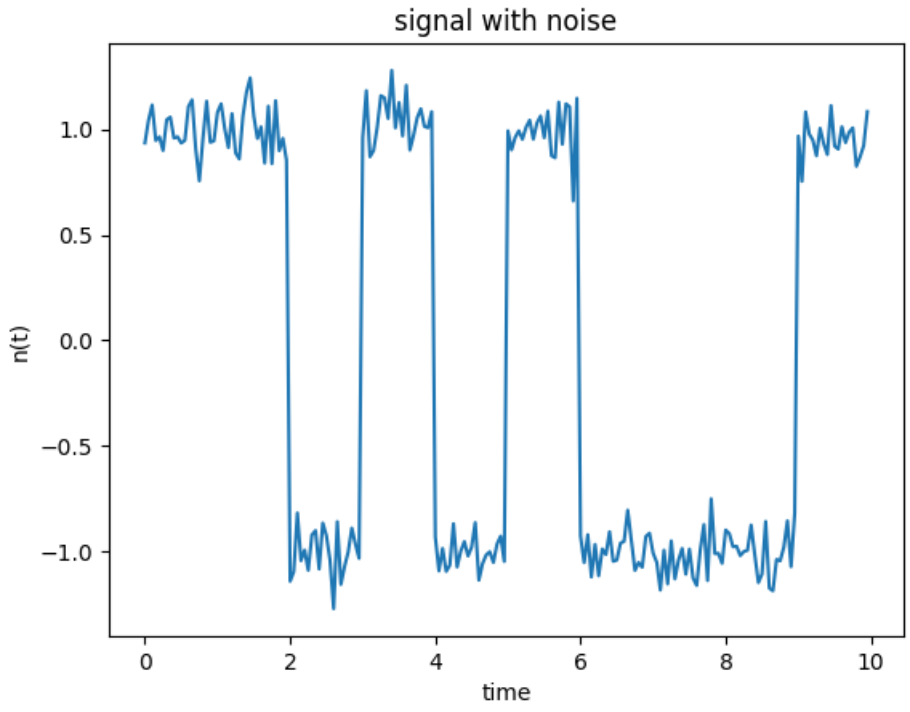
$A = 1, T = 1$

$$P(R) = P(R|'0') = \frac{1}{2}\,\mathrm{erfc}\left(\frac{\sqrt{3}/2}{\sqrt{N_0}}\right)$$

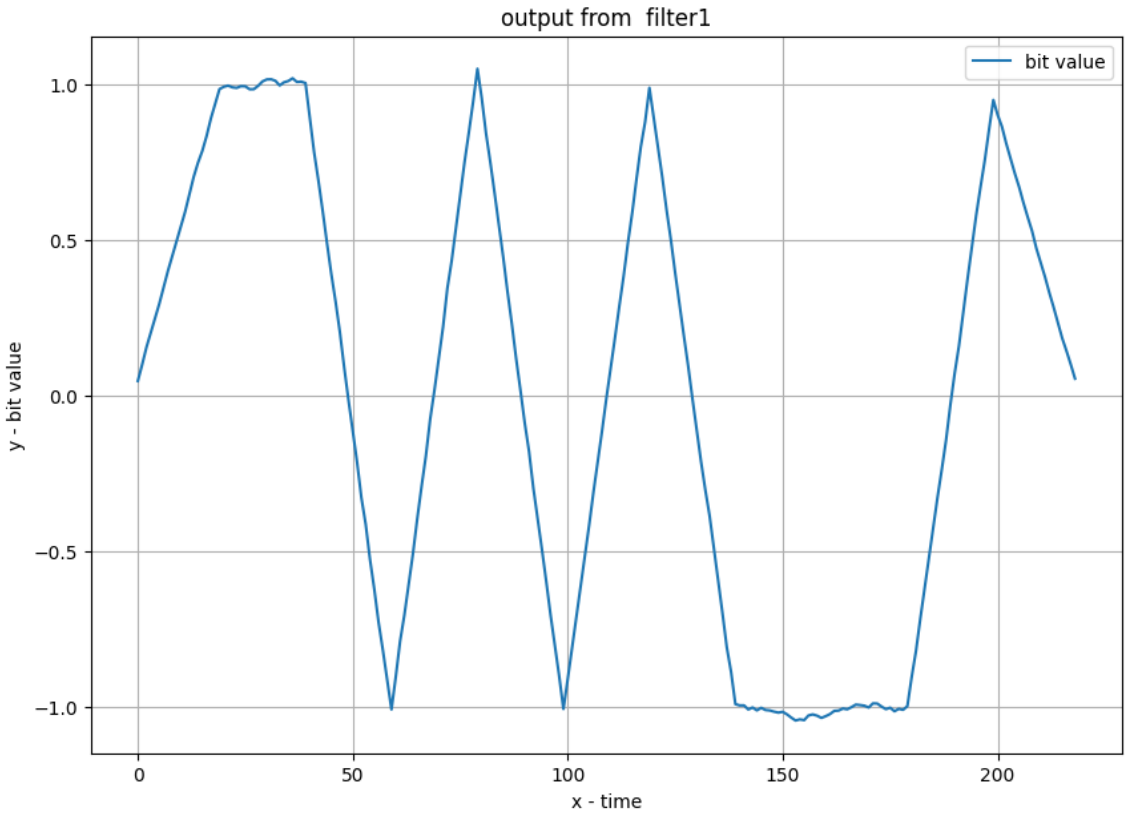
4- the required figures:

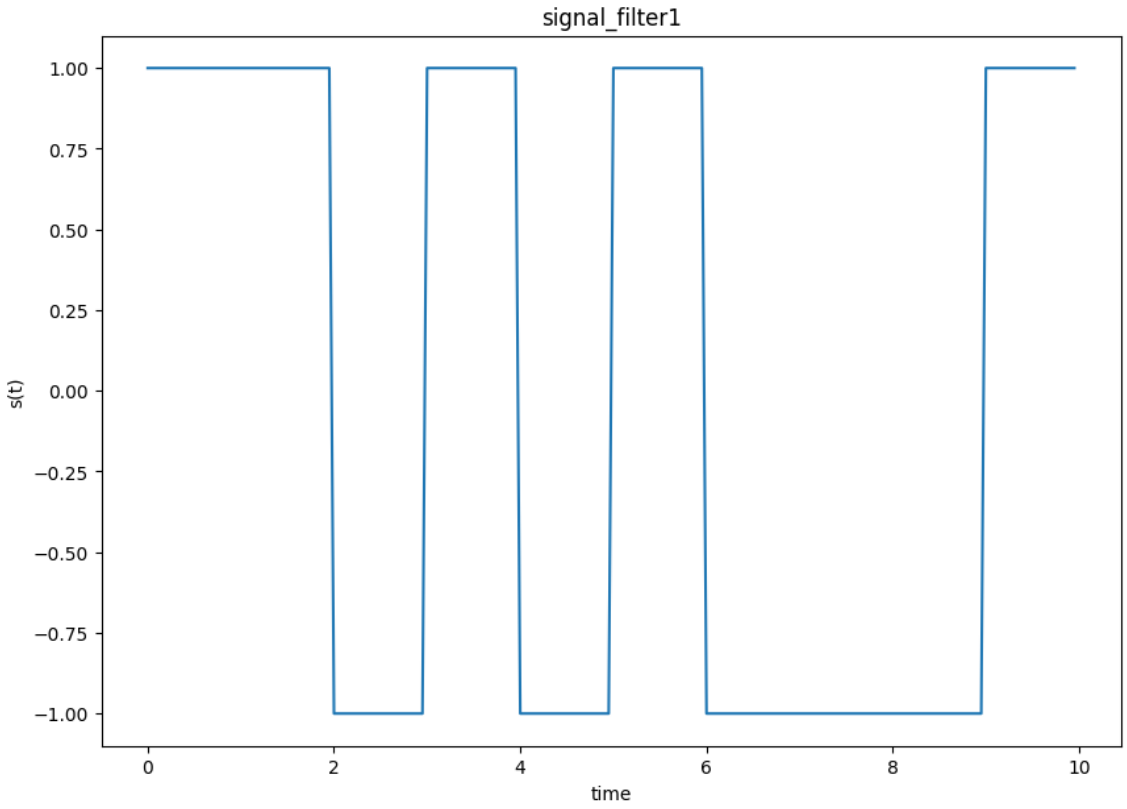
input bitstream: `1 1 0 1 0 1 0 0 0 1` with number of samples =20



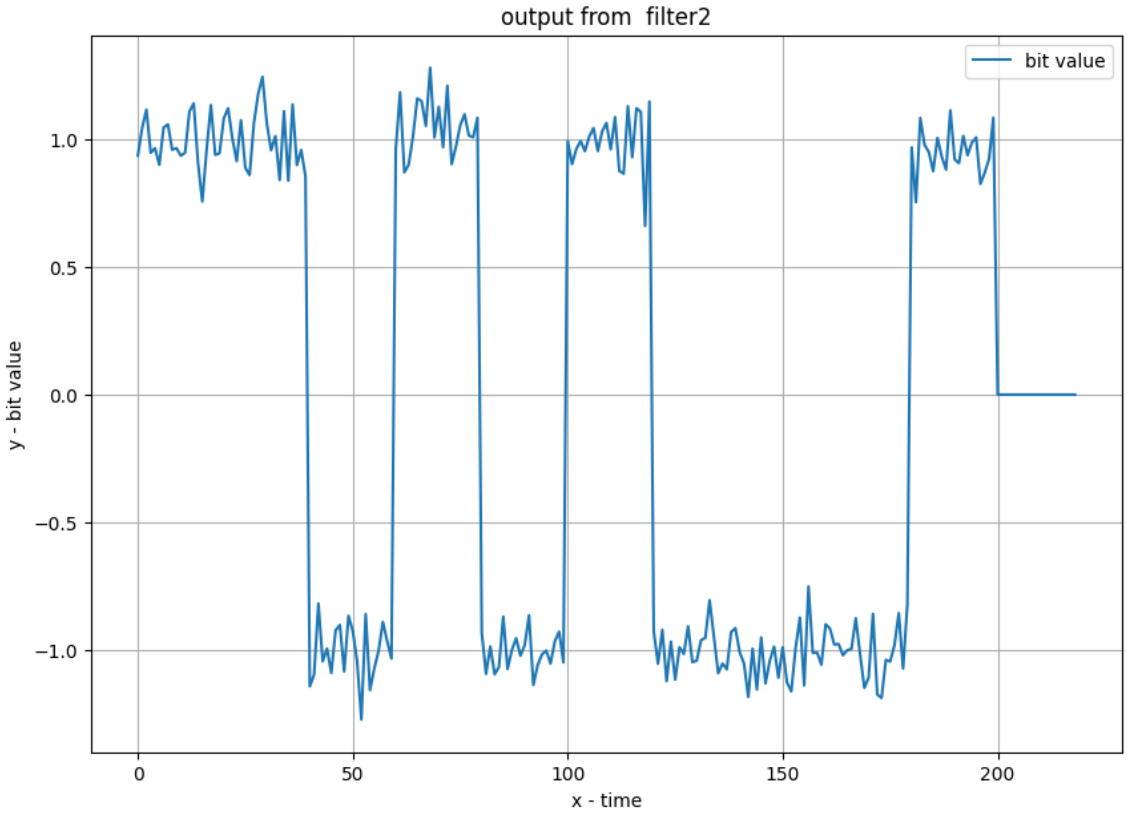


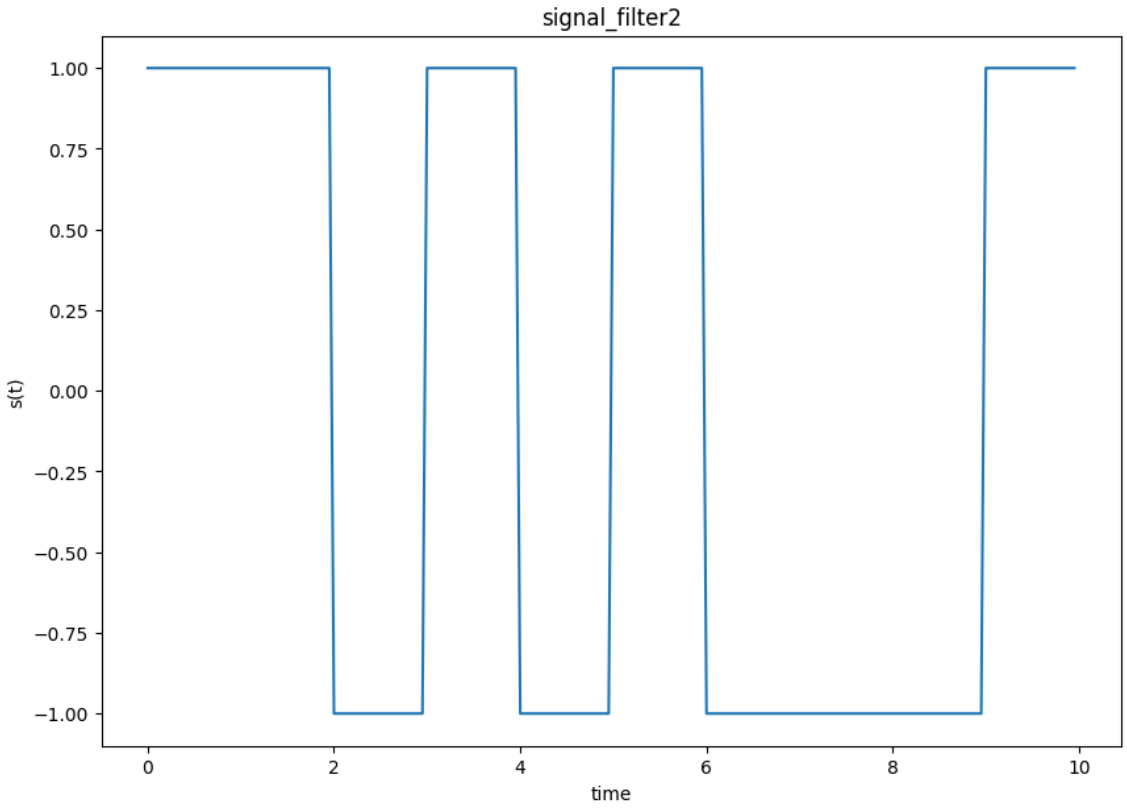
Output due to matched filter:



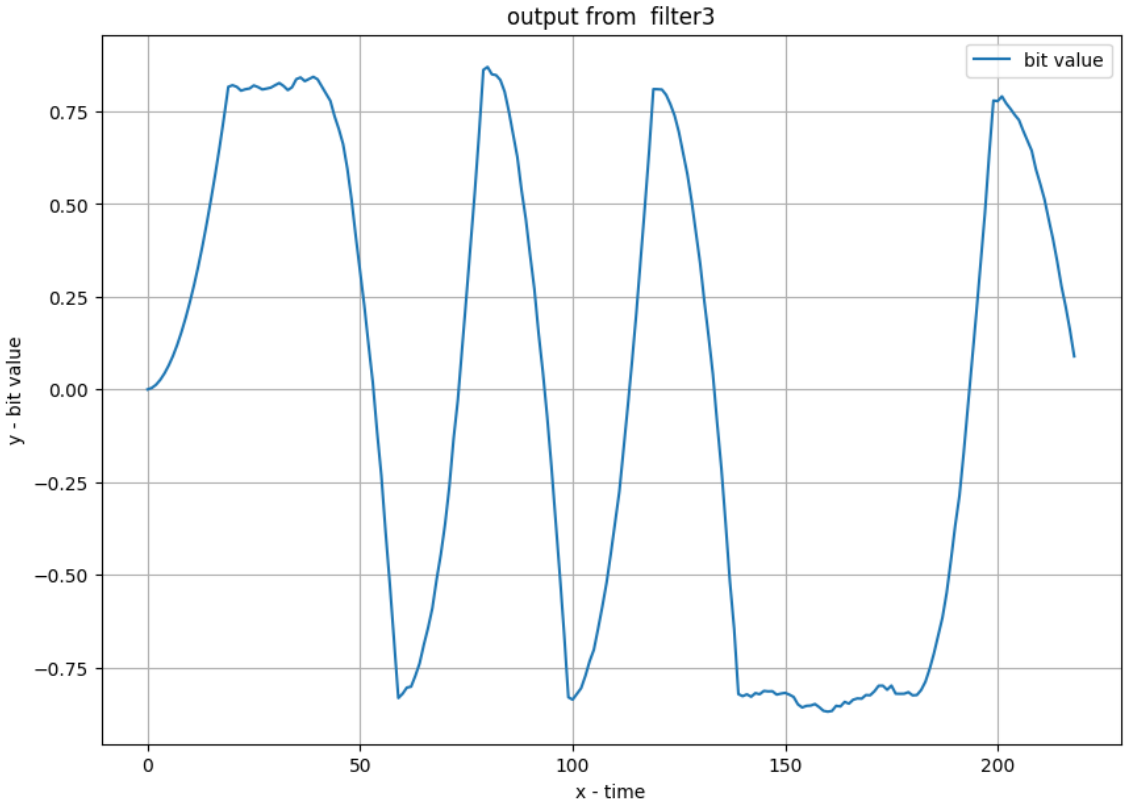


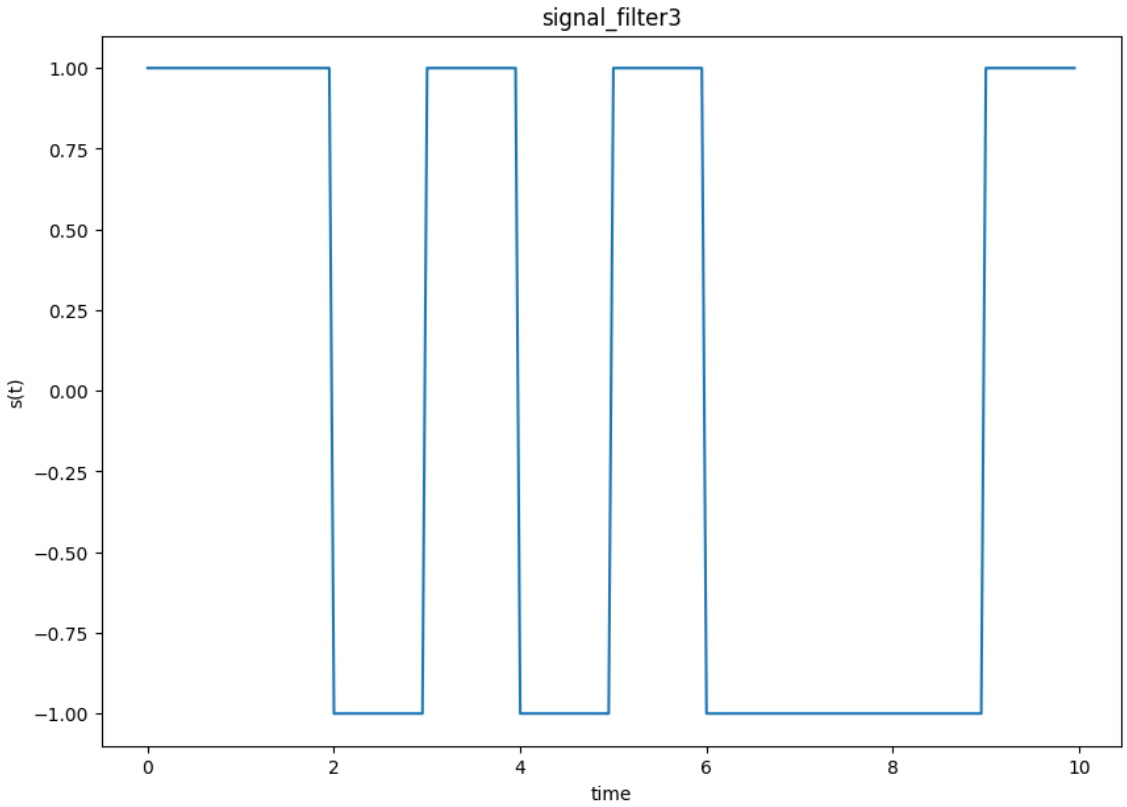
Output due to no filter:



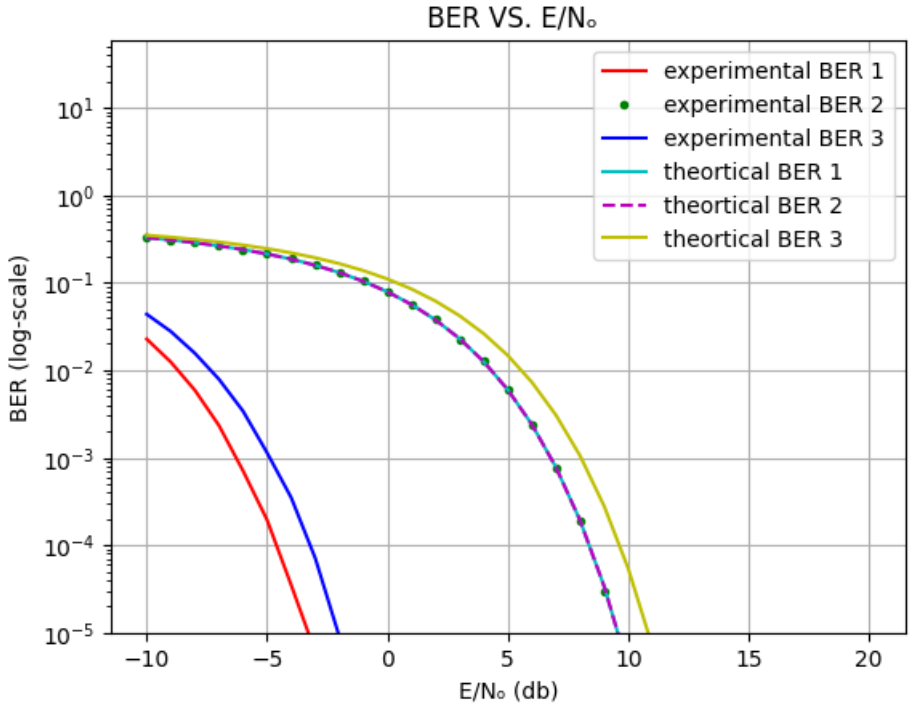


Output due to linear filter:


output from filter3
bit value
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
y - bit value
0
50
100
150
200
x - time


signal_filter3
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
s(t)
0
2
4
6
8
10
time

4-number of bits:1000000 , with number of samples =20



Comments to 5 and 6:

5-Is the BER an increasing or a decreasing function of E/No? Why?

Ans:The BER is decreasing as a function of E/No as in the plot. Due to two reasons:
1 - As E/No increases (here E is constant) No decreases and thus sigma which means the added AWGN involves less variations (corresponds to a thinner Gaussian distribution) and thus any noise added corresponds to small values close to zero which do not affect the signal that much (hence BER decreases)

2 -The relationship between E/No ratio and BER is not linear, but logarithmic
BER decreases exponentially as the E/No ratio increases

6-Which case has the lowest BER? Why?

Ans:The matched filter case is the lowest BER since it uses a filter matched to the pulse to minimize the probability of error so it is maximizes the peak pulse SNR at the samples.
And it is designed to match the characteristics of the transmitted signal
and this makes the filter provides the best detection strategy
by correlating the received signal with a replica of the transmitted signal
so as a result the BER will be reduced and the SNR will be maximized

Note : Theoritcally using a filter or not yields the same expression due to our assumptions on variance and PSD.

Code:

first try each filter:

*initial parameter:

```
#simulation parameters
n=10                        #number of bits
step=0.05                   #simulation time step (one pulse of duration 1
has 20 samples)
t=np.arange(0,n,step)
sigma_noise = 0.1
```

1- Get binary bits and change to pulses:

```
2- bitstream = np.random.randint(0,2,10)
3- print("Bitstream: ",bitstream)
4-
5- signal=binarycode_to_signal(bitstream,step)
6- plt.plot(t,signal)
7- plt.xlabel('time')
8- plt.ylabel('s(t)')
9- plt.title('signal')
10-plt.show()
```

2- Generate noise and add it to signal:

```
#generate the noise
signal_noise=add_AWGN_noise(signal,len(signal),sigma_noise)

plt.plot(t,signal_noise)
plt.xlabel('time')
plt.ylabel('n(t)')
plt.title('signal with noise')
plt.show()
```

3- try 3 recieve filters

```
signal_noise_filter_1=receive_filter(signal_noise, 1,step)
signal_noise_filter_2=receive_filter(signal_noise, 2, step)
signal_noise_filter_3=receive_filter(signal_noise, 3, step)
#ploting
plt.figure(figsize=(10,7))
plt.plot(range(0, signal_noise_filter_1.flatten().shape[0]),
signal_noise_filter_1.flatten(), label = "bit value")

plt.xlabel('x - time')
plt.ylabel('y - bit value')
plt.title('output from  filter1')

plt.legend()
plt.grid()
plt.show()
```

```
#ploting
plt.figure(figsize=(10,7))
plt.plot(range(0, signal_noise_filter_2.flatten().shape[0]),
signal_noise_filter_2.flatten(), label = "bit value")

plt.xlabel('x - time')
plt.ylabel('y - bit value')
plt.title('output from  filter2')

plt.legend()
plt.grid()
plt.show()
#ploting
plt.figure(figsize=(10,7))
plt.plot(range(0, signal_noise_filter_3.flatten().shape[0]),
signal_noise_filter_3.flatten(), label = "bit value")

plt.xlabel('x - time')
plt.ylabel('y - bit value')
plt.title('output from  filter3')

plt.legend()
plt.grid()
plt.show()
```

4- Reconstructing bits

```
#sample the filtered signal
sampling_period = int(1/step)
samples_1 = sampling(sampling_period, signal_noise_filter_1)
samples_2 = sampling(sampling_period, signal_noise_filter_2)
samples_3 = sampling(sampling_period, signal_noise_filter_3)
```

```
#decode the samples
reconstructed_bitstram_1 = signal_to_binarycode(samples_1, 0)
reconstructed_bitstram_2 = signal_to_binarycode(samples_2, 0)
reconstructed_bitstram_3 = signal_to_binarycode(samples_3, 0)
print('Reconstructed Bitstram 1:',reconstructed_bitstram_1)
print('Reconstructed Bitstram 2:',reconstructed_bitstram_2)
print('Reconstructed Bitstram 3:',reconstructed_bitstram_3)
#ploting
#generate the binary signal
signal_filter1=binarycode_to_signal(reconstructed_bitstram_1,step)
#generate the binary signal
plt.figure(figsize=(10,7))
plt.plot(t,signal_filter1)
plt.xlabel('time')
plt.ylabel('s(t)')
plt.title('signal_filter1')
plt.show()
```

```
#ploting
#generate the binary signal
signal_filter2=binarycode_to_signal(reconstructed_bitstram_1,step)
plt.figure(figsize=(10,7))
plt.plot(t,signal_filter2)
plt.xlabel('time')
plt.ylabel('s(t)')
plt.title('signal_filter2')
plt.show()

#ploting
#generate the binary signal
signal_filter3=binarycode_to_signal(reconstructed_bitstram_1,step)
plt.figure(figsize=(10,7))
plt.plot(t,signal_filter3)
plt.xlabel('time')
plt.ylabel('s(t)')
plt.title('signal_filter3')
plt.show()
```

Second Campare BER of Different receive filter:

```
num_of_bits = 1000000
step = 0.05 #Samples per pulse of duration 1
T=1 # pulse period
# generate the binary symbols (1)
input_bitstream = np.random.randint(0,2,num_of_bits)
# generate the binary signal (2)
g_t = binarycode_to_signal(input_bitstream, step)
# plot BER VS. E/No for each filter
E_N₀=np.arange(-10, 21, 1)    # E_N₀ range
N₀ = 1/(10**(E_N₀/10))
sigma = np.sqrt(N₀/2) # the range of sigma.

# Filter 1:
filter1_BER, filter1_BER_th = np.zeros(len(sigma)), np.zeros(len(sigma))
# Filter 2:
filter2_BER, filter2_BER_th = np.zeros(len(sigma)), np.zeros(len(sigma))
# Filter 3:
filter3_BER, filter3_BER_th = np.zeros(len(sigma)), np.zeros(len(sigma))

for i in range(len(sigma)):
    # generate the noise
    s_t = add_AWGN_noise(g_t,len(g_t), sigma[i])
    # apply the filter to the signal (4)
    y_t_1 = receive_filter(s_t,1, step)
    y_t_2 = receive_filter(s_t,2, step)
```

```
    y_t_3 = receive_filter(s_t,3, step)
    # sample the filtered signal (5)
    sampling_period = int(T/step)
    y_iT_1 = sampling(sampling_period, y_t_1,num_of_bits)
    y_iT_2 = sampling(sampling_period, y_t_2,num_of_bits)
    y_iT_3 = sampling(sampling_period, y_t_3,num_of_bits)
    # decode the samples (6)
    λ = 0      # due to
Assumation
    bitstream_output_1 = signal_to_binarycode(y_iT_1, λ)
    bitstream_output_2 = signal_to_binarycode(y_iT_2, λ)
    bitstream_output_3 = signal_to_binarycode(y_iT_3, λ)
    filter1_BER[i] =
calc_simulated_BER(input_bitstream,bitstream_output_1)
    filter1_BER_th[i] = calc_theoretical_BER(1/sigma[i])
    filter2_BER[i] =
calc_simulated_BER(input_bitstream,bitstream_output_2)
    filter2_BER_th[i] = calc_theoretical_BER(1/sigma[i])
    filter3_BER[i] =
calc_simulated_BER(input_bitstream,bitstream_output_3)
    filter3_BER_th[i] = calc_theoretical_BER(np.sqrt(3)/2*1/sigma[i])
```

plotting:

```
#ploting BER
E_N₀=np.arange(-10, 21, 1)
plt.semilogy(E_N₀, filter1_BER, 'r',label = "experimental BER 1")
plt.semilogy(E_N₀, filter2_BER, 'g.',label = "experimental BER 2")
plt.semilogy(E_N₀, filter3_BER, 'b',label = "experimental BER 3")
plt.semilogy(E_N₀, filter1_BER_th, 'c',label = "theortical BER 1")
plt.semilogy(E_N₀, filter2_BER_th, 'm--',label = "theortical BER 2")
plt.semilogy(E_N₀, filter3_BER_th, 'y',label = "theortical BER 3")

plt.xlabel('E/N₀ (db)')
plt.ylabel('BER (log-scale)')
plt.title(' BER VS. E/N₀')
plt.ylim(10**(-5))
plt.legend()
plt.grid()
plt.show()
```

Functions Used:

```
def binarycode_to_signal(bitstream, step):
    T = 1
    A = 1
    pulse = np.ones(int(T/step))
    pulse = pulse*A
    signal = np.zeros(len(bitstream)*len(pulse))
    for i in range(len(bitstream)):
        if bitstream[i] == 1:
            signal[i*len(pulse):(i+1)*len(pulse)] = 1*pulse
        else:
            signal[i*len(pulse):(i+1)*len(pulse)] = -1*pulse
    return signal
# add noise to signal
def add_AWGN_noise(signal,n, sigma):
     w_t =np.random.normal(0,sigma,n)
     s_t = signal + w_t
     return s_t
# received filter
def receive_filter(signal_noise, filter_num, step):
   filter_num-=1
   filters = [np.ones(int(1/step)), np.ones(1), np.sqrt(3)*np.arange(0,
1, step)]
   filter = filters[filter_num]
   filter = np.concatenate((filter, np.zeros(int(1/step)-len(filter))))
   signal_noise_filter=np.convolve(signal_noise, filter)
   if (filter_num==0 or filter_num==2):
      signal_noise_filter=signal_noise_filter*step
   return signal_noise_filter
# sampling filter
def sampling(sampling_period, signal_noise_filtered, n=10):
   samples = np.zeros(n)
   for i in range(len(samples)):
      samples[i] = signal_noise_filtered[sampling_period-
1+i*sampling_period]
   return samples

def signal_to_binarycode(samples, λ):
   return (samples>λ)*1

def calc_simulated_BER(input_bitstream , bitstream_output):

    sim_BER=np.sum(input_bitstream !=
bitstream_output)/len(input_bitstream)
    return sim_BER

def calc_theoretical_BER(erfc_parameter):
    return 0.5 * math.erfc(erfc_parameter/math.sqrt(2))
```