

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:

```
1      clear all, clc, close all;
2
3      Am = 1;      fm = 10;      %Parameters
4
5      %<=====1(a)=====>
6      figure(1)
7      fs = 10000;      %Plot the given signal
8      T = 0:1/fs:0.2;
9      mt = Am*sin(2*pi*fm*T);
10
11     subplot(2,2,1);
12     plot(T, mt, 'LineWidth', 1.5);
13     xlabel("----> Time(s)");      ylabel('----> m(t)');
14     xlim([0 0.2]);      ylim([-1.2 1.2]);
15     title('1(a) Given Signal: m(t) = Amsin(2\pif.mt)');
16     grid on;
17
18     fs = 10*fm;
19     t = 0:1/fs:0.2;
20     mt = Am*sin(2*pi*fm*t); %Sample the signal at the rate of fs = 10fm
21
22     subplot(2,2,2);      %Plot the sampled signal
23     stem(t, mt, 'MarkerSize', 5);
24     xlabel("----> Time(s)");      ylabel('----> m(t)');
25     xlim([0 0.2]);      ylim([-1.2 1.2]);
26     title('1(a) Sampled Signal for fs = 10fm');
27     grid on;
28
29     %<=====1(b)=====>
30     mt_reconstructed = zeros(1,length(T));
31     Ts = 1/fs;
32     for i=1:length(T)      %Reconstruct the signal
33         for n=1:length(mt)
34             mt_reconstructed(i) = mt_reconstructed(i) + mt(n)*sinc((T(i)-n*Ts)/Ts);
35         end
36     end
37
38     subplot(2,2,3:4);      %Plot the reconstructed signal
39     plot(T, mt_reconstructed, 'LineWidth', 1.5);
40     xlabel("----> Time(s)");      ylabel('----> m(t)');
```

```

41     xlim([0 0.2]);                ylim([-1.2 1.2]);
42     title('1(b) Reconstructed Signal');
43     grid on;
44
45     %<=====1(c)=====>
46     figure(2);
47     fs = 2*fm;
48     t = 0:1/fs:0.2;
49     mt = Am*sin(2*pi*fm*t);        %Sample the signal at rate of fs=2fm
50
51     subplot(2,2,1);
52     stem(t, mt, 'MarkerSize', 5);  %Plot the sampled signal
53     xlabel("----> Time(s)");       ylabel('----> m(t)');
54     xlim([0 0.2]);                ylim([-1.2 1.2]);
55     title('1(c) Sampled Signal for fs = 2fm');
56     grid on;
57
58     mt_reconstructed = zeros(1,length(T));
59     Ts = 1/fs;
60     for i=1:1:length(T)            %Reconstruct the signal
61         for n=1:1:length(mt)
62             mt_reconstructed(i) = mt_reconstructed(i) + mt(n)*sinc((T(i)-n*Ts)/Ts);
63         end
64     end
65
66     subplot(2,2,2);                %Plot the reconstructed signal
67     plot(T, mt_reconstructed, 'LineWidth', 1.5);
68     xlabel("----> Time(s)");       ylabel('----> m(t)');
69     xlim([0 0.2]);                ylim([-1.2 1.2]);
70     title('1(c) Reconstructed Signal for fs = 2fm');
71     grid on;
72
73
74     fs = fm;
75     t = 0:1/fs:0.2;
76     mt = Am*sin(2*pi*fm*t);        %Sample the signal at rate of fs=fm
77
78     subplot(2,2,3);                %Plot the sampled signal
79     stem(t, mt, 'MarkerSize', 5);
80     xlabel("----> Time(s)");       ylabel('----> m(t)');
81     xlim([0 0.2]);                ylim([-1.2 1.2]);
82     title('1(c) Sampled Signal for fs = fm');
83     grid on;
84
85     mt_reconstructed = zeros(1,length(T));
86     Ts = 1/fs;
87     for i=1:1:length(T)            %Reconstruct the signal
88         for n=1:1:length(mt)
89             mt_reconstructed(i) = mt_reconstructed(i) + mt(n)*sinc((T(i)-n*Ts)/Ts);
90         end
91     end
92
93     subplot(2,2,4);                %Plot the reconstructed signal
94     plot(T, mt_reconstructed, 'LineWidth', 1.5);
95     xlabel("----> Time(s)");       ylabel('----> m(t)');
96     xlim([0 0.2]);                ylim([-1.2 1.2]);
97     title('1(c) Reconstructed Signal for fs = fm');
98     grid on;

```

The plots obtained for this question is as follows:

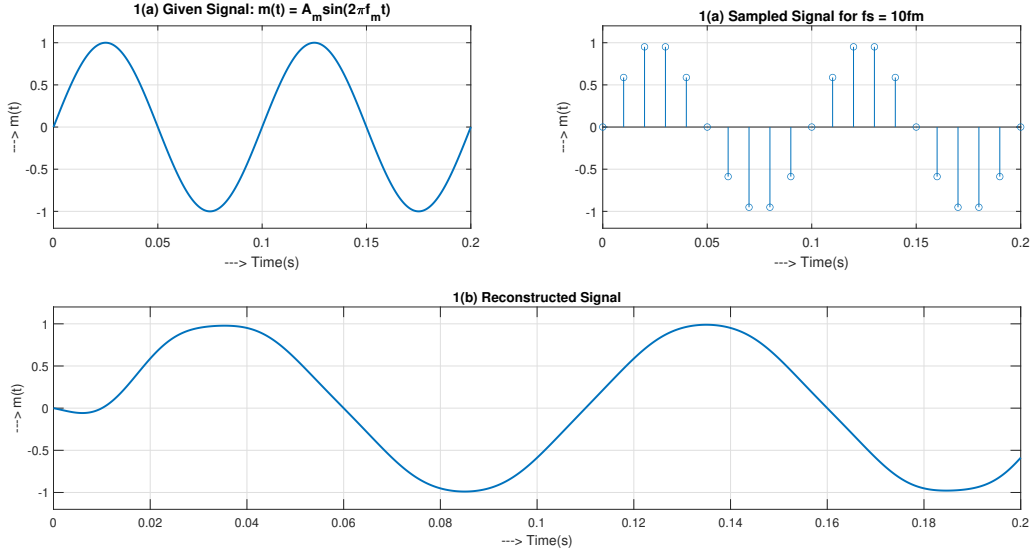


Figure 1: Given signal, Sampled signal for $f_s = 10f_m$ and its reconstructed signal (1(a) and 1(b))

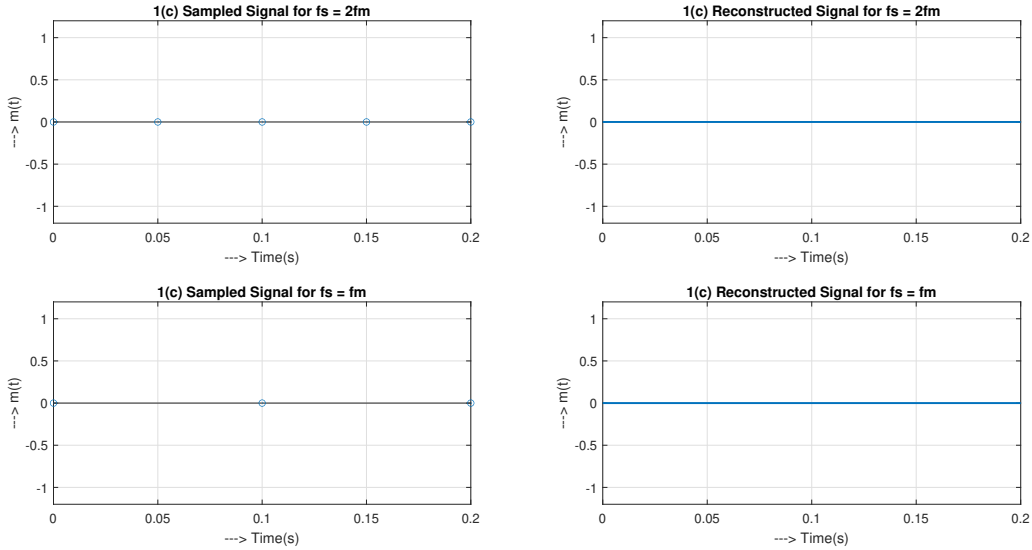


Figure 2: 1(c) Sampled and reconstructed signal for $f_s = 2f_m$ and $f_s = f_m$

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:

```

1      clear all, clc, close all;
2
3      Am = 2;      fm = 10;      %Parameters
4
5      %<=====2(a)=====>
6      figure(1);
7      fs = 10000;      %Plot the given signal
8      T = 0:1/fs:0.1;
9      mt = Am*sin(2*pi*fm*T);
10
11     subplot(2,1,1);
12     plot(T, mt, 'LineWidth', 1.5);
13     xlabel("----> Time(s)");      ylabel('----> m(t)');
14     xlim([0 0.1]);      ylim([-2.2 2.2]);
15     title('2(a) Given Signal: m(t) = Amsin(2\pif*mt)');
16     grid on;
17
18     fs = 50*fm;
19     t = 0:1/fs:0.1;
20     mt = Am*sin(2*pi*fm*t);
21
22     for i=1:length(mt)
23         if(mt(i) > 1.995)
24             mt(i) = 2;
25         end
26     end
27
28     subplot(2,1,2);
29     stem(t, mt, 'MarkerSize', 3);
30     xlabel("----> Time(s)");      ylabel('----> m(t)');
31     xlim([0 0.1]);      ylim([-2.2 2.2]);
32     title('2(a) Sampled Signal for fs = 50fm');
33     grid on;
34
35     %<=====2(b)=====>
36     figure(2);
37     %<-----L = 16----->
38     L = 16;
39     mt1 = quantize(L, mt);
40
41     subplot(3,2,1);
42     stem(t, mt1, 'MarkerSize', 3);
43     xlabel("----> Time(s)");      ylabel('----> m(t)');
44     xlim([0 0.1]);      ylim([-2.2 2.2]);
45     title('2(b) Quantized Sampled Signal for L = 16');
46     grid on;
47
48     %<-----L = 64----->
49     L = 64;
50     mt2 = quantize(L, mt);

```

```

51
52 subplot(3,2,3);
53 stem(t, mt2, 'MarkerSize', 3);
54 xlabel("----> Time(s)");      ylabel('----> m(t)');
55 xlim([0 0.1]);                ylim([-2.2 2.2]);
56 title('2(b) Quantized Sampled Signal for L = 64');
57 grid on;
58
59 %<-----L = 256----->
60 L = 256;
61 mt3 = quantize(L, mt);
62
63 subplot(3,2,5);
64 stem(t, mt3, 'MarkerSize', 3);
65 xlabel("----> Time(s)");      ylabel('----> m(t)');
66 xlim([0 0.1]);                ylim([-2.2 2.2]);
67 title('2(b) Quantized Sampled Signal for L = 256');
68 grid on;
69
70 %<=====2(c)=====>
71 %<-----L = 16----->
72 L = 16;
73 mt1Encoded = encode(L, mt1);
74 display(mt1Encoded);
75
76 %<-----L = 64----->
77 L = 64;
78 mt2Encoded = encode(L, mt2);
79 display(mt2Encoded);
80
81 %<-----L = 256----->
82 L = 256;
83 mt3Encoded = encode(L, mt3);
84 display(mt3Encoded);
85
86 %<=====2(d)=====>
87 %<-----L = 16----->
88 mt_reconstructed = zeros(1,length(T));
89 Ts = 1/fs;
90 for i=1:1:length(T)           %Reconstruct the signal
91     for n=1:1:length(mt1)
92         mt_reconstructed(i) = mt_reconstructed(i) + ...
93             mt1(n)*sinc((T(i)-n*Ts)/Ts);
94     end
95 end
96
97 subplot(3,2,2);
98 plot(T, mt_reconstructed);
99 xlabel("----> Time(s)");      ylabel('----> m(t)');
100 xlim([0 0.1]);                ylim([-2.2 2.2]);
101 title('2(d) Reconstructed Signal for L = 16');
102 grid on;
103
104 %<-----L = 64----->
105 mt_reconstructed = zeros(1,length(T));
106 Ts = 1/fs;
107 for i=1:1:length(T)           %Reconstruct the signal
108     for n=1:1:length(mt2)
109         mt_reconstructed(i) = mt_reconstructed(i) + ...

```

```

109         mt2(n)*sinc((T(i)-n*Ts)/Ts);
110     end
111 end
112 subplot(3,2,4);
113 plot(T, mt_reconstructed);
114 xlabel("----> Time(s)");      ylabel('----> m(t)');
115 xlim([0 0.1]);                ylim([-2.2 2.2]);
116 title('2(d) Reconstructed Signal for L = 64');
117 grid on;
118
119
120 %<-----L = 256----->
121 mt_reconstructed = zeros(1,length(T));
122 Ts = 1/fs;
123 for i=1:length(T)              %Reconstruct the signal
124     for n=1:length(mt3)
125         mt_reconstructed(i) = mt_reconstructed(i) + ...
126             mt3(n)*sinc((T(i)-n*Ts)/Ts);
127     end
128 end
129 subplot(3,2,6);
130 plot(T, mt_reconstructed);
131 xlabel("----> Time(s)");      ylabel('----> m(t)');
132 xlim([0 0.1]);                ylim([-2.2 2.2]);
133 title('2(d) Reconstructed Signal for L = 256');
134 grid on;
135
136
137 function mtEnc = encode(L, mt)
138     Gap = 4/L;
139     LowLimit = -2 + Gap/2;
140     HighLimit = 2 - Gap/2;
141     levels = linspace(LowLimit, HighLimit, L);
142     mtEnc = [];
143
144     for i=1:length(mt)
145         for j=1:length(levels)
146             if(mt(i) == levels(j))
147                 mtEnc = [mtEnc dec2bin(j-1, log2(L))];
148                 break;
149             end
150         end
151     end
152 end
153
154 function mtq = quantize(L, mt)
155     scalingFactor = 4/L;
156     mtq = ((mt + 2) / scalingFactor) + 0.5;
157     mtq = round(mtq);
158     for i=1:length(mtq)
159         if(mtq(i) == L + 1)
160             mtq(i) = L;
161         end
162     end
163     mtq = ((mtq - 0.5) * scalingFactor) - 2;
164 end

```

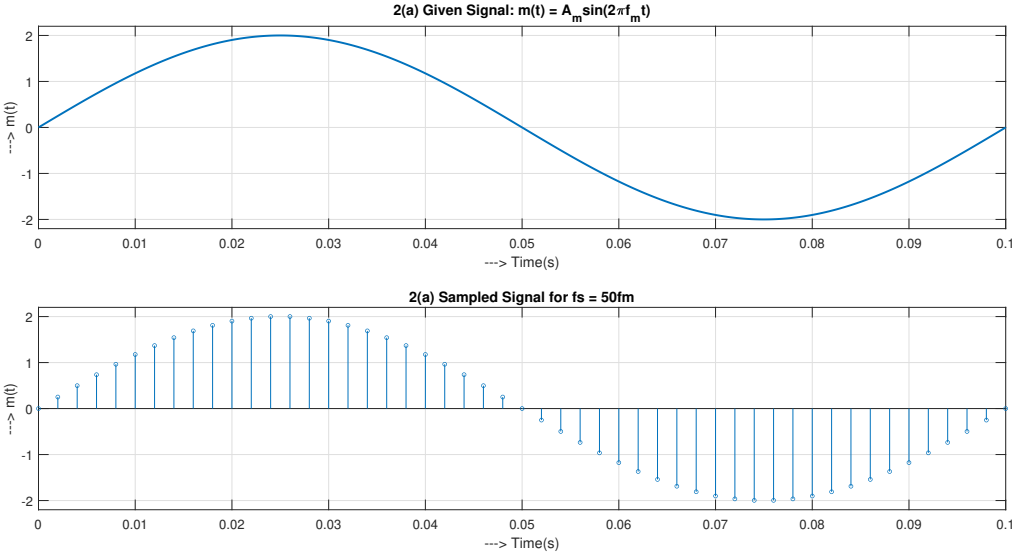



Figure 3: 2(a) Given signal and its Sampled signal for $f_s = 50f_m$

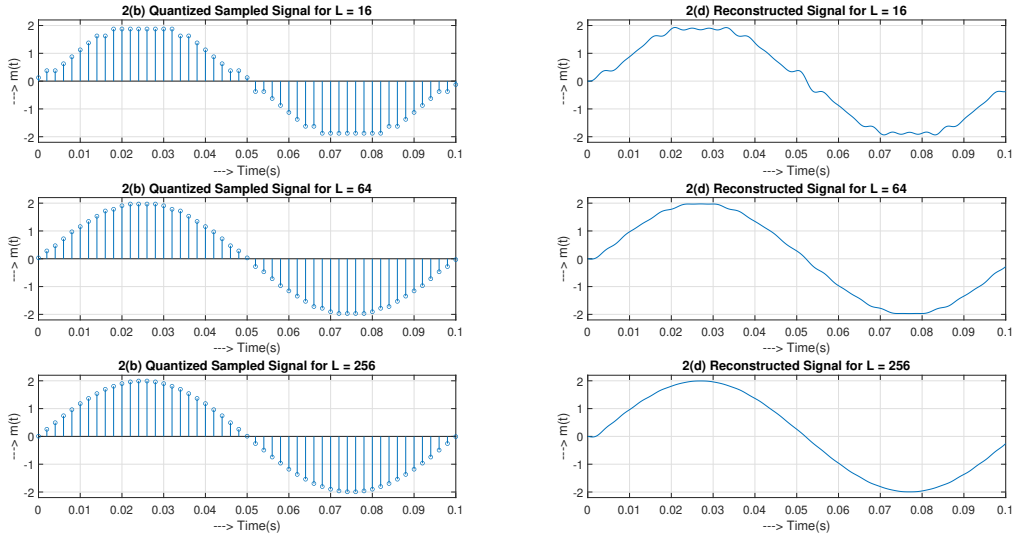


Figure 4: 2(b) Sampled signal for $L = 16, 64$ and 256 and 2(d) Reconstructed signal from bit sequence

The bit sequence obtained by encoding the signal is as follows:

L = 16:

```
'1000100110011010101111001101111011101111111111111111111111111111111110111011
0111001011101010011001100001100110010101000011001000010001000000000000000000000
0000000000000000100010010001101000101011001100111'
```

L = 64:

```
'1000001001001001111010111011111100101101011110001110111110011111011111111111
111111111111111111011110011101111100011010111001010111110101110011110010010000
001101101100001010001000000011010010100001110001000000110000010000000000000000
0000000000001000011000100000111001010001101010000010100011000011011011111'
```

L = 256:

```
'10000000100100001001111110101111101111011100101111010111111000101110110011110
01111111001111111011111111111111111111111111111110111111001111100111110110011100010110
101111100101110111101101011111001111110010000100000000110111101100000010100000
100001000110100001010000001110100010011000011000000011000000010000000000000000
000000010000001100000110000010011000111010010100000110100010000100101000001100
000011011110111111'
```

Here, we see that as the value of L increases, the reconstructed signal comes closer to original signal.