Cairo University
Faculty of
Engineering

Department of Computer
Engineering

ELC 325B – Spring 2024

Digital Communications

Assignment #2

Matched Filter

Submitted to

Dr. Hala

Dr. Mai

Eng. Mohamed Khaled

Submitted by

Name	Sec	BN
Mennatallah Ahmed Moustafa	2	25
Michael Ehab Mikhail	2	5

Contents

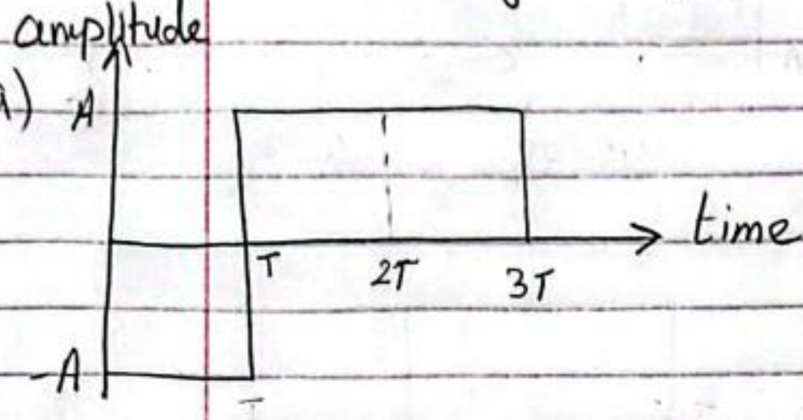
Figures

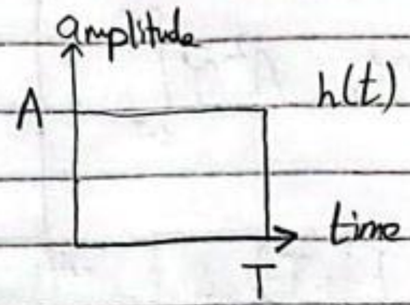
Part 1:

assuming '1' $\longrightarrow$ positive pulse

'0' $\longrightarrow$ negative pulse

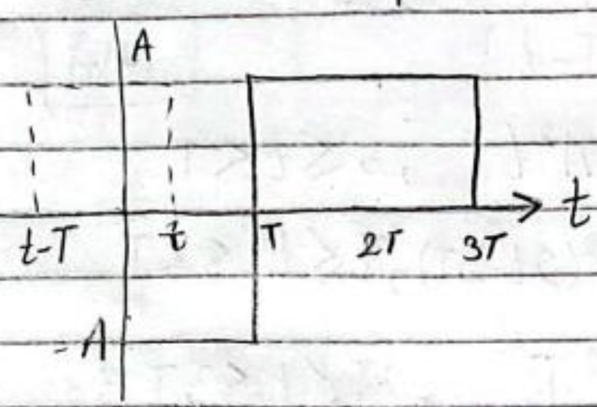
a)





b) $y(t) = s(t) \circledast h(t)$

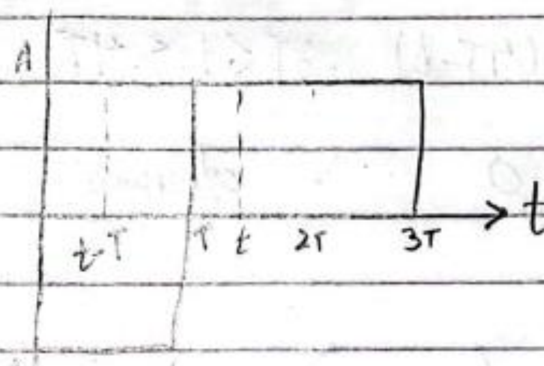
① $0 \leqslant t < T$



$$y(t) = \int_0^t -A \cdot A \, d\tau$$

$$= -A^2 \tau \Big|_0^t = -A^2 t$$

② $T \leqslant t < 2T$



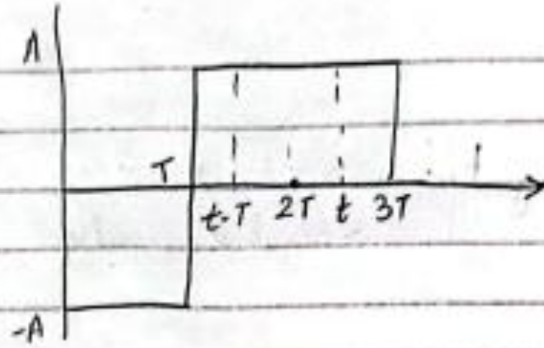
$$y(t) = \int_{t-T}^{T} -A \cdot A \, d\tau + \int_T^t A^2 \, d\tau$$

$$= -A^2 \tau \Big|_{t-T}^{T} + A^2 \tau \Big|_T^t$$

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

$$= A^2(2t - 3T)$$

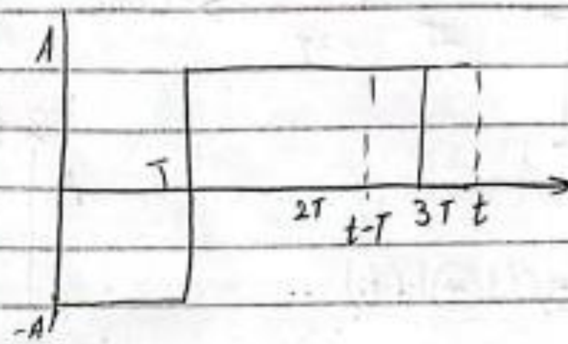
③ $2T \leq t < 3T$



$$y(t) = \int_{t-T}^{t} A^2 \, d\tau = A^2 \tau \Big|_{t-T}^{t}$$

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

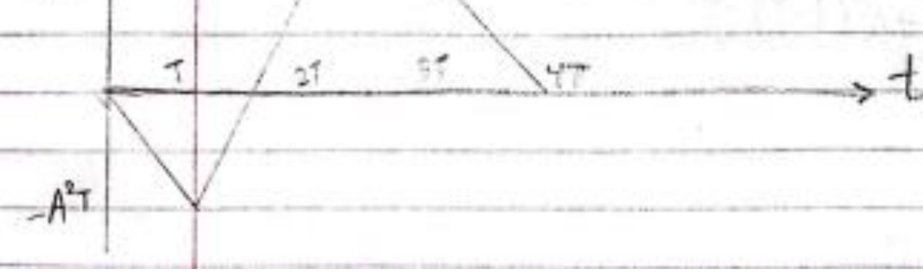
④ $3T \leq t < 4T$



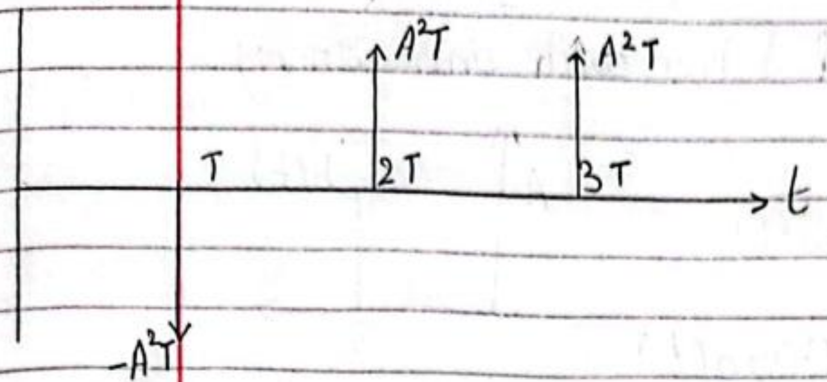
$$y(t) = \int_{t-T}^{3T} A^2 \, d\tau = A^2 \tau \Big|_{t-T}^{3T}$$

$$= A^2(3T - t + T)$$
$$= A^2(4T - t)$$

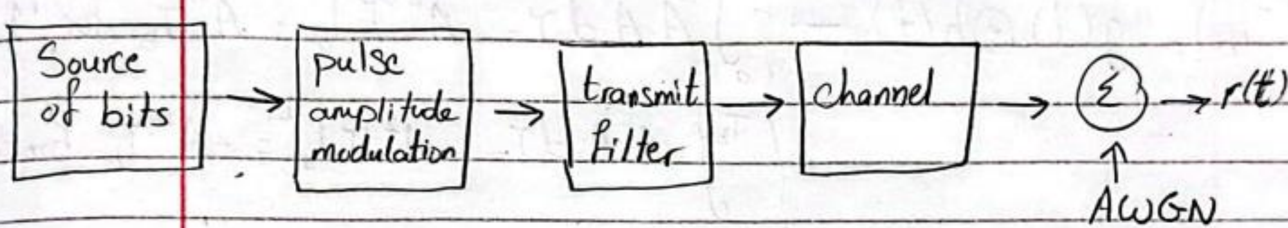
$$y(t) = \begin{cases} -A^2 t & , 0 \leq t < T \\ A^2(2t - 3T) & , T \leq t < 2T \\ A^2 T & , 2T \leq t < 3T \\ A^2(4T - t) & , 3T \leq t < 4T \\ 0 & , \text{otherwise} \end{cases}$$



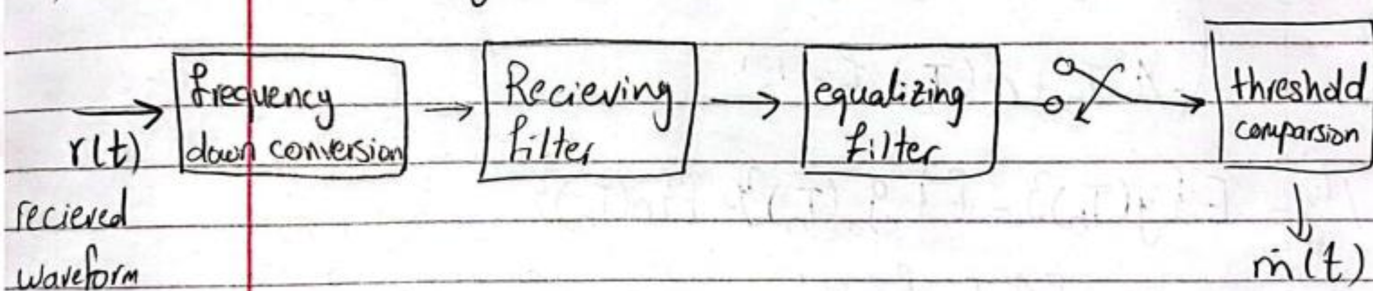
c) Signal will be sampled at T, 2T, 3T



d) Transmitter block diagram:



e) Reciever block diagram:



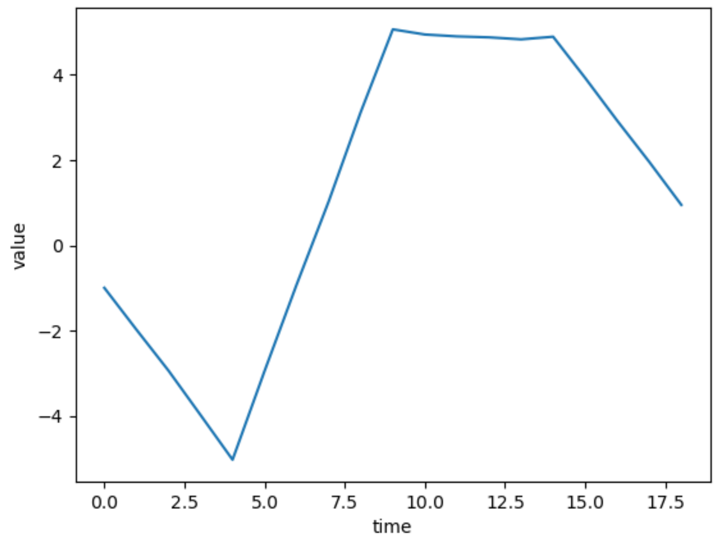


Figure 1: Output of matched filter with input 011

- The result obtained from hand analysis (Page 5) is matching what we obtained from simulating the same input.

```
# Matched Filter (Part 1)
binary_source = np.array([0,1,1])
pulse_signal1, filtered_signal1, decoded_signal1, sim_error1, th_error1 = system(binary_source, receive_filter1)
plot_signals(pulse_signal1, filtered_signal1, decoded_signal1)
print(decoded_signal1)
```
✓ 0.4s

```
[0, 1, 1]
```

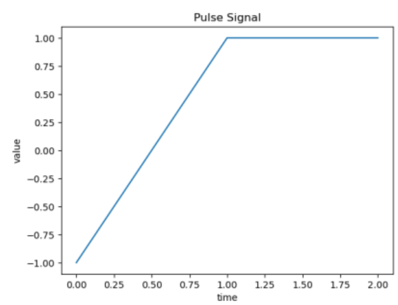


Figure 2: Input (011) after pulse block of system

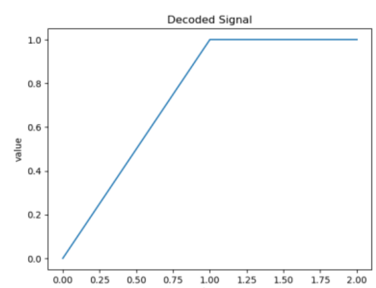
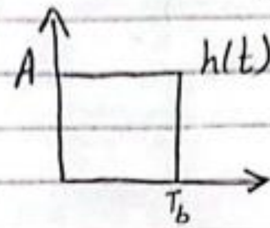


Figure 3: Decoded values (0s and 1s)

Part 2: Theoretical

a) $h(t)$ is matched filter with unit energy

$y(t) = r(t) \circledast h(t)$
where $r(t) = g(t) + w(t)$



$y(t) = g(t) \circledast h(t) + w(t) \circledast h(t)$

$$g(t) = \begin{cases} A & \text{for '1'} \\ -A & \text{for '0'} \end{cases}$$

$$g(T_b) = g(t) \circledast h(t) = \begin{cases} \int_0^{T_b} A.A\, d\tau = A^2 \tau \Big|_0^{T_b} = A^2 T_b & \text{for '1'} \\ \int_0^{T_b} -A.A\, d\tau = -A^2 \tau \Big|_0^{T_b} = -A^2 T_b & \text{for '0'} \end{cases}$$

$$y(T_b) = \begin{cases} A^2 T_b + n(T_b) & \text{for '1'} \\ -A^2 T_b + n(T_b) & \text{for '0'} \end{cases}$$

$\mu_y = E\{y(T_b)\} = E\{g_o(T_b)\} + E\{n(T_b)\}$

$$E\{g_o(T_b)\} = \begin{cases} A^2 T_b & \text{for '1'} \\ -A^2 T_b & \text{for '0'} \end{cases}$$

$E\{n(T_b)\} = E\{\int w(\tau) h(T_b - \tau) d\tau\} = E\{\int w(\tau) g(\tau) d\tau\}$
$= \int_0^1 E\{w(\tau)\} d\tau = 0$

$$\mu_y = \begin{cases} A^2 T_b & \text{for '1'} \\ -A^2 T_b & \text{for '0'} \end{cases}$$

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

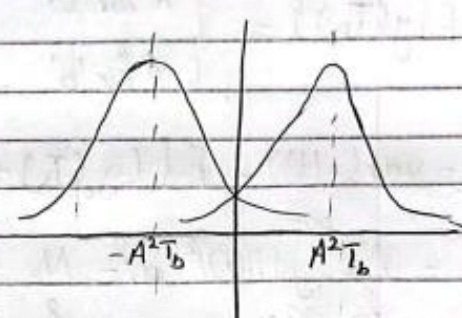
$$= E\{(n(T_b))^2\} = \frac{N_o}{2}\int_{-\infty}^{\infty} |H(f)|^2 df$$

$$= \frac{N_o}{2}\int_0^1 |h(t)|^2 dt = \frac{N_o}{2} A^2 T_b$$

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

$$p(y|'0') = \frac{1}{\sqrt{2\pi \frac{N_o}{2} A^2 T_b}} e^{-\frac{(y+A^2T_b)^2}{2\frac{N_o}{2}A^2T_b}}$$

$$p(y|'1') = \frac{1}{\sqrt{2\pi \frac{N_o}{2} A^2 T_b}} e^{-\frac{(y-A^2T_b)^2}{2\frac{N_o}{2}A^2T_b}}$$



assuming $P('0') = P('1') = 0.5$

$$P(e) = P(e|0)P(0) + P(e|1)P(1)$$

$$= P(e|0)$$

$$= \int_0^{\infty} P(y|'0')\,dy = \int_0^{\infty} \frac{1}{\sqrt{\pi N_o A^2 T_b}} e^{-\frac{(y+A^2T_b)^2}{N_o A^2 T_b}} dy$$

let $z = \dfrac{y + A^2 T_b}{\sqrt{N_o A^2 T_b}}$

$$dz = \frac{dy}{\sqrt{N_o A^2 T_b}}$$

at $y = 0$ $z = \sqrt{\dfrac{A^2 T_b}{N_o}}$, at $y = \infty$ $z = \infty$

$$P(e) = \int_{\sqrt{\frac{A^2T_b}{N_o}}}^{\infty} \frac{1}{\sqrt{\pi}} e^{-z^2} dz = \frac{1}{2}\,\mathrm{erfc}\left(\sqrt{\frac{A^2T_b}{N_o}}\right)$$

at $A = T = 1$, $P(e) = \frac{1}{2}\,\mathrm{erfc}\left(\frac{1}{\sqrt{N_o}}\right)$

b) $g(t) = \begin{cases} A & \text{for '1'} \\ -A & \text{for '0'} \end{cases}$

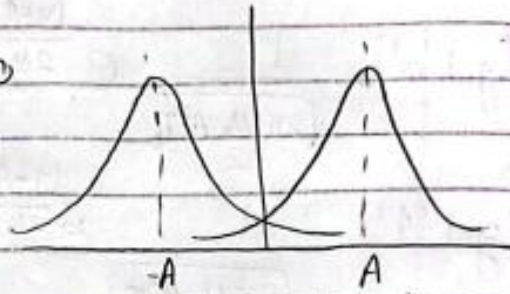
$h(t) = \delta(t)$

$y(t) = r(t) \circledast h(t) = r(t) \circledast \delta(t) = r(t) = g(t) + w(t)$

$y(T_b) = \begin{cases} A + w(t) & \text{for '1'} \\ -A + w(t) & \text{for '0'} \end{cases}$

as proven previously, $E\{w(t)\} = 0$

$\mu_y = E\{y(T_b)\} = \begin{cases} A & \text{for '1'} \\ -A & \text{for '0'} \end{cases}$



$\sigma_y^2 = var(y(t)) = E\{(g_0(T_b) + n(T_b) - \mu_y)^2\} = E\{(n(T_b))^2\}$

$= \frac{N_0}{2} \int_{-\infty}^{\infty} |H(f)|^2 \, df = \frac{N_0}{2}$

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

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

assuming $p('0') = p('1') = 0.5 \therefore p(e) = p(e|'0') = p(e|'1')$

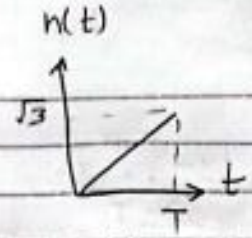
$p(e) = \int_{-\infty}^{0} p(y|'0') \, dy = \int_{-\infty}^{0} \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(y+A)^2}{N_0}} \, dy$

let $z = \frac{y+A}{\sqrt{N_0}}$ $\quad dz = \frac{dy}{\sqrt{N_0}}$, at $y = \infty$ $\quad z = \infty$

at $y = 0$ $\quad z = A/\sqrt{N_0}$

$p(e) = \frac{1}{\sqrt{\pi}} \int_{\frac{A}{\sqrt{N_0}}}^{\infty} e^{-z^2} \, dz = \frac{1}{2} erfc\left(\frac{A}{\sqrt{N_0}}\right)$, at $A = 1$ $\boxed{p(e) = \frac{1}{2} erfc\left(\frac{1}{\sqrt{N_0}}\right)}$

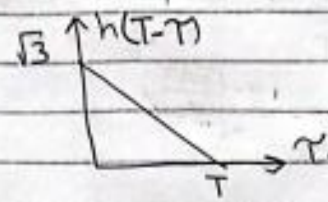
c) $g(t) = \begin{cases} A & \text{for '1'} \\ -A & \text{for '0'} \end{cases}$



$h(t) = \frac{\sqrt{3}\,t}{T}$

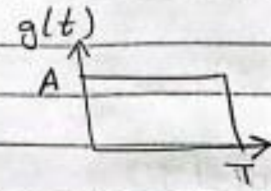
$y(t) = r(t) \circledast h(t)$

$= \underbrace{g(t) \circledast h(t)}_{g_o(t)} + \underbrace{w(t) \circledast h(t)}_{n(t)}$



for '1':

$g_o(T) = \int_{-\infty}^{\infty} g(\tau)\, h(T-\tau)\, d\tau$



$g_o(T) = \int_0^T A \frac{\sqrt{3}}{T} (T-\tau)\, d\tau = \frac{\sqrt{3}A}{T} \left(T\tau - \frac{\tau^2}{2}\right)\Big|_0^T = \frac{\sqrt{3}AT}{2}$

for '0'

$g_o(T) = \int_0^T -A \frac{\sqrt{3}}{T} (T-\tau)\, d\tau = \frac{-\sqrt{3}\,AT}{2}$

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

$E\{n(t)\} = E\left\{ \int_{-\infty}^{\infty} w(\tau)\, h(t-\tau)\, d\tau \right\}$

$= \int_{-\infty}^{\infty} \underbrace{E\{w(\tau)\}}_{zero}\, h(t-\tau)\, d\tau = 0$

$\mu_y = E\{y(T_b)\} = \begin{cases} \frac{\sqrt{3}AT}{2} & \text{for '1'} \\ \frac{-\sqrt{3}AT}{2} & \text{for '0'} \end{cases}$

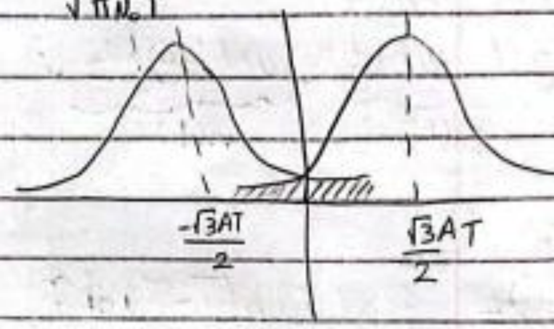
$\sigma_y^2 = E\{(n(T_b))^2\} = \frac{N_o}{2} \int_0^T |h(t)|^2\, dt = \frac{N_o}{2} \int_0^T \left(\frac{\sqrt{3}\,t}{T}\right)^2 dt$

$= \frac{3N_o}{2T^2} \frac{t^3}{3}\Big|_0^T = \frac{TN_o}{2}$

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

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

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



assuming $p('0') = p('1') = 0.5$

$\therefore p(e) = p(e|'0') = p(e|'1')$

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

let $z = \dfrac{y+\frac{\sqrt{3}AT}{2}}{\sqrt{N_0 T}}$, $dz = \dfrac{dy}{\sqrt{N_0 T}}$ at $y=0$ $z = \dfrac{\sqrt{3}AT}{2\sqrt{N_0 T}}$, $y=\infty$ $z=\infty$

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

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

at $T=1$, $A=1$

$$p(e) = \frac{1}{2} \operatorname{erfc}\left(\frac{\sqrt{3}}{2\sqrt{N_0}}\right)$$

Part 2: Simulation

Plot the output of the receive filter for the three mentioned cases

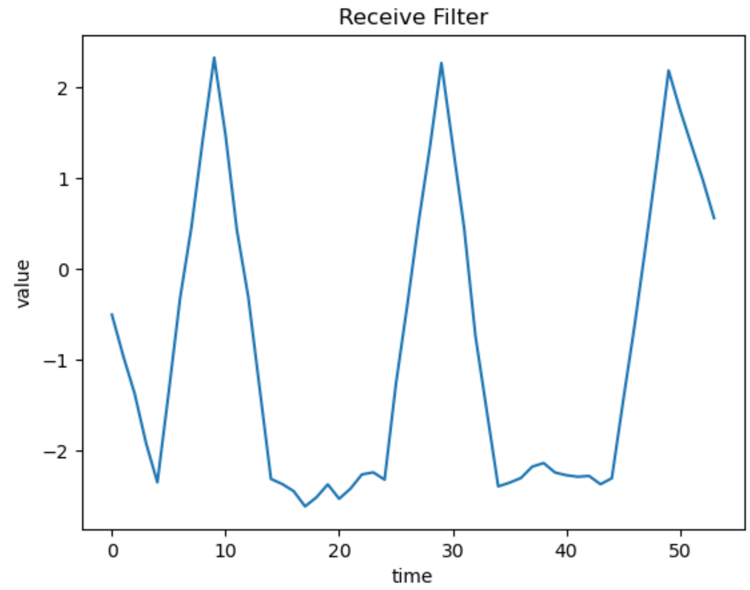


Figure 4: Output of Receive Filter 1 (matched filter with unit energy)

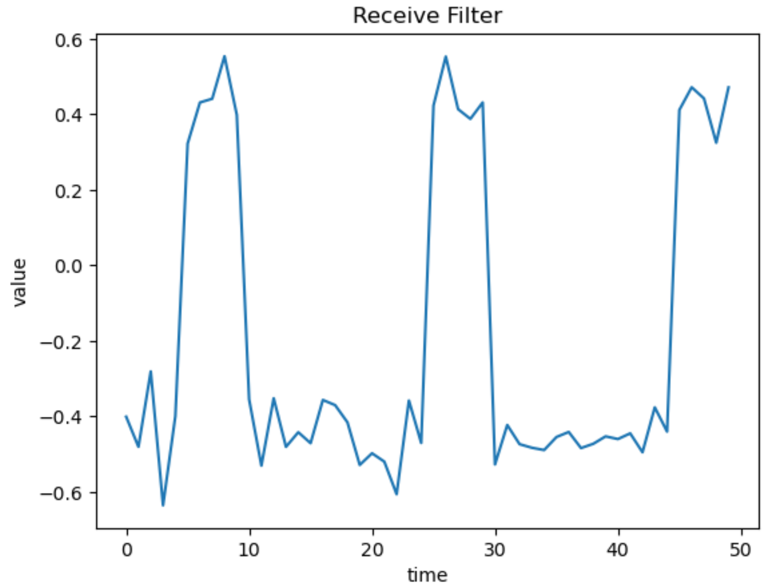


Figure 5: Output of Receive Filter 2 (not existent (i.e. h(t) = δ(t)))

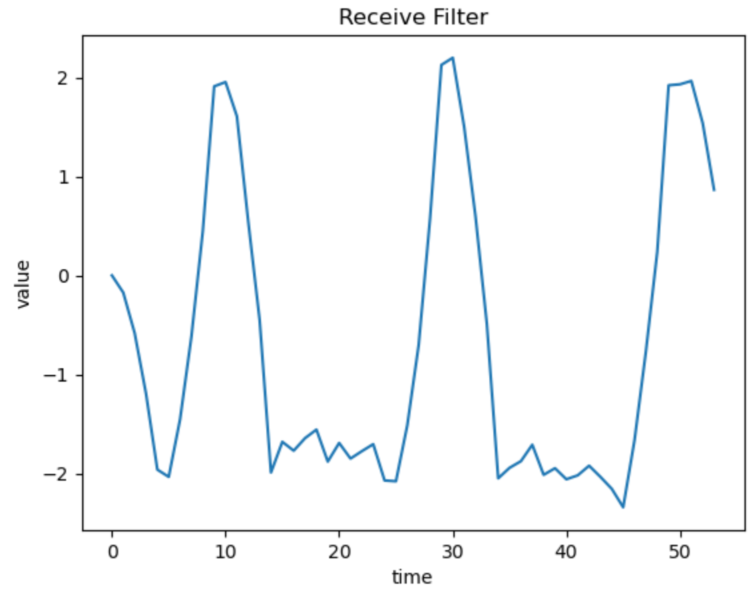


Figure 6: Output of Receive Filter 3 (Specific impulse response)

Plot the theoretical and simulated Bit Error Rate (BER) Vs E/No

Q: On the same figure, plot the theoretical and simulated Bit Error Rate (BER) Vs E/No (where E is the average symbol energy) for the three mentioned cases. Take E/No to be in the range -10 dB: 20:dB. (Use a semilogy plot)

Using number_of_bits = 10^5 and number_of_bit_samples = 10

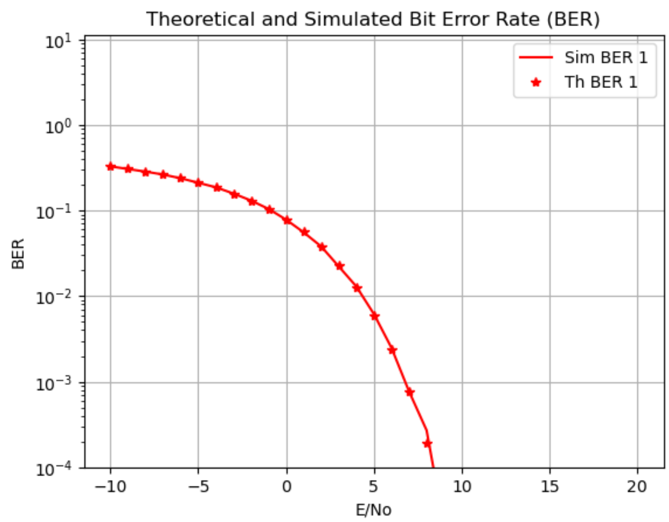


Figure 7: Theoretical and Simulated BER (matched filter with unit energy)

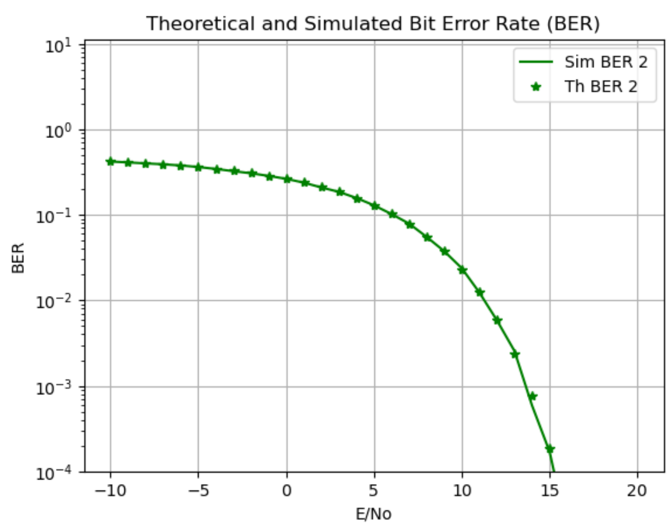


Figure 8: Theoretical and Simulated BER (not existent (i.e. h(t) = δ(t)))

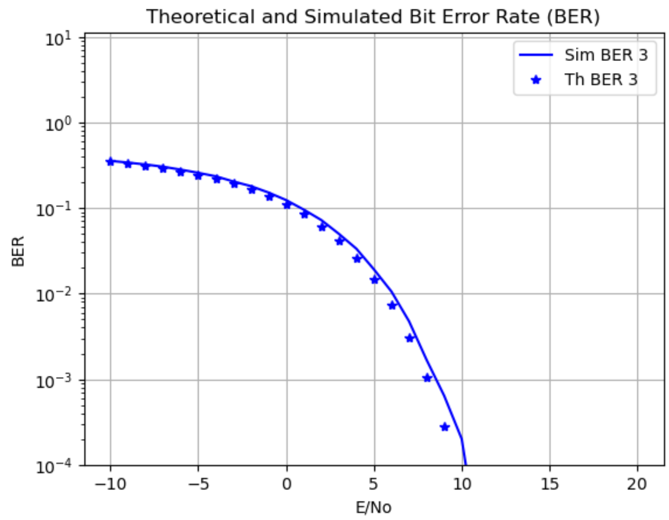


Figure 9: Theoretical and Simulated BER (Specific impulse response)

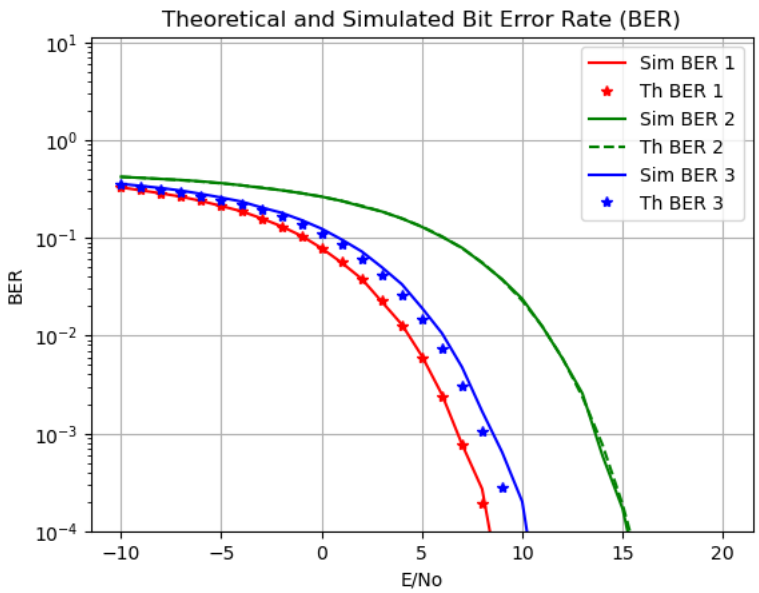


Figure 10: Theoretical and Simulated Bit Error Rate (BER)

As we can see the theoretical BER for filter 1 is not equal to the theoretical BER for filter 2 due to normalization

Average Simulation Bit Error Rate for filter 1 = 0.07844354838709677

Average Simulation Bit Error Rate for filter 2 = 0.16552774193548386

Average Simulation Bit Error Rate for filter 3 = 0.09837322580645162

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

Yes, the Bit Error Rate (BER) is a decreasing function of E/No.

Because increasing E/No means increasing the ratio of transmitted signal energy (E) to noise energy (No). This leads to a narrower Gaussian noise distribution (Sigma = No /2 and No decreases so sigma decreases as well), making it less likely for noise to corrupt the input signal. With less noise affecting the transmitted bits, it becomes easier for the receiver side to recover them correctly (Decreased BER)

But the relationship between E/No and BER is not linear. This means that even small improvements in E/No can significantly reduce the number of bit errors (BER).

Which case has the lowest BER? Why?

The receive filter ($\boldsymbol{t}$) is a matched filter with unit energy has the lowest BER.

A matched filter is designed to align with the characteristics of the transmitted signal.

During the matching, high values in the received signal are multiplied by high values in the filter, further emphasizing the desired signal. Same with low values in the signal which are also multiplied by low filter values, minimizing their impact.

This process amplifies the intended signal compared to the noise which leads to an easier distinction between high and low values during the thresholding step, leading to more accurate detection of transmitted bits and a lower Bit Error Rate (BER).

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

number_of_bit_samples = 5

receive_filter1 = np.ones(number_of_bit_samples)
receive_filter2 = None
receive_filter3 = np.linspace(0, np.sqrt(3),
number_of_bit_samples)

def generate_binary_source(num_bits):
    np.random.seed(15)
    return np.random.choice([0, 1], size=num_bits)

def pulse_shape(input_signal):
    def pulse(bit):
        return 1 if bit == 1 else -1

    pulse_array = np.array([pulse(bit) for bit in
input_signal])

    return pulse_array

def AWGN(No, num_bits, num_bit_samples, input_signal):
    total_size = num_bit_samples * num_bits
    noise = np.random.normal(scale=math.sqrt(No/2),
size=total_size)

    # Upsampling
    scaling_factor = 1.0 / np.sqrt(num_bit_samples)
```

```
    repeated_elements = np.repeat(input_signal,
num_bit_samples)
    modified_samples = repeated_elements * scaling_factor
    return (modified_samples + noise)

def receive_filter(input_signal, filter):
    if filter is not None:
        return np.convolve(input_signal, filter)
    else:
        return input_signal

def sample(input_signal, num_bits, num_samples):
    sampled_values = np.zeros(num_bits)

    for i in range(num_bits):
        sampled_index = i * num_samples + (num_samples - 1)
        sampled_values[i] = input_signal[sampled_index]

    return sampled_values

def decode(input_signal):
    res = []
    for i in range(len(input_signal)):
        if input_signal[i] <= 0:
            res.append(0)
        else:
            res.append(1)
    return res

def calculate_simulation_error(signal, decoded_signal):
    return (np.sum(signal != decoded_signal) / len(signal))
```

```
def calculate_theoretical_error(No, th_err_coeff,
filter_type):
    if filter_type == 2:
        return 0.5 *
math.erfc((1/math.sqrt(number_of_bit_samples)) * (1 /
math.sqrt(No)))
    return 0.5 * math.erfc(th_err_coeff * (1 /
math.sqrt(No)))

def calculate_errors(pulse_signal, decoded_signal,
th_err_coeff, No, filter_type):
    sim_err = calculate_simulation_error(pulse_signal,
decoded_signal)
    th_err = calculate_theoretical_error(No, th_err_coeff,
filter_type)
    return sim_err, th_err

def system(input_signal, filter, filter_type,
th_err_coeff=1):
    pulse_signal = pulse_shape(input_signal)
    num_bits = input_signal.shape[0]
    sim_error, th_error = [], []
    # E/No to be in the range -10 dB: 20:dB.
    for E_No_db in range(-10, 21):
        E_No = 10 ** (E_No_db / 10)
        E = 1
        No = E/(E_No)
        channel_signal = AWGN(No, num_bits,
number_of_bit_samples, pulse_signal)
        filtered_signal = receive_filter(channel_signal,
filter)
```

```
        sampled_filtered_signal = sample(filtered_signal,
num_bits, number_of_bit_samples)
        decoded_signal = decode(sampled_filtered_signal)
        x, y = calculate_errors(input_signal, decoded_signal,
th_err_coeff, No, filter_type)
        sim_error.append(x)
        th_error.append(y)

    return pulse_signal, filtered_signal, decoded_signal,
sim_error, th_error

def plot_signals(pulse_signal, filtered_signal,
decoded_signal):
    plt.figure()
    plt.plot(range(0,len(pulse_signal)), pulse_signal)
    plt.title("Pulse Signal")
    plt.xlabel('time')
    plt.ylabel('value')
    plt.figure()
    plt.plot(range(0,len(filtered_signal)), filtered_signal)
    plt.title("Receive Filter")
    plt.xlabel('time')
    plt.ylabel('value')
    plt.figure()
    plt.plot(range(0,len(decoded_signal)), decoded_signal)
    plt.title("Decoded Signal")
    plt.xlabel('time')
    plt.ylabel('value')

number_of_bits = 3
number_of_bit_samples = 5
total_bits_size = number_of_bits * number_of_bit_samples
```

```
# Matched Filter (Part 1)
binary_source = np.array([0,1,1])
pulse_signal1, filtered_signal1, decoded_signal1, sim_error1,
th_error1 = system(binary_source, receive_filter1, 1)
plot_signals(pulse_signal1, filtered_signal1,
decoded_signal1)
print(decoded_signal1)

number_of_bits = 10
number_of_bit_samples = 5
total_bits_size = number_of_bits * number_of_bit_samples
binary_source = generate_binary_source(number_of_bits)
pulse_signal1, filtered_signal1, decoded_signal1, _, _ =
system(binary_source, receive_filter1, 1)
pulse_signal2, filtered_signal2, decoded_signal2, _, _ =
system(binary_source, receive_filter2, 2)
pulse_signal3, filtered_signal3, decoded_signal3, _, _ =
system(binary_source, receive_filter3, 3)

plot_signals(pulse_signal1, filtered_signal1,
decoded_signal1)
plot_signals(pulse_signal2, filtered_signal2,
decoded_signal2)
plot_signals(pulse_signal3, filtered_signal3,
decoded_signal3)

sim_error1, th_error1, sim_error2, th_error2, sim_error3,
th_error3 = [], [], [], [], [], []
number_of_bits = 10**5
binary_source = generate_binary_source(number_of_bits)
_, _, _, sim_error1, th_error1 = system(binary_source,
receive_filter1, 1)
```

```
_, _, _, sim_error2, th_error2 = system(binary_source,
receive_filter2, 2)
_, _, _, sim_error3, th_error3 = system(binary_source,
receive_filter3, 3, (math.sqrt(3)/2))

E_No_range = range(-10, 21)
plt.figure()
plt.title('Theoretical and Simulated Bit Error Rate (BER)')
plt.xlabel('E/No')
plt.ylabel('BER')
plt.yscale('log') # dB
plt.ylim(1e-4)
plt.semilogy(E_No_range, sim_error1, 'r-')
plt.semilogy(E_No_range, th_error1, 'r*')

plt.legend(['Sim BER 1', 'Th BER 1'])
plt.grid()
plt.show()

E_No_range = range(-10, 21)
plt.figure()
plt.title('Theoretical and Simulated Bit Error Rate (BER)')
plt.xlabel('E/No')
plt.ylabel('BER')
plt.yscale('log') # dB
plt.ylim(1e-4)
plt.semilogy(E_No_range, sim_error2, 'g-')
plt.semilogy(E_No_range, th_error2, 'g*')

plt.legend(['Sim BER 2', 'Th BER 2'])
plt.grid()
plt.show()
```

```
E_No_range = range(-10, 21)
plt.figure()
plt.title('Theoretical and Simulated Bit Error Rate (BER)')
plt.xlabel('E/No')
plt.ylabel('BER')
plt.yscale('log') # dB
plt.ylim(1e-4)
plt.semilogy(E_No_range, sim_error2, 'b-')
plt.semilogy(E_No_range, th_error2, 'b*')

plt.legend(['Sim BER 3', 'Th BER 3'])
plt.grid()
plt.show()

E_No_range = range(-10, 21)
plt.figure()
plt.title('Theoretical and Simulated Bit Error Rate (BER)')
plt.xlabel('E/No')
plt.ylabel('BER')
plt.yscale('log') # dB
plt.ylim(1e-4)
plt.semilogy(E_No_range, sim_error1, 'r-')
plt.semilogy(E_No_range, th_error1, 'r*')
plt.semilogy(E_No_range, sim_error2, 'g-')
plt.semilogy(E_No_range, th_error2, 'g--')
plt.semilogy(E_No_range, sim_error3, 'b-')
plt.semilogy(E_No_range, th_error3, 'b*')
plt.legend(['Sim BER 1', 'Th BER 1', 'Sim BER 2', 'Th BER 2',
'Sim BER 3', 'Th BER 3'])
plt.grid()
plt.show()
```

```
print("Average Simulation Bit Error Rate for filter 1 = {}".format(np.mean(sim_error1)))
print("Average Simulation Bit Error Rate for filter 2 = {}".format(np.mean(sim_error2)))
print("Average Simulation Bit Error Rate for filter 3 = {}".format(np.mean(sim_error3))
```