International Institute of Information Technology
Bangalore

Principles of Communication Systems Lab

ECE 303P

Lab 8: Analog to Digital Communication

Mohd. Rizwan Shaikh
(IMT2019513)

October 25, 2021

Lab 8: Analog to Digital Communication

Mohd. Rizwan Shaikh

October 25, 2021

Abstract

In this lab report we study the process of analog to digital communication. We sample the signal using various sampling frequencies and analyze its effect on reconstruction of the signal. We also quantize and encode the given analog signal and also recover the signal from the encoded bits.

1 Introduction

1.1 Question 1

Consider an information signal $m(t) = A_m sin(2\pi f_m t)$ over two complete cycles with $A_m = 1$ V and $f_m = 10$ Hz.

(a) Sample this signal at rate $f_s = 10f_m$. Plot the continuous time and sampled signals.

(b) Reconstruct the signal from its samples. Plot the reconstructed signal.

(c) Sample the signal m(t) at the rate $f_s = 2f_m$ and $f_s = f_m$, and plot the reconstructed signal in each case.

(d) Write your observations.

1.2 Question 2

Consider an information signal $m(t) = A_m sin(2\pi f_m t)$ over one complete cycles with $A_m = 2$ V and $f_m = 10$ Hz.

(a) Sample this signal at rate $f_s = 50f_m$. Plot the continuous time and sampled signals.

(b) Quantize the sampled signal by dividing its range in L = 16, L = 64 and L = 256 uniforms steps. Assume mid point of a step as quantization level. Plot the quantize signal.

(c) Generate bit sequence by encoding the quantize samples in each case.

(d) Recover the signal from the bit sequence in each case, and write your observations.

2 Method

2.1 Question 1

We use the `plot` command to plot the give signal. Sampling is done by taking f_s as frequency. To reconstruct the signal we use sinc interpolation formula.

2.2 Question 2

We generate the sampled signal by considering frequency as f_s Hertz. For mid-step quantization, we first scale the signal, to [0, L] range. Then we add 0.5 to the signal so that the mid-step quantization levels correspond to the integers in between 0 to L. This is followed by rounding off the signal to the closest integer. Finally, we scale back the signal to its original range. Using this process, we get the quantized signal. For encoding the signal, we create a decimal to binary converter function from which we can get the encoded binary level.

3 Results and Analysis

3.1 Question 1

The MATLAB code for this question is as follows:

```
clear all, clc, close all;

Am = 1;     fm = 10;     %Parameters

%<===========1(a)=============>
figure(1)
fs = 10000;             %Plot the given signal
T = 0:1/fs:0.2;
mt = Am*sin(2*pi*fm*T);

subplot(2,2,1);
plot(T, mt, 'LineWidth', 1.5);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.2]);              ylim([-1.2 1.2]);
title('1(a) Given Signal: m(t) = A_msin(2\pif_mt)');
grid on;

fs = 10*fm;
t = 0:1/fs:0.2;
mt = Am*sin(2*pi*fm*t); %Sample the signal at the rate of fs = 10fm

subplot(2,2,2);         %Plot the sampled signal
stem(t, mt, 'MarkerSize', 5);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.2]);              ylim([-1.2 1.2]);
title('1(a) Sampled Signal for fs = 10fm');
grid on;

%<===========1(b)=============>
mt_reconstructed = zeros(1,length(T));
Ts = 1/fs;
for i=1:1:length(T)         %Reconstruct the signal
    for n=1:1:length(mt)
        mt_reconstructed(i) = mt_reconstructed(i) + mt(n)*sinc((T(i)-n*Ts)/Ts);
    end
end

subplot(2,2,3:4);             %Plot the reconstructed signal
plot(T, mt_reconstructed,'LineWidth', 1.5);
xlabel("---> Time(s)");     ylabel('---> m(t)');
```

```
xlim([0 0.2]);              ylim([-1.2 1.2]);
title('1(b) Reconstructed Signal');
grid on;

%<===========1(c)=============>
figure(2);
fs = 2*fm;
t = 0:1/fs:0.2;
mt = Am*sin(2*pi*fm*t);         %Sample the signal at rate of fs=2fm

subplot(2,2,1);
stem(t, mt, 'MarkerSize', 5);   %Plot the sampled signal
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.2]);              ylim([-1.2 1.2]);
title('1(c) Sampled Signal for fs = 2fm');
grid on;

mt_reconstructed = zeros(1,length(T));
Ts = 1/fs;
for i=1:1:length(T)         %Reconstruct the signal
    for n=1:1:length(mt)
        mt_reconstructed(i) = mt_reconstructed(i) + mt(n)*sinc((T(i)-n*Ts)/Ts);
    end
end

subplot(2,2,2);             %Plot the reconstructed signal
plot(T, mt_reconstructed,'LineWidth', 1.5);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.2]);              ylim([-1.2 1.2]);
title('1(c) Reconstructed Signal for fs = 2fm');
grid on;


fs = fm;
t = 0:1/fs:0.2;
mt = Am*sin(2*pi*fm*t);     %Sample the signal at rate of fs=fm

subplot(2,2,3);             %Plot the sampled signal
stem(t, mt, 'MarkerSize', 5);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.2]);              ylim([-1.2 1.2]);
title('1(c) Sampled Signal for fs = fm');
grid on;

mt_reconstructed = zeros(1,length(T));
Ts = 1/fs;
for i=1:1:length(T)         %Reconstruct the signal
    for n=1:1:length(mt)
        mt_reconstructed(i) = mt_reconstructed(i) + mt(n)*sinc((T(i)-n*Ts)/Ts);
    end
end

subplot(2,2,4);             %Plot the reconstructed signal
plot(T, mt_reconstructed,'LineWidth', 1.5);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.2]);              ylim([-1.2 1.2]);
title('1(c) Reconstructed Signal for fs = fm');
grid on;
```

The plots obtained for this question is as follows:

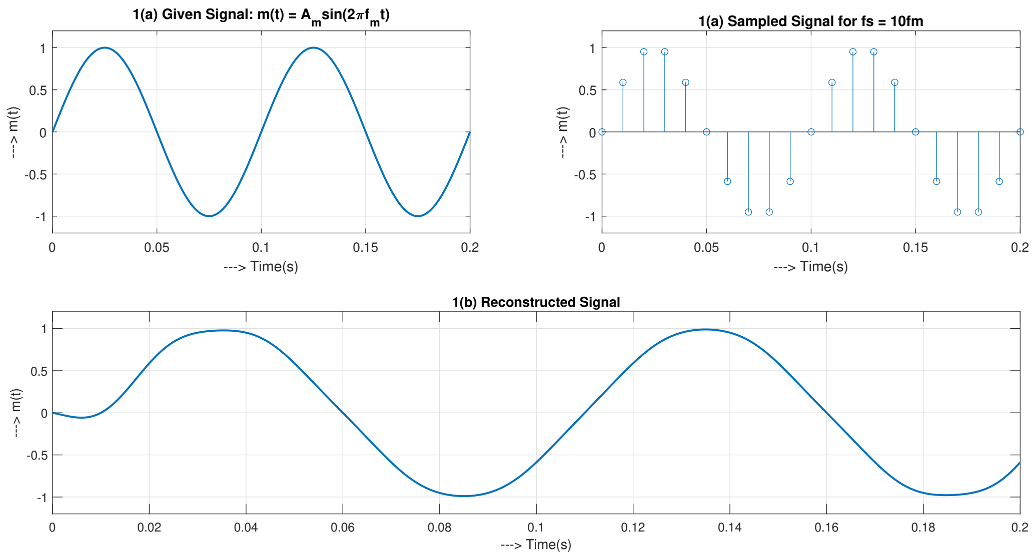

Figure 1: Given signal, Sampled signal for $f_s = 10f_m$ and its reconstructed signal (1(a) and 1(b))

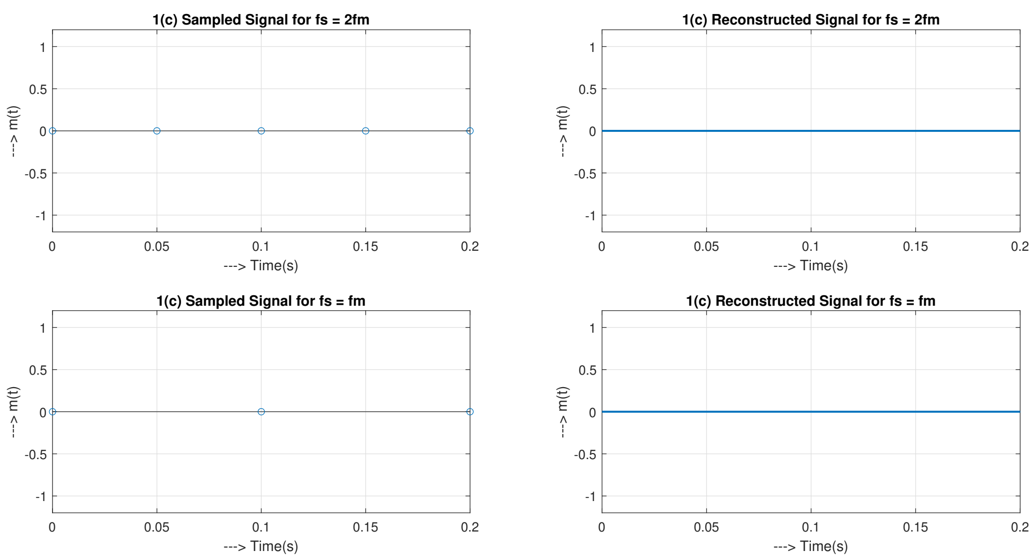

Figure 2: 1(c) Sampled and reconstructed signal for $f_s = 2f_m$ and $f_s = f_m$

For the given signal, we sampled the signal for $f_s = f_m$, $2f_m$ and $10f_m$. We used sinc interpolation formula to get back the signal from its sampled signal.

We see that for $f_s = 10f_m$, we are able to get back the original signal from its sampled signal. Nyquist theorem justifies this observation.

For $f_s = 2f_m$, we are not getting the original signal. In fact, the given signal is a straight line. Theoretically, we should have been able to reconstruct the signal due to Nyquist theorem. But it is not happening here because all the points are exactly on zeros.

For $f_s = f_m$, we are not able to reconstruct the signal. This is expected result because of Nyquist theorem.

3.2 Question 2

The MATLAB code for this question is as follows:

```
clear all, clc, close all;

Am = 2;      fm = 10;      %Parameters

%<===========2(a)=============>
figure(1);
fs = 10000;              %Plot the given signal
T = 0:1/fs:0.1;
mt = Am*sin(2*pi*fm*T);

subplot(2,1,1);
plot(T, mt, 'LineWidth', 1.5);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.1]);              ylim([-2.2 2.2]);
title('2(a) Given Signal: m(t) = A_msin(2\pif_mt)');
grid on;

fs = 50*fm;
t = 0:1/fs:0.1;
mt = Am*sin(2*pi*fm*t);

for i=1:length(mt)
    if(mt(i) > 1.995)
        mt(i) = 2;
    end
end

subplot(2,1,2);
stem(t, mt, 'MarkerSize', 3);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.1]);              ylim([-2.2 2.2]);
title('2(a) Sampled Signal for fs = 50fm');
grid on;

%<===========2(b)=============>
figure(2);
%<------L = 16----->
L = 16;
mt1 = quantize(L, mt);

subplot(3,2,1);
stem(t, mt1, 'MarkerSize', 3);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.1]);              ylim([-2.2 2.2]);
title('2(b) Quantized Sampled Signal for L = 16');
grid on;

%<------L = 64----->
L = 64;
mt2 = quantize(L, mt);
```

```
subplot(3,2,3);
stem(t, mt2, 'MarkerSize', 3);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.1]);              ylim([-2.2 2.2]);
title('2(b) Quantized Sampled Signal for L = 64');
grid on;

%<------L = 256----->
L = 256;
mt3 = quantize(L, mt);

subplot(3,2,5);
stem(t, mt3, 'MarkerSize',3);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.1]);              ylim([-2.2 2.2]);
title('2(b) Quantized Sampled Signal for L = 256');
grid on;

%<===========2(c)=============>
%<------L = 16----->
L = 16;
mt1Encoded = encode(L, mt1);
display(mt1Encoded);

%<------L = 64----->
L = 64;
mt2Encoded = encode(L, mt2);
display(mt2Encoded);

%<------L = 256----->
L = 256;
mt3Encoded = encode(L, mt3);
display(mt3Encoded);

%<===========2(d)=============>
%<------L = 16----->
mt_reconstructed = zeros(1,length(T));
Ts = 1/fs;
for i=1:1:length(T)          %Reconstruct the signal
    for n=1:1:length(mt1)
        mt_reconstructed(i) = mt_reconstructed(i) + ...
            mt1(n)*sinc((T(i)-n*Ts)/Ts);
    end
end

subplot(3,2,2);
plot(T, mt_reconstructed);
xlabel("---> Time(s)");     ylabel('---> m(t)');
xlim([0 0.1]);              ylim([-2.2 2.2]);
title('2(d) Reconstructed Signal for L = 16');
grid on;

%<------L = 64----->
mt_reconstructed = zeros(1,length(T));
Ts = 1/fs;
for i=1:1:length(T)          %Reconstruct the signal
    for n=1:1:length(mt2)
        mt_reconstructed(i) = mt_reconstructed(i) + ...
```

```
                mt2(n)*sinc((T(i)-n*Ts)/Ts);
        end
    end

    subplot(3,2,4);
    plot(T, mt_reconstructed);
    xlabel("---> Time(s)");     ylabel('---> m(t)');
    xlim([0 0.1]);              ylim([-2.2 2.2]);
    title('2(d) Reconstructed Signal for L = 64');
    grid on;


    %<------L = 256----->
    mt_reconstructed = zeros(1,length(T));
    Ts = 1/fs;
    for i=1:1:length(T)          %Reconstruct the signal
        for n=1:1:length(mt3)
            mt_reconstructed(i) = mt_reconstructed(i) + ...
                mt3(n)*sinc((T(i)-n*Ts)/Ts);
        end
    end

    subplot(3,2,6);
    plot(T, mt_reconstructed);
    xlabel("---> Time(s)");     ylabel('---> m(t)');
    xlim([0 0.1]);              ylim([-2.2 2.2]);
    title('2(d) Reconstructed Signal for L = 256');
    grid on;


    function mtEnc = encode(L, mt)
        Gap = 4/L;
        LowLimit = -2 + Gap/2;
        HighLimit = 2 - Gap/2;
        levels = linspace(LowLimit, HighLimit, L);
        mtEnc = [];

        for i=1:length(mt)
            for j=1:length(levels)
                if(mt(i) == levels(j))
                    mtEnc = [mtEnc dec2bin(j-1, log2(L))];
                    break;
                end
            end
        end
    end

    function mtq = quantize(L, mt)
        scalingFactor = 4/L;
        mtq = ((mt + 2) / scalingFactor) + 0.5;
        mtq = round(mtq);
        for i=1:length(mtq)
            if(mtq(i) == L + 1)
                mtq(i) = L;
            end
        end
        mtq = ((mtq - 0.5) * scalingFactor) - 2;
    end
```

The plots for this question is as follows:

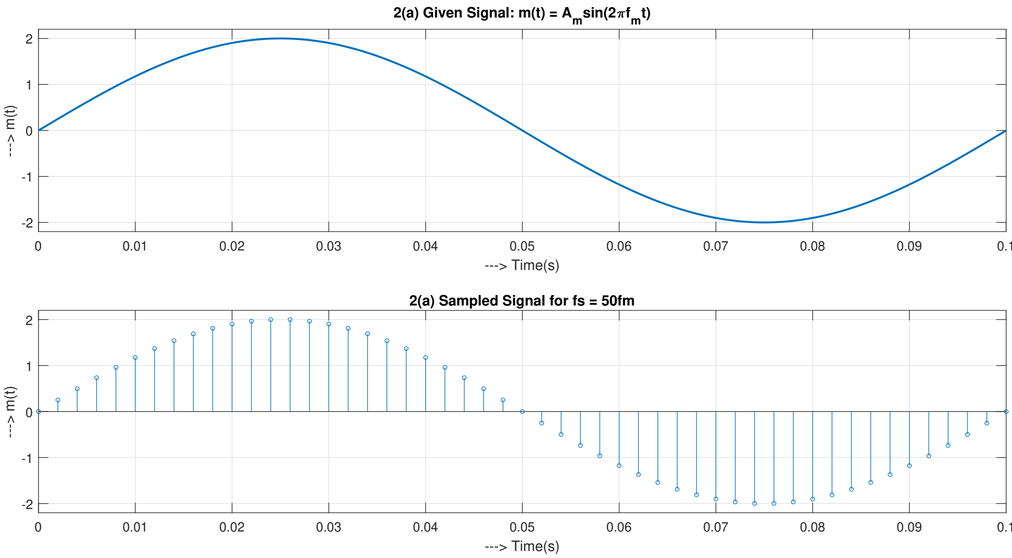

Figure 3: 2(a) Given signal and its Sampled signal for $f_s = 50f_m$

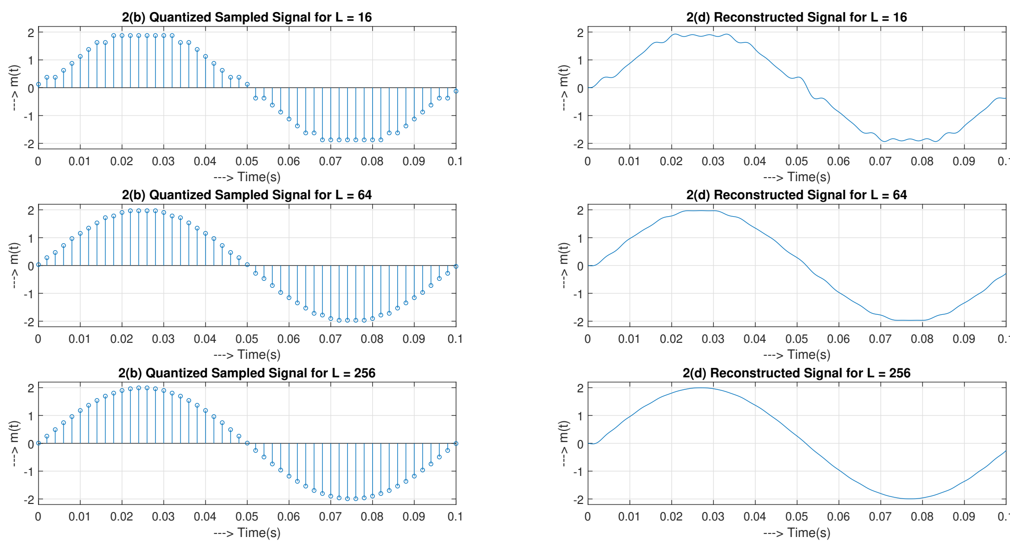

Figure 4: 2(b) Sampled signal for L = 16, 64 and 256 and 2(d) Reconstructed signal from bit sequence

The bit sequence obtained by encoding the signal is as follows:
L = 16:
’10001001100110101011110011011110111011111111111111111111111111111111111110111011
011100101110101001100110000110011001010100001100100001000100000000000000000000
00000000000000100010010001101000101011001100111’
L = 64:

’1000001001001001111010111011111100101101011110001110111111001111101111111111
1111111111111111110111100111011111000110101110010101111101011100111100100100000
0011011011000010100010000001101001010000111000100000011000001000000000000000000
0000000000001000011000100000111001010001101010000010100011000011011011111’
L = 256:
’100000001001000010011111101011111011110111001011110101111110001011101100111100
0111111100111111101111111111111111111111110111111001111100111110110011100010110
1011111001011101111011010111110011111100100001000000001101111011000000010100000
1000010001101000010100000011101000100110000110000000110000000100000000000000000
0000000100000011000000110000010011000111010010100000110100010000100101000001100
0000110111101111111’
Here, we see that as the value of L increases, the reconstructed signal comes closer to original signal.