Cairo University
Faculty of Engineering

Department of Computer Engineering

ELC 325B – Spring 2023

Digital Communications

Assignment #2

Filters

Submitted to

Dr. Mai

Dr.Hala

Eng.Mohamed Khaled

Submitted by

Name	Sec	BN
Sara Bisheer Fekry	1	18
Menna Mohamed Abdelbaset	2	26

Contents

Part I: Solve the following question: 3

Hand Analysis 3

Part II: Simulation: 6

Hand Analysis 6

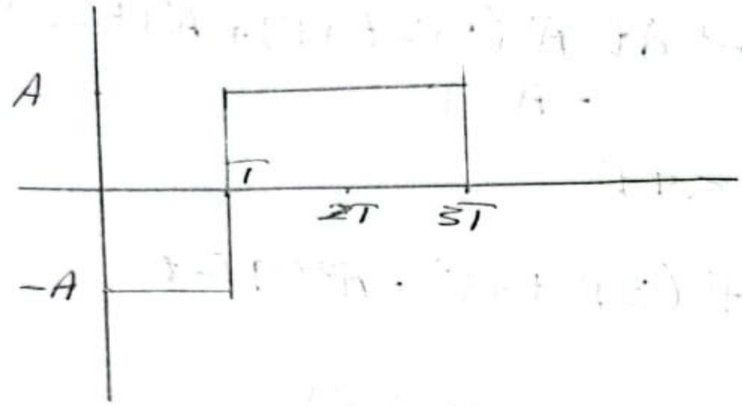
Figures With Comment 12

Part I: Solve the following question:

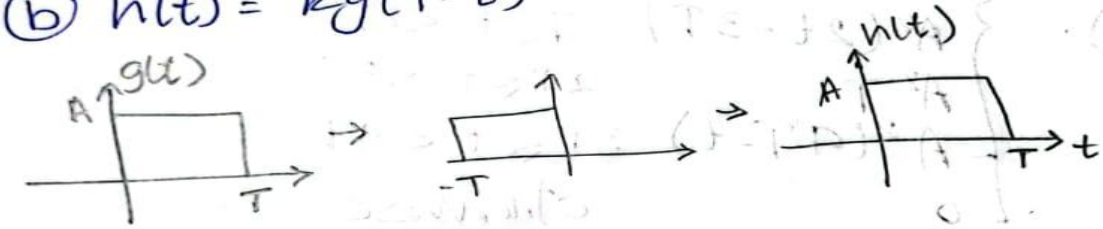
Hand Analysis

1. Part 1:-

(a)



(b) $h(t) = kg(T-t)$



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

$= h(t) * s(t) + \cancel{h(t) * w(t)}$ ignore noise

- Case I: $0 \leqslant t \leqslant T$

$$\int_0^t A(-A)\,dt = -A^2(t)$$

- Case II: $T \leqslant t \leqslant 2T$

$$\int_{t-T}^{T} -A^2\,dt + \int_T^t A^2\,dt = -A^2(2T-t) + A^2(t-T)$$
$$= A^2(-3T+2t)$$

2

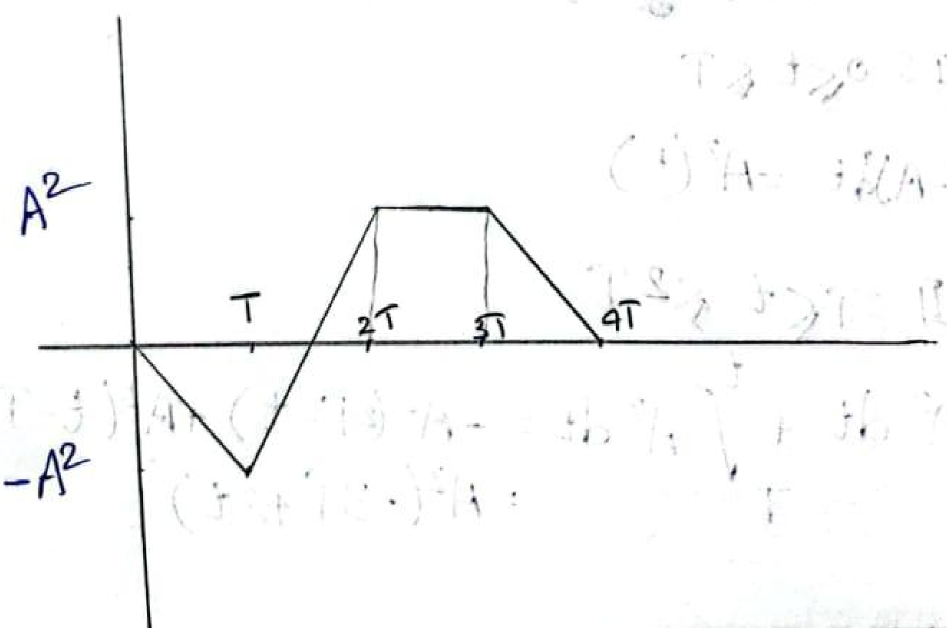
Case III: $2T \leq t < 3T$

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

Case IV: $3T \leq t < 4T$

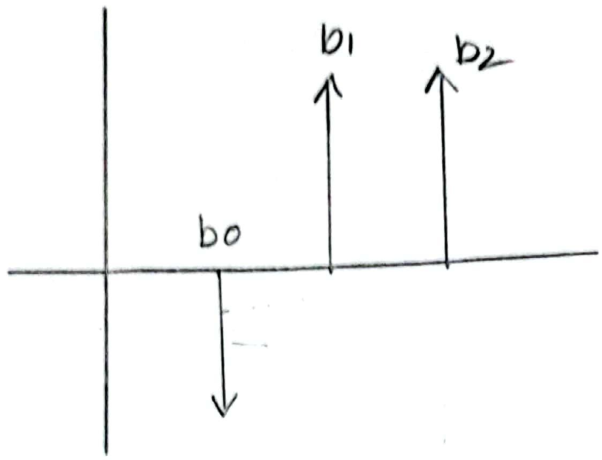
$$\int_{t-T}^{3T} A^2\,dt = A^2(3T - t + T) = A^2(4T - t)$$

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

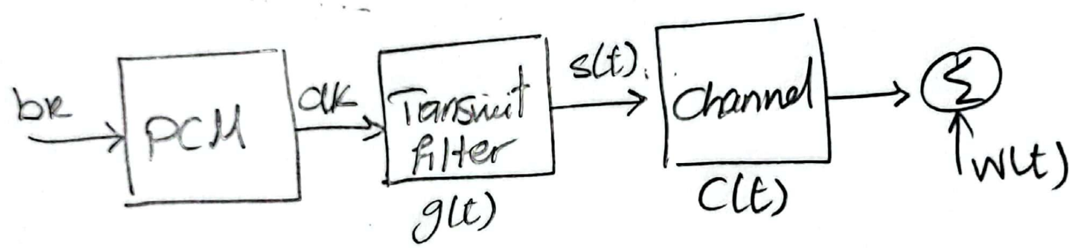


3

(c)

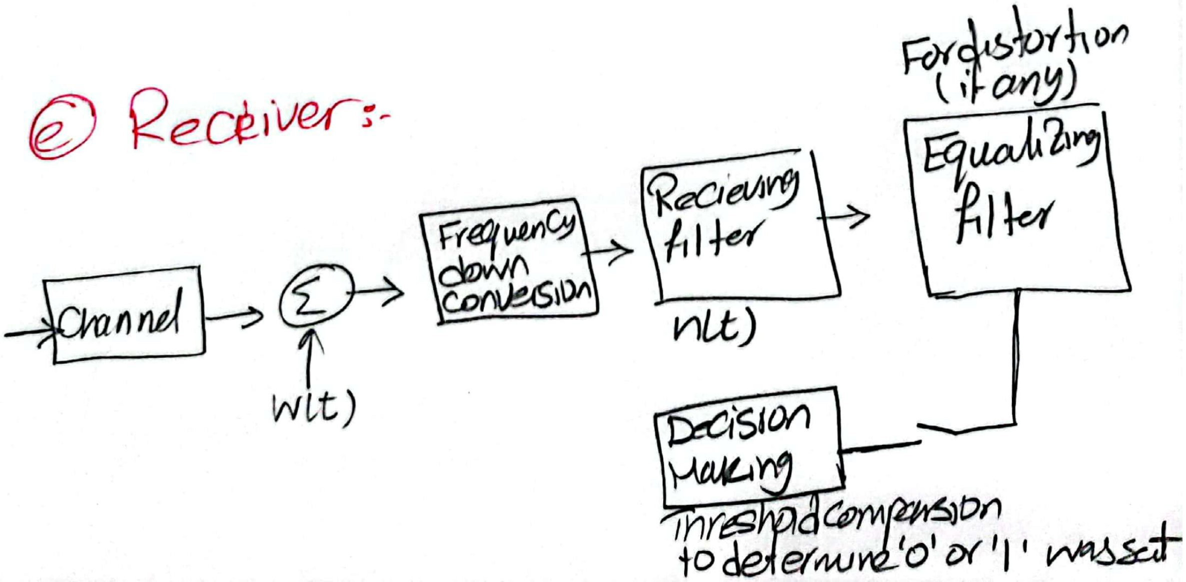


(d) Transmitter



$$s(t) = \sum a_k\, g(t - kT_b)$$

(e) Receiver:-

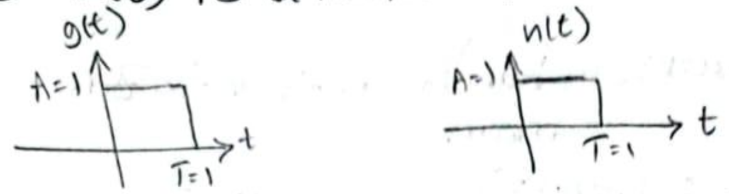


Part II: Simulation:

Hand Analysis

* Analysis for part II

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



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

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

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

$$g_o(t) = g(t) * h(t) = \int g(t) \underbrace{h(t-z)}_{kg(T-t+z)} dz = k\int g(t) g(T-t+z)\, dz$$

$$g_o(T_p)\Big|_0 = k \int_0^{T_p} A g(T_p)\, g(T_p)\, dz = k \int_0^{T_p} A \cdot -A\, dt = -kA^2 T_p$$

$$g_o(T_p)\Big|_1 = kA^2 T_p$$

$$y(T_p) = \begin{cases} -kA^2 T_p + n(t_p) & \text{when '0' is sent} \\ kA^2 T_p + n(t_p) & \text{when '1' is sent} \end{cases}$$

* We can make $k = \frac{1}{T_p A}$

$$y(T_p) = \begin{cases} -A + n(t_p) & \rightarrow \text{when '0' is sent} \\ A + n(t_p) & \rightarrow \text{when '1' is sent} \end{cases}$$

Scanned with CamScanner

2

$n(t) = h(t) * w(t)$

$n(t)$ is Gaussian Random noise with zero mean

$$\mathrm{Var}(n(T_p)) = E[n^2[T_p]] - E^2[\underbrace{\cancel{n(T_p)}}_{0}]$$

$$E[n^2[T_p]] = \int G_n(f)\,df = \int |H(f)|^2 \underbrace{\cancel{G_w(f)}}_{\frac{N_0}{2}}\,df$$

'power'
average of energy

$$= \frac{N_0}{2}\underbrace{\int |H(f)|^2\,df}_{\frac{1}{T_p}\ (\text{as } E = A^2T)} = \frac{N_0}{2T_p}$$

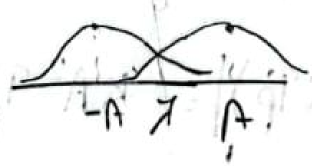
To clarify:

$$= \frac{N_0}{2}K^2 \int |G(f)|^2\,df\ T_p = \frac{N_0}{2T_p}$$

$$\cancel{s(T_p)} = n(T_p) \sim N\left(0, \frac{N_0}{2T_p}\right)$$

$$y(T_p) \sim N\left(\pm A, \frac{N_0}{2T_p}\right)$$

$$P(\text{error}) = P(y<\lambda|'1')\,P('1') + P(y>\lambda|'0')\,P('0')$$



$$P(y<\lambda|'1') = Q\left(\frac{\lambda + A}{\sqrt{N_0/2T_p}}\right)$$

$$P(y>\lambda|'0') = 1 - Q\left(\frac{\lambda - A}{\sqrt{N_0/2T_p}}\right) = Q\left(\frac{A-\lambda}{\sqrt{N_0/2T_p}}\right)$$

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

8

To Choose λ we can get $\frac{\partial P_{error}}{\partial \lambda} = 0$

then $\lambda_{opt} = \frac{N_0}{4AT_p} \ln\left(\frac{P('0')}{P('1')}\right)$

then

$\lambda_{opt} = 0$

$P(error) = \frac{1}{2} Q\left(\frac{A}{\sqrt{N_0/2T_p}}\right) + \frac{1}{2} Q\left(\frac{A}{\sqrt{N_0/2T_p}}\right)$

$\boxed{x}$ from Given $T = 1$, $A = 1$

$P(error) = Q\left(\frac{1}{\sqrt{N_0/2}}\right)$

$erfc(x) = 2Q(\sqrt{2}x)$

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

(b) $h(t)$ is not existent (ie $h(t) = \delta(t)$)

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

$= h(t) * g(t) + h(t) * w(t) = \delta(t) * g(t) + \delta(t) * w(t)$

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

$g(T_p) = \begin{cases} A & \text{when '1' is sent} \\ -A & \text{when '0' is sent} \end{cases}$

$y(T_p) = \begin{cases} A + w(t) & \text{when '1' is sent} \\ -A + w(t) & \text{when '0' is sent} \end{cases}$

4

$$w(t) \sim N(0, \frac{N_0}{2})$$

$$y(t) \sim (\pm A, \frac{N_0}{2})$$

$$P(\text{error}) = P(y<\lambda|'1')\, P('1') + P(y\cancel{>}\lambda|'0')\, P('0')$$

$$P(y<\lambda|'1') = Q\left(\frac{\lambda + A}{\sqrt{N_0/2}}\right)$$

$$P(y>\lambda|'0') = 1 - Q\left(\frac{\lambda - A}{\sqrt{N_0/2}}\right) = Q\left(\frac{A-\lambda}{\sqrt{N_0/2}}\right)$$

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

& $\lambda = 0$ as derived before in (a)

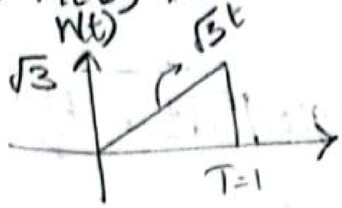
$$P(\text{error}) = Q\left(\frac{\overset{1\ (\text{given})}{A}}{\sqrt{N_0/2}}\right)$$

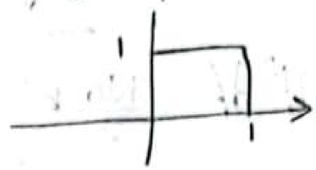
$$\text{erfc}(x) = 2Q(\sqrt{2}x)$$

$$Q\left(\sqrt{2}\frac{1}{\sqrt{N_0}}\right) = \frac{1}{2}\,\text{erfc}\left(\frac{1}{\sqrt{N_0}}\right)$$

$$P(\text{error}) = \frac{1}{2}\,\text{erfc}\left(\frac{1}{\sqrt{N_0}}\right)$$

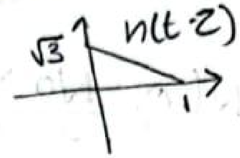
(c) $h(t)$ is the following impulse response





$$y(t) = \underbrace{h(t) * g(t)}_{g_0(t)} + h(t) * w(t)$$

$$g_0(t) = \int_0^1 h(t-z)\, g(z)\, dz$$



$$= \int_0^1 (-\sqrt{3}\, z + \sqrt{3})\, dz = \frac{\sqrt{3}}{2}$$

(or) we can make $g_0(t) = \int_0^1 h(t)\, g(t-z)\, dz)$

$$g_0(t) = \int_0^1 \sqrt{3}\, t\, dt = \frac{\sqrt{3}\, t^2}{2}\Big|_0^1 = \frac{\sqrt{3}}{2}$$

$$g_0(T_p) = \begin{cases} \frac{\sqrt{3}}{2} & \text{when '1' is sent} \\ -\frac{\sqrt{3}}{2} & \text{when '0' is sent} \end{cases}$$

$$y(T_p) = \begin{cases} \frac{\sqrt{3}}{2} + n(T_p) & \text{when '1' is sent} \\ -\frac{\sqrt{3}}{2} + n(T_p) & \text{when '0' is sent} \end{cases}$$

$$n(t) = h(t) * w(t)$$

$$w(t) \sim N(0, \frac{N_0}{2})$$

$$E[n[T_p]] = E\left[\int h(t-z)\, w(z)\, dz\right]$$

$$= \int h(t-z)\, E[w(z)]\, dz = 0$$

6

$$E[n^2(t)] = \int G_n(f)\,df = \int_{-\infty}^{\infty} \frac{N_0}{2} |H(f)|^2 df$$

$$= \frac{N_0}{2}\int_{-\infty}^{\infty} k^2 |G(f)|^2 df = \frac{N_0}{2} k^2 \int_{-\infty}^{\infty} |g(t)|^2 dt = \frac{N_0}{2}$$

$$Var(n(t)) = E[n^2(t)] - E^2(n(t))$$ ($E(n(t)) = 0$)

$$n(t) \sim N\left(0, \frac{N_0}{2}\right)$$

$$y(t) \sim N\left(\pm \frac{\sqrt{3}}{2}, \frac{N_0}{2}\right)$$

$$P(error) = P(y<\lambda \mid '1')P('1') + P(y>\lambda \mid '0')P('0')$$

$$P(y<\lambda \mid '1') = Q\left(\frac{\cancel{\lambda} + \frac{\sqrt{3}}{2}}{\sqrt{N_0/2}}\right)$$

$$P(y>\lambda \mid '0') = 1 - P(y<\lambda \mid '0') = 1 - Q\left(\frac{\lambda - \frac{\sqrt{3}}{2}}{\sqrt{N_0/2}}\right)$$

$$= Q\left(\frac{\frac{\sqrt{3}}{2} - \lambda}{\sqrt{N_0/2}}\right)$$

$$P(error) = Q\left(\frac{\frac{\sqrt{3}}{2}}{\sqrt{N_0/2}}\right)$$ $\lambda = 0$

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

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

$$P(error) = \frac{1}{2}\,erfc\left(\frac{\sqrt{3}/2}{\sqrt{N_0}}\right)$$

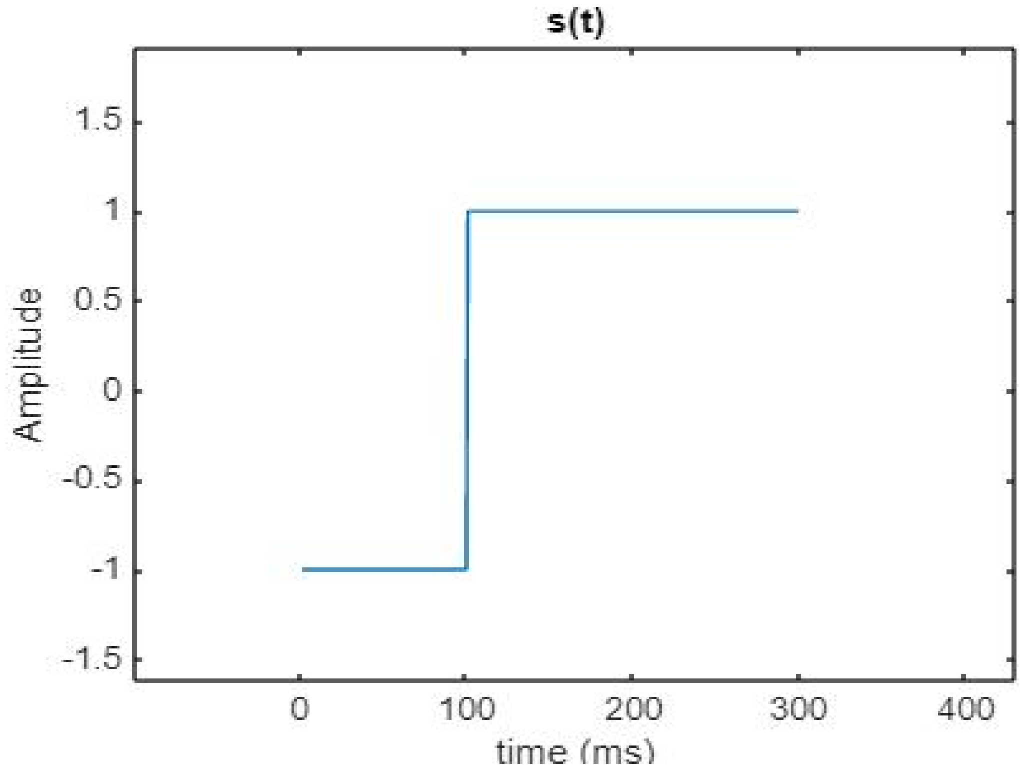
Figures With Comment

Input = [0 ,1 ,1]

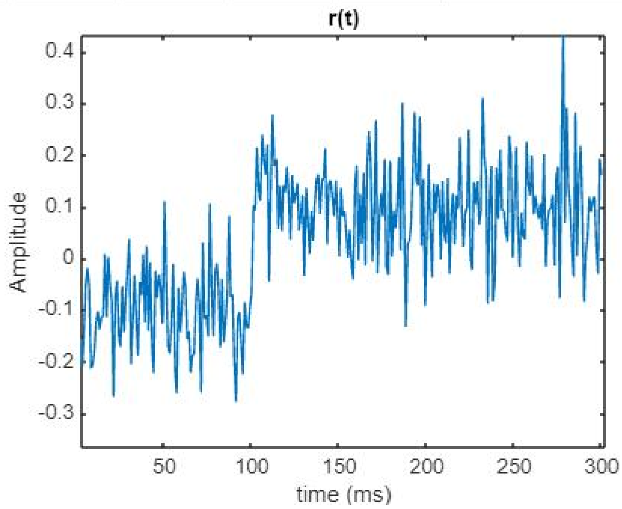
SNR = 20

Number of samples = 100

This figure represents the output of the binary source as a series of 0's and 1's



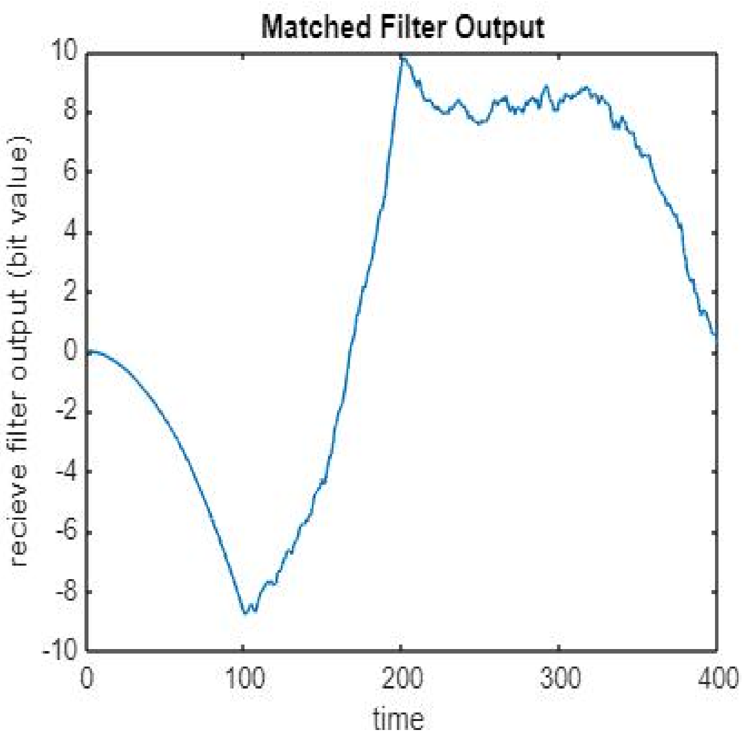
Note The x_axis is multiple of 100 as the number of samples is 100

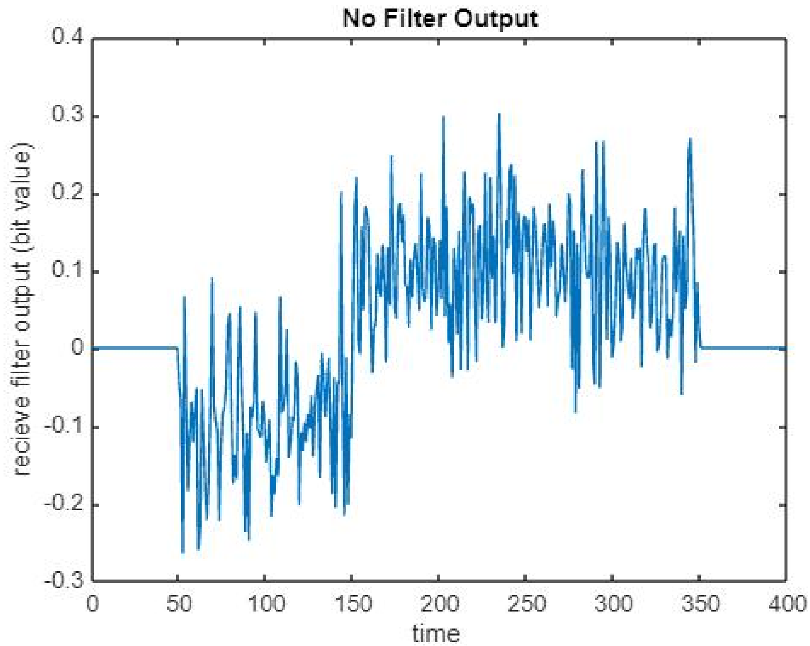


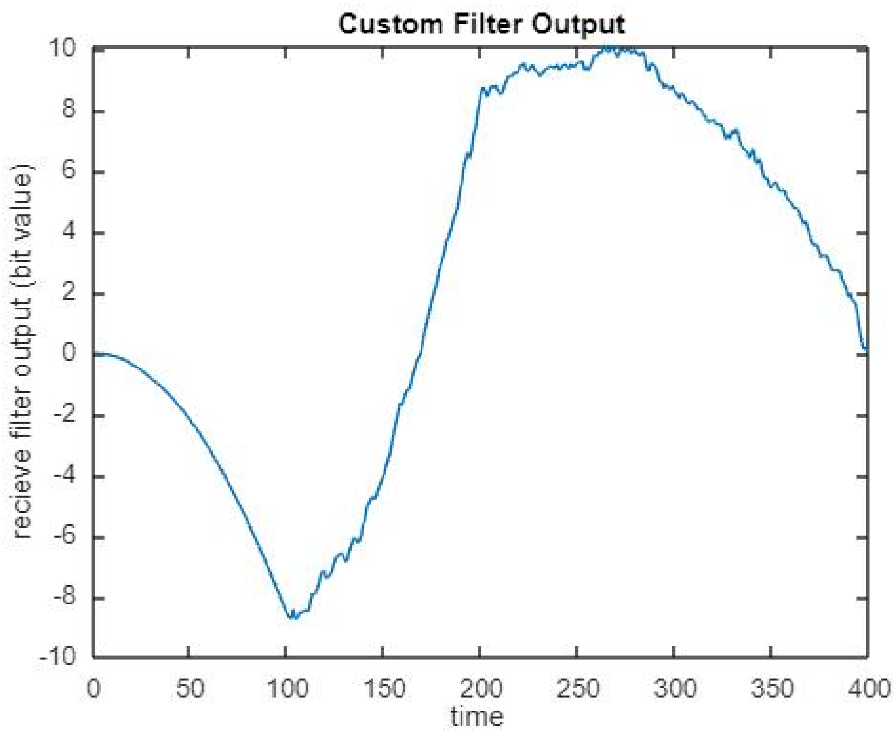
This figure represents the input after adding noise to it

Amplitude is divided by square root of sample number (sample_number = 100) for normalization the bit energy

Output After Filters







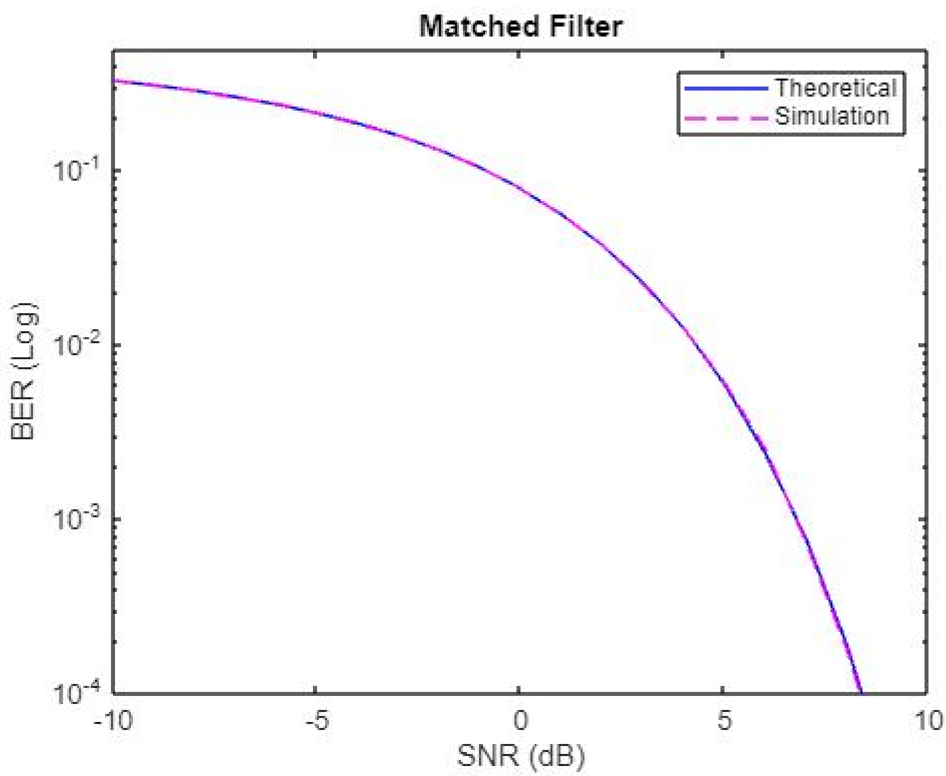
Comment

No filter has the highest effect of noise compared to the two other filters

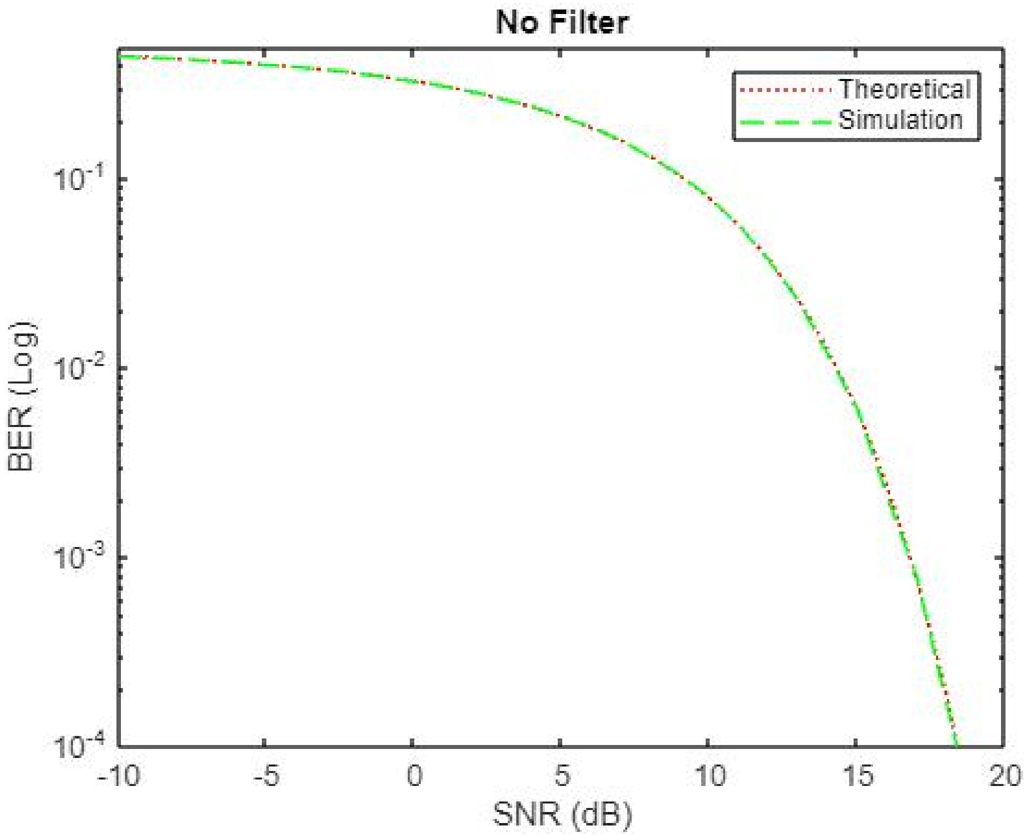
Output of the matched filter and this custom filter are smooth and when we increases the number of bit the matched filter appears to be more smooth slightly.

BER Theoretical and Simulation For The Three Filters

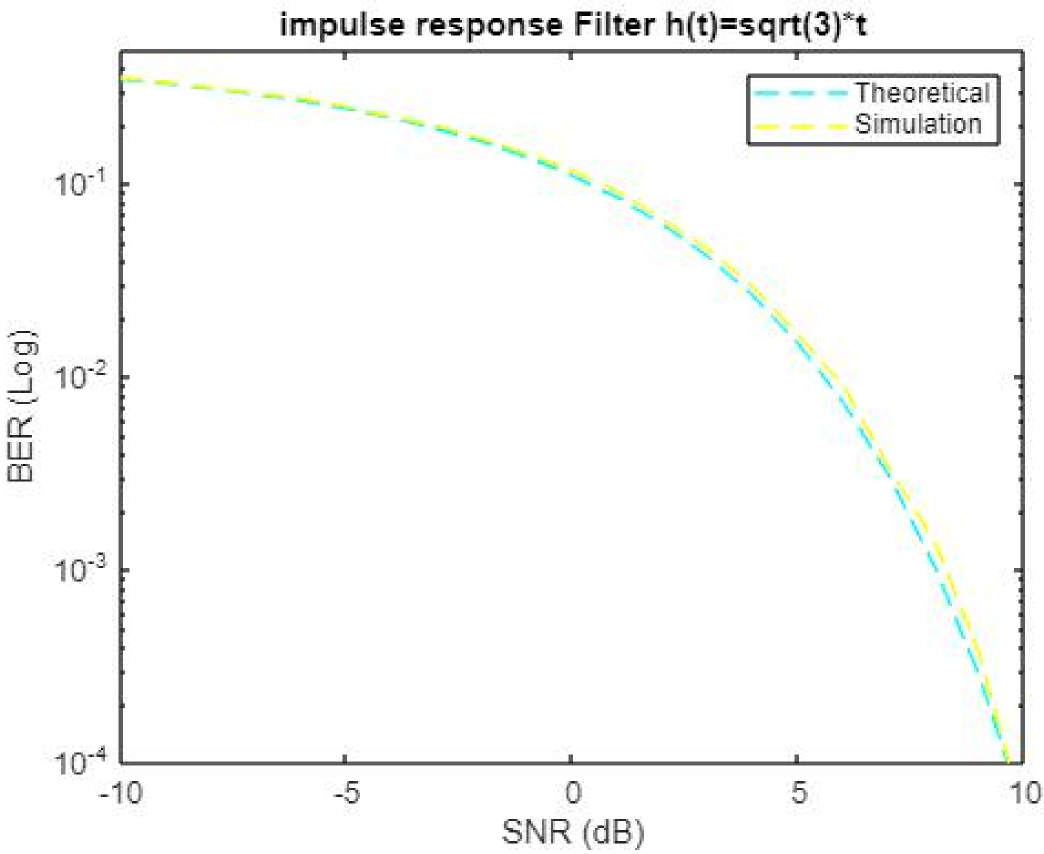
Matched Filter



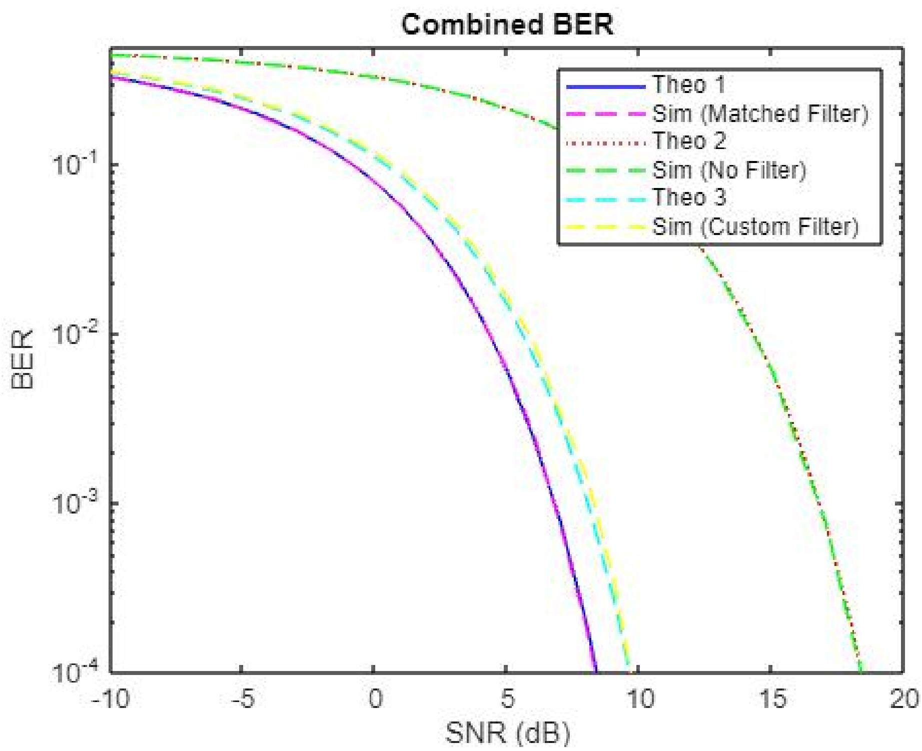
No Filter


No Filter
Theoretical
Simulation
BER (Log)
10^{-1}
10^{-2}
10^{-3}
10^{-4}
-10
-5
0
5
10
15
20
SNR (dB)

Custom Filter



Combined BER



Comment

After plotting theoretical and simulated BER , It shows that the simulation of each filter almost follows the theoretical of its filter

6) Is the BER an increasing or a decreasing function of E/N_o? Why?

Answer

It is a decreasing function as if the energy of the signal compared the energy of noise is high then the noise will not affect the signal so much and BER will decrease

so the more the E/N_o is the less the BER

6) Which case has the lowest BER? Why?

Answer

The first one with the matched filter.

The scenario with the matched filter achieves the lowest BER. This is because the matched filter is specifically designed to maximize the peak SNR of the signal.

Consequently, this minimizes the probability of errors, leading to a lower BER.

Code

```
clear;
close all;

samples_number = 100;
bits = [0,1,1,0,1,1,0,1];
bits_number = length(bits);

% Pulse Shape

[input] = pulse_shape(bits_number,samples_number,bits);

s = reshape(input.', [], 1);
figure;
plot(s);
title('s(t)');
xlabel('time (ms)');
ylabel('Amplitude');

% Channel AWGN
% Generate Noise To Add

E = 1;
snr_range = -10:1:20;
snr = 10 ^(snr_range(30)/10);

[input_with_noise] = add_noise(bits_number,samples_number,input,E,snr);

figure;
plot(input_with_noise);
title('r(t)');
xlabel('time (ms)');
ylabel('Amplitude');
```

```
% filters definations
delta_filter = zeros(1,samples_number);
delta_filter(samples_number/2)=1;
t = 0 : 1 : samples_number -1;
tri_filter = (sqrt(3)/samples_number)*t;
matched_filter = ones(1,samples_number);
filter ={matched_filter,delta_filter,tri_filter};

output = {0,0,0};


for k=1 : 3
output{k} = conv(input_with_noise,filter{k});
end




% show output of each filter

figure;
plot(output{1});
title('Matched Filter Output');
xlabel('time ');
ylabel('recieve filter output (bit value)');
hold on ;

figure;
plot(output{2});
title('No Filter Output');
xlabel('time ');
ylabel('recieve filter output (bit value)');
hold on ;
```

```
figure;
plot(output{3});
title('Custom Filter Output');
xlabel('time ');
ylabel('recieve filter output (bit value)');
hold on ;


% sample the output to get stream of bits
for i=0:bits_number-1
output_1_samples = sample(output{1},bits_number,samples_number);
output_2_samples = sample(output{2},bits_number,samples_number);
output_3_samples = sample(output{3},bits_number,samples_number);
end

% disp(output_1_samples)
% disp(output_2_samples)
% disp(output_3_samples)

% calculate accuracy of each filter
err_prob_1 = sum(output_1_samples ~= bits);
BER_1 = err_prob_1/bits_number;
err_prob_2 = sum(output_2_samples ~= bits);
BER_2 = err_prob_2/bits_number;
err_prob_3 = sum(output_3_samples ~= bits);
BER_3 = err_prob_3/bits_number;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculate BER for different SNR
bits_number = 100000;
samples_number = 10;
```

```
indices = randperm(bits_number, 1);

bits= ones(bits_number,1);

bits(indices)=0;


delta_filter = zeros(1,samples_number);

delta_filter(samples_number/2)=1;

t = 0 : 1 : samples_number -1;

tri_filter = (sqrt(3)/samples_number)*t;

matched_filter = ones(1,samples_number);

filter ={matched_filter,delta_filter,tri_filter};



input =pulse_shape(bits_number,samples_number,bits);

snr_range = -10:1:20;

% Preallocate arrays to store BER simulations

BER_sim_1 = zeros (length(snr_range),1);

BER_sim_2 = zeros (length(snr_range),1);

BER_sim_3 = zeros (length(snr_range),1);

BER_theo_1 = zeros(length(snr_range),1);

BER_theo_2 = zeros(length(snr_range),1);

BER_theo_3 = zeros(length(snr_range),1);


for i = 1:length(snr_range)

snr = 10 ^(snr_range(i)/10);


input_with_noise = add_noise(bits_number,samples_number,input,E,snr);


for k = 1:3

output{k} = conv(input_with_noise, filter{k}); % Consider using 'same' to maintain
dimensionality

end


% Extracting the middle point for each bit period after convolution

output_1_samples = sample(output{1},bits_number,samples_number);
```

```
output_2_samples = sample(output{2},bits_number,samples_number);
output_3_samples = sample(output{3},bits_number,samples_number);

% disp(size(bits));
% Calculate errors and BER for each filter
err_prob_1 = sum(output_1_samples.' ~= bits);
BER_sim_1(i) = err_prob_1 / bits_number;
err_prob_2 = sum(output_2_samples.' ~= bits);
BER_sim_2(i) = err_prob_2 / bits_number;
err_prob_3 = sum(output_3_samples.' ~= bits);
% disp(BER_sim_1)
BER_sim_3(i) = err_prob_3 / bits_number;
BER_theo_1(i)=0.5*erfc(sqrt(snr));
BER_theo_2(i)=0.5*erfc(sqrt(snr/samples_number));
BER_theo_3(i)=0.5*erfc(((sqrt(3)/(2))*sqrt(snr)));
end

% Update plot commands to reflect all data
figure;
semilogy(snr_range, BER_theo_1, 'b-');
hold on;
semilogy(snr_range, BER_sim_1, 'm--');
semilogy(snr_range, BER_theo_2, 'r:');
semilogy(snr_range, BER_sim_2, 'g--');
semilogy(snr_range, BER_theo_3, 'c--');
semilogy(snr_range, BER_sim_3, 'y--');
hold off;

ylim([10^-4 0.5]);
xlabel('SNR (dB)');
ylabel('BER');
legend('Theo 1', 'Sim (Matched Filter)', 'Theo 2', 'Sim (No Filter)', 'Theo 3', 'Sim (Custom Filter)');
title('Combined BER');
```

```
figure;
semilogy(snr_range, BER_theo_1, 'b-');
hold on;
semilogy(snr_range, BER_sim_1, 'm--');
hold off;
title('Matched Filter');
ylim([10^-4 0.5]);
xlabel('SNR (dB)');
ylabel('BER (Log)');
legend('Theoretical', 'Simulation');

figure;
semilogy(snr_range, BER_theo_2, 'r:');
hold on;
semilogy(snr_range, BER_sim_2, 'g--');
hold off;
title('No Filter');
ylim([10^-4 0.5]);
xlabel('SNR (dB)');
ylabel('BER (Log)');
legend('Theoretical', 'Simulation');

figure;
semilogy(snr_range, BER_theo_3, 'c--');
hold on;
semilogy(snr_range, BER_sim_3, 'y--');
hold off;

title('impulse response Filter h(t)=sqrt(3)*t');
ylim([10^-4 0.5]);
xlabel('SNR (dB)');
ylabel('BER (Log)');
legend('Theoretical', 'Simulation');
```

```
function [input] = pulse_shape(bits_number,samples_number,bits)

input = ones(bits_number,samples_number);

for i=1 : bits_number

if bits(i) == 0

input(i,:) = -input(i,:);

end

end

end


function [input_with_noise] = add_noise(bits_number,samples_number,input,E,snr)

sigma = sqrt(E/(2.0*snr));


noise = normrnd(0,sigma,[1,bits_number*samples_number]);


input_with_noise = input/sqrt(samples_number);


% add noise to input

for i=1 : bits_number

input_with_noise(i,:) = input_with_noise(i,:) + noise((samples_number)*(i-
1)+1:(samples_number)*(i));

end

input_with_noise = reshape(input_with_noise.', [], 1);

end


function [samples]=sample(output,bits_number,samples_number)

samples = ones(1,bits_number);

for i=0:bits_number-1

samples(i+1) = (output((samples_number - 1) + samples_number * i+1)) > 0;

end

end
```