Communication Theory Lab Report

Name: Aditya Sarkar
Roll No.: B19003
Branch: Electrical Engineering
Date: April 15, 2022
Communication Theory Lab
(EE304P, Spring-2022)

Lab Assignment # 7

Indian Institute of Technology Mandi, HP, India
School of Computing and Electrical Engineering (SCEE)

Quantization Techniques and Pulse Code Modulation

Abstract

In this experiment, we have tried to understand the concept of quantization in digital communication and pulse code modulation (PCM). Here we have only focused on uniform quantisation. We analysed the waveforms generated after applying quantization, by plotting their graphs in MATLAB. In all the techniques, we observed the quantized waveforms and have tried to explained them in this report.

1 Theory

In communication theory, quantization is defined as a way of mapping continuous values into a finite set of values depending on the number of bits being used to transmit data. Suppose we have 4 bits, then the number of levels of quantization will be $2^4 = 16$ levels. Quantization techniques are widely used for digital communication because of its improved transmission and information storage capacity. We have two types of quantization, but in this report, we will be focusing only on the uniform quantization.

There are two types of quantization techniques

1. Uniform Quantization
2. Non Uniform Quantization

Uniform Quantization is a way of quantization in which one tries to keep uniformly spaced quantization levels. Quantization always results in a loss of information. The difference between the quantized signal and the input signal is called quantization error. On increasing the number of levels will reduce the loss of information i.e. the quantization error. These levels are also called reconstruction level and the space between them is termed as step size.

Pulse Code Modulation is a widely used method to represent any sampled analog signal in a digital format. In this, we sample the amplitude of the analog signal regularly at uniform intervals and each sample is quantized to the nearest value depending on the range of digital steps. PCM encompasses

many techniques, out of which, the most popular is Linear pulse-code modulation (LPCM). Here the quantization levels are linearly uniform. In other PCM techniques, quantization levels vary as a function of amplitude, for example, the A-law algorithm or the μ-law algorithm. LPCM is so commonly used that if someone says PCM, then there is high probability that he/she is referring to data encoded as LPCM.

1.1 Block Diagram

The block diagram is for quantization. In step 1, we convert an analog signal to discrete signal using sampling techniques. Analog signal has a continuous amplitude and time period while discrete signal has a continuous amplitude but discrete time period. In step 2, we convert an discrete signal to digital signal using quantization techniques. Digital signal has a discrete amplitude and a discrete time period.

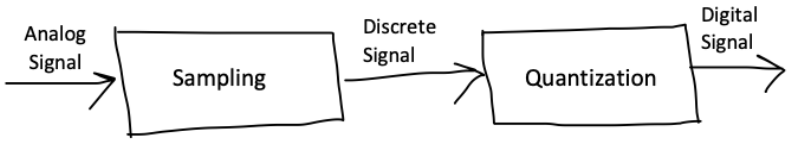

Figure 1: Quantization block diagram

The next block diagram is for PCM. The three major components of PCM are : 1) sampler, 2) quantizer and 3) encoder. In step 1, sampler samples the input signals at regular intervals. In this, we ensure that we follow the Nyquist rate of sampling. In step 2, quantizer rounds off each sample to the nearest discrete level. In step 3, the quantized signal is converted to a binary sequence having bits of 0 and 1.

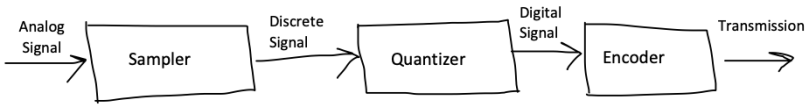

Figure 2: PCM block diagram

1.2 Expected Outcome

The figures below are not drawn to scale.

1.2.1 Answer 1

This question was taken directly from what was taught in the lectures.

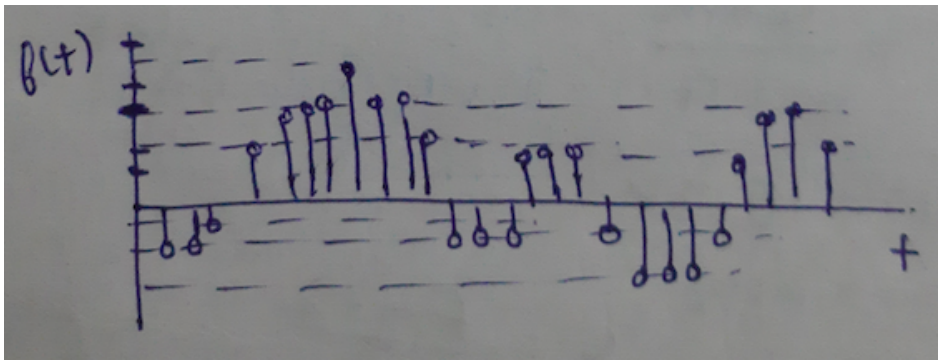

Figure 3: Figure for part A

1.2.2 Answer 2

This is directly taken from the lecture.

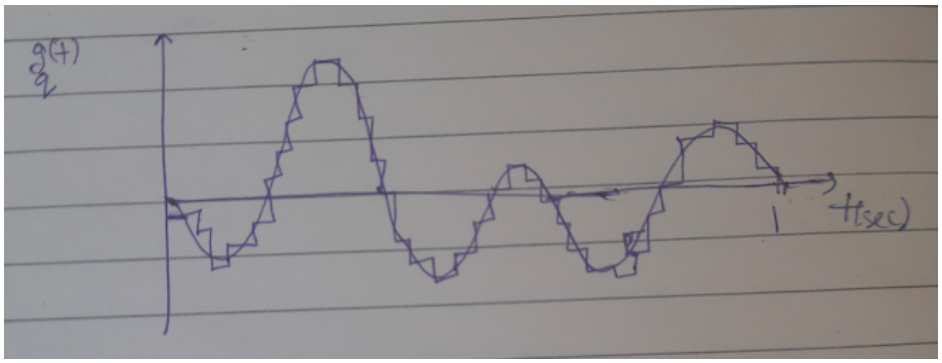

Figure 4: Figure for part B

1.2.3 Answer 3

This question was taking inference from what was taught in the lectures.

1.3 Application

Applications of quantization:

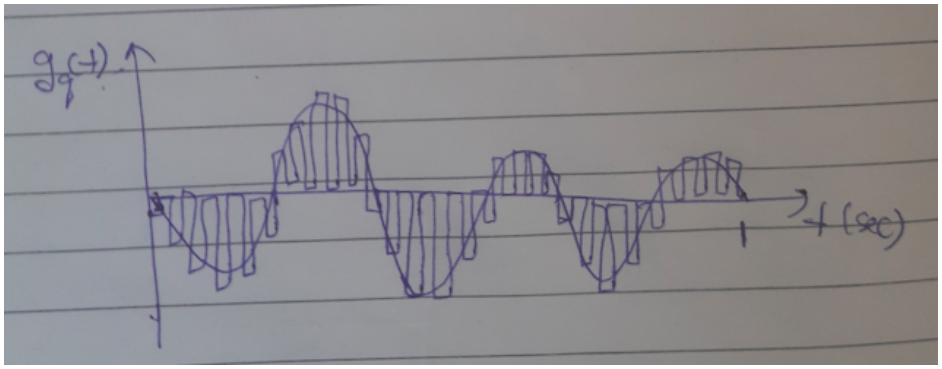

Figure 5: Figure for part C

1. `Control systems:` It is often used in low power micro-controllers. It helps produce an output signal given an input signal with a reasonable precision under low power constraints.
2. `Deep learning :` To reduce power consumption and memory and to accelerate the inferences, one performs quantization. 8 bit integers are used to make inferences, while having a decent accuracy and size of networks.
3. `Image compression:` Quantization is used in JPEG image compression.

Applications of Pulse code modulation:

1. `Compact Disc (CD):` PCM is often used in CDs.
2. `Telephony :` PCM is used in telephony and space communications. It is also used in satellite transmission system.

2 Results and Inferences

2.1 Answer 1

We have,

$$g(t) = (sin(2 * pi * 1 * t) – sin(2 * pi * 3 * t))$$

Signal amplitude range that is from -2 to 2 is divided into 16 levels for quantization. Hence there will be a constant difference between successive levels which would be 0.25 (=2mp/L = 2*2/16).This is uniform quantization as we can see that the step size is fixed or uniform.

Inferences are :

1. We can observe uniform quantization from the plot as the step size is constant and uniform.

2. Non-uniform quantisation can be a better option here because we can see smaller amplitude pulses in the waveform. Uniform quantization is more susceptible to adding more noise to the signal. Non-uniform distribution will better adapt by having smaller step sizes at low amplitudes and bigger step sizes at higher amplitudes.

3. Number of levels are 16. Thus $log_2(n) = 4$ bits are required for information transmission.

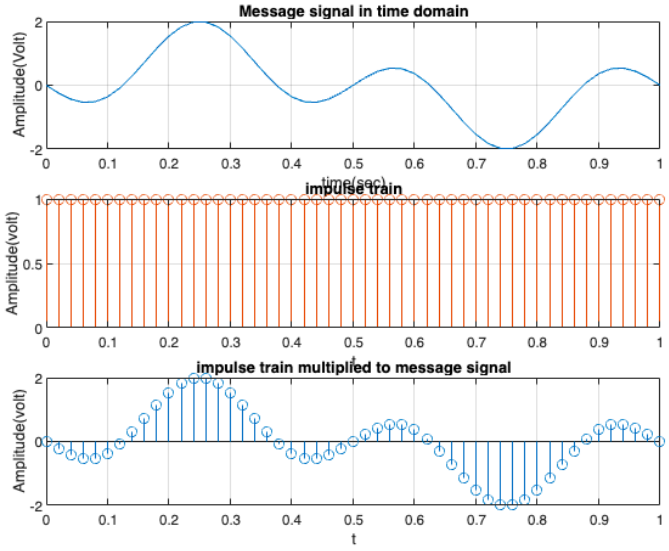

Figure 6: Figure 1 for part A

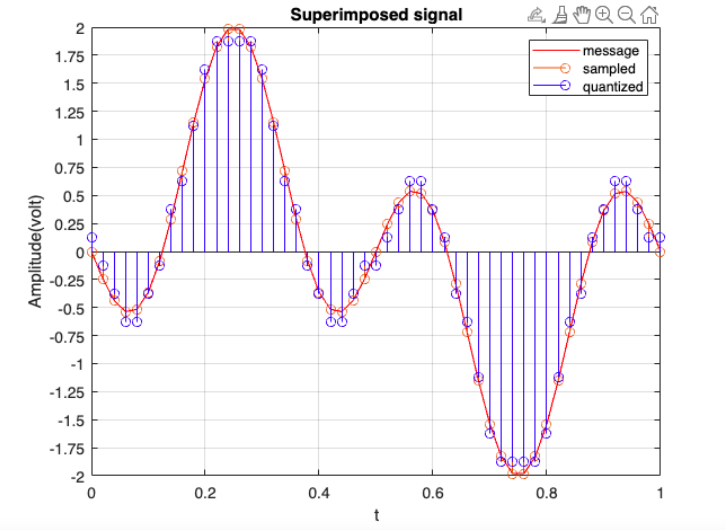

Figure 7: Figure 2 for part A

2.2 Answer 2

Our pulse train is defined as -

$$P(t) = \Sigma p(t - k.T_s)$$

Where p(t) is defined as -

p(t) = 1 for $0 \leq t \geq T_p$
p(t) = 0 for $T_p \leq t \geq T_s$

For the first part, we have $T_p = T_s$. Thus the pulse train is high for T_p duration and we know $T_p = T_s = 0.02$ sec, so it is high for every value of Ts.

Inferences are :

1. We can see that the impulse is a Non-return-to-zero (NRZ) type. This is because from the plot we can observe that the pulse train is not returning to 0, or it is always high. Hence by definition, it is NRZ.

2. We derive the signal g_q(t) when we multiply the quantized signal with the pulse train for interval $T_p = T_s$ duration. So, what we should get is that the amplitude of the g_q(t) signal remains constant for each T_p interval. Here $T_p = T_s$, so it will remain 1 for the whole time period of 1 sec. However, in general case when $T_p < T_s$, it will changes depending on the change of the amplitude of quantized signal, which we made, and for T_p duration, it will remain constant.

3. Step size is fixed or constant, so it's uniform quantization.

4. We have 16 levels, hence we need $log_2(16) = 4$ bits for representation.

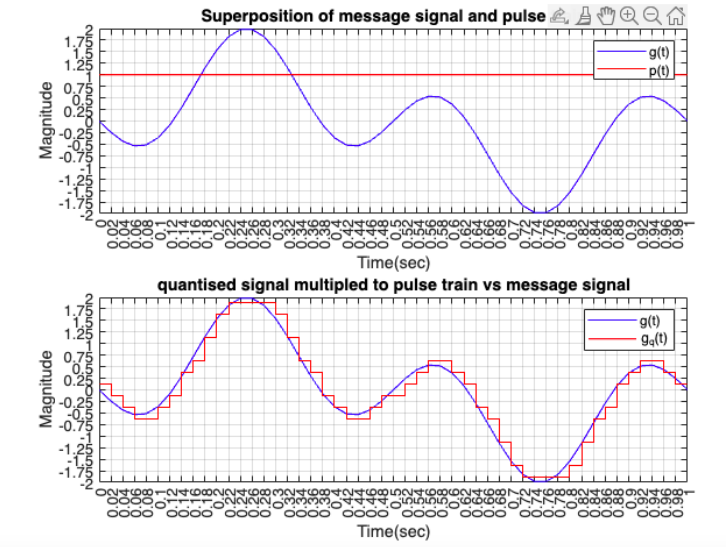

Figure 8: Figure for part B

2.3 Answer 3

It is given that the pulse train is high for $T_s/2$ duration and for other $T_s/2$ duration, it will remain low. We know that the sampling frequency is 50Hz. So the time period Ts = 0.02 sec. For half of the time that is $T_p = 0.01$ sec, it will be high (1) and for the next half time, it will be low (0).

Inferences are :

1. We can observe that the pulse train is returning to zero within the sample time Ts only, so it is a return-to-zero (RZ) type pulse.
2. We derive the signal g_q(t) when we multiply the quantized signal with the pulse train for interval of $T_p = T_s/2$ (half of sample time) duration. So, what we should get is that the amplitude of the g_q(t) signal remains constant for each T_p interval which is half of sample duration. After that, upto full sample time (T_s), it will be 0. We can observe the same thing in the plots. This will keep on repeating.
3. Step size is fixed or constant, so it's uniform quantization.
4. We have 16 levels, hence we need $log_2(16) = 4$ bits for representation.

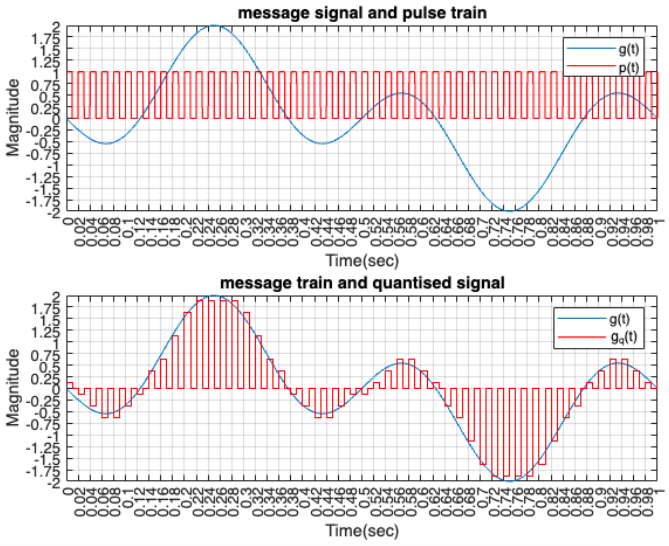

Figure 9: Figure for part C

2.4 Answer 4

The encoding process that is being described in the question is Pulse code modulation (PCM). It is used for converting analog signal to digital signal.It is not the best way of encoding, because it has some limitations. First is, if your message signal changes smoothly, then PCM will be a good idea to implement. However, if it is not the case, that is there is abrupt changes in the message signal, then it will fail to encode it. Second is, it requires larger number of bits to represent the quantization levels. For 16 levels, it is requiring 4 bits. If you consider DM, where we send the differences between previous and present sample, it would have required only 1 bit, as it only has to represent the change in each levels. Third is, PCM adds in more noise as compared to DM due to its larger dynamic range and step size.

Table 1: Question 1b - Table

Question 1(b)							
Signal	0	Ts	2Ts	3Ts	4Ts	5Ts	6Ts
g(t)	0	-0.242	-0.435	-0.536	0.516	-0.363	-0.085
p(t)	1	1	1	1	1	1	1
gs(kTs)	0	-0.242	-0.435	-0.536	0.516	-0.363	-0.085
gs1(kTs)	0.125	-0.125	-0.375	-0.625	-0.625	-0.375	-0.125
Enc	1000	0111	0110	0101	0101	0110	0111

Table 2: Question 1c - Table

Question 1(c)							
Signal	0	Ts	2Ts	3Ts	4Ts	5Ts	6Ts
g(t)	0	-0.242	-0.435	-0.536	0.516	-0.363	-0.085
p(t)	1	1	1	1	1	1	1
gs(kTs)	0	-0.242	-0.435	-0.536	0.516	-0.363	-0.085
gs1(kTs)	0.125	-0.125	-0.375	-0.625	-0.625	-0.375	-0.125
Enc	1000	0111	0110	0101	0101	0110	0111

Inferences :

1. At the time stamps in 1b and 1c, the value of p(t) is 1 for both cases. As a result, there is no change in the value of the quantized signal.

2. For representing the quantization levels, we need 4 bits, as seen in the encoding signal.

Appendix

A Matlab Commands

Table 3: Matlab commands used in this lab.

Matlab Command	Function
`plot(`x, y`)`	plots values of the simulation series y along the y-axis, with values of the simulation series x along the x-axis.
`figure()`	creates a new figure in MATLAB.
`title(x)`	adds a title x to the plot
`xlabel(x)`	adds a horizontal label x (along x axis) to the plot
`ylabel(x)`	adds a vertical label x (along y axis) to the plot
`grid on`	adds a grid to the plot.
`clc`	clears everything from the matlab command line window.
`linspace(x1,x2,p)`	generates p equally distant points between x1 and x2.
`subplot(abc)`	generates a subplot of size a x b, and the current image is of index c
`ones(length)`	creates an array of 1s.
`size`	gives length of an array
`xticks`	marks the ticks in x axis
`yticks`	marks the ticks in y axis

B Matlab Code

Matlab codes for each part.

B.1 Q1(a)

```
% question1(a)
```

```
clc;
close all;

% message signal
Am =1;
fm1 = 1;
fm2 = 3;
t = 0:0.02:1;

ym = Am*(sin(2*pi*fm1*t)-sin(2*pi*fm2*t));
peaku=round(max(ym));
peakl=round(min(ym));
figure(1);
subplot(3,1,1);
plot(t,ym);
title("Message signal in time domain");
xlabel("time(sec)");
ylabel("Amplitude(Volt)");
grid on;

%impulse train
t=0:0.02:1;
y = ones(length(t));
subplot(3,1,2)
stem(t,y);
title("impulse train");
xlabel("t");
ylabel("Amplitude(volt)");
grid on;

%sampled signal
ym =  [ym(1,:)];
y = [y(1,:)];

gs = zeros(1,length(y));
for i=1:length(y)
    gs(i) = y(i)*ym(i);
end

subplot(3,1,3);
stem(t,gs);
```

```
title("impulse train multiplied to message signal");
xlabel("t");
ylabel("Amplitude(volt)");
grid on;

% superimposed signal
l = 16;
peak = max(abs(ym));
delta = 2*peak/16;

gkts=zeros(1,length(gs));

%defining the quantised signal
k=(peaku-peakl)/16;
Vmax=peaku-k/2;
Vmin=peakl+k/2;
for i=Vmin:k:Vmax
    for j=1:length(ym)
        if((i-k/2)<gs(j) && (gs(j)<(i+k/2)) && ...
                gs(j) ~= 0)
            gkts(j)=i;
        end
        if(gs(j) == 0)
            gkts(j)=0.125;
        end
    end
end

figure(2);
plot(t,ym,'r');
hold on;
stem(t,ym,'o');
hold on;
stem(t,gkts,'b');
legend('message','sampled','quantized');
title("Superimposed signal");
xlabel("t");
ylabel("Amplitude(volt)");
yticks([-2 -1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25 0 0.25
0.5 0.75 1 1.25 1.5 1.75 2]);
grid on;
```

B.2 Q1(b)

```
%1b
clc;
close all;

%message signal
fa=1;
fb=3;
fs=50;
t=0:1/fs:1;
ym=sin(2*pi*fa*t)-sin(2*pi*fb*t);
peaku=round(max(ym));
peakl=round(min(ym));

%impulse train
y=zeros(size(t));
y(1:1:end)=1;

%sampled signal
ym =  [ym(1,:)];
y = [y(1,:)];

gs = zeros(1,length(y));
for i=1:length(y)
    gs(i) = y(i)*ym(i);
end

%Initialise zeros array
gskt=zeros(1,length(gs));

%defining the quantised signal
k=(peaku-peakl)/16;
Vmax=peaku-k/2;
Vmin=peakl+k/2;
for i=Vmin:k:Vmax
    for j=1:length(ym)
        if((i-k/2)<=gs(j) && (gs(j)<=(i+k/2)) && ...
                gs(j) ~= 0)
            gskt(j)=i;
        end
```

```
        if(gs(j) == 0)
            gskt(j)=0.125;
        end
    end
end

% pulse train
ts = 1/fs;
tp = ts;
w = zeros(1,length(gskt));
for i=1:length(w)
    w(i) = 1;
end
y2=w;

%creating g_h signal
gq=y2.*gskt;

subplot(2,1,1);
plot(t,ym,'b');
hold on;
plot(t,y2,'r');
title('Superposition of message signal and pulse train');
xlabel('Time(sec)');
ylabel('Magnitude');
xticks(t);
yticks([-2 -1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25 0 0.25
0.5 0.75 1 1.25 1.5 1.75 2]);
legend('g(t)','p(t)');
grid on;

subplot(2,1,2);
plot(t,ym,'b');
hold on;
stairs(t,gq,'r');
yticks([-2 -1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25 0 0.25
0.5 0.75 1 1.25 1.5 1.75 2]);
grid on;
title('quantised signal multipled to pulse train vs message signal');
xlabel('Time(sec)')
ylabel('Magnitude')
```

```
xticks(t);
yticks([-2 -1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25 0 0.25
0.5 0.75 1 1.25 1.5 1.75 2]);
legend('g(t)','g_q(t)')
grid on;
```

B.3 Q1(c)

```
%1c
clc;
close all;

%message signal
fa=1;
fb=3;
f=50;
n_samples=100;
fs=n_samples*f;
t=0:1/fs:1;
ym=sin(2*pi*fa*t)-sin(2*pi*fb*t);
peaku=round(max(ym));
peakl=round(min(ym));

%impulse train
y=zeros(size(t));
y(1:n_samples:end)=1;

%sampled signal
ym =  [ym(1,:)];
y = [y(1,:)];

gs = zeros(1,length(y));
for i=1:length(y)
    gs(i) = y(i)*ym(i);
end

%quantized signal;
gkts=zeros(1,length(gs));
k=(peaku-peakl)/16;
```

```
Vmax=peaku-k/2;
Vmin=peakl+k/2;
for i=Vmin:k:Vmax
    for j=1:length(gs)
        if((i-k/2)<gs(j) && (gs(j)<(i+k/2)))
            gkts(j)=i;
        end
    end
end

%pulse train
w = 0.01;
delay = w/2:w*2:10;
y2=pulstran(t,delay,'rectpuls',w);
delay1 = [t;y2.*gkts;]';
y3 = pulstran(t,delay1,'rectpuls',w);

current = 0.125;
for c = 1:length(y3)
    if(y2(c)==0)
        y3(c) = 0;
    elseif y3(c)==0
        y3(c) = current;
    else
        current = y3(c);
    end
end

% Plots
figure(1);
subplot(2,1,1);
plot(t,ym);
hold on;
plot(t,y2,'r');
title('message signal and pulse train');
xlabel('Time(sec)');
ylabel('Magnitude');
yticks([-2 -1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25 0
0.25 0.5 0.75 1 1.25 1.5 1.75 2]);
xticks(0:0.02:1);
legend('g(t)','p(t)');
```

```
grid on;

subplot(2,1,2);
plot(t,ym);
hold on;
stairs(t,y3,'r');
yticks([-2 -1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25
0 0.25 0.5 0.75 1 1.25 1.5 1.75 2]);
grid on;
title('message train and quantised signal');
xlabel('Time(sec)');
ylabel('Magnitude');
xticks(0:0.02:1);
yticks([-2 -1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25
0 0.25 0.5 0.75 1 1.25 1.5 1.75 2]);
legend('g(t)','g_q(t)');
grid on;
```

References

[1] IIT Mandi lectures on EE304 offered by Dr Adarsh `https://cloud.iitmandi.ac.in/d/4bb3a5f304334160ab67/`

[2] Tutorialspoint lectures on quantisation `https://www.tutorialspoint.com/digital_communication/digital_communication_quantization.htm`

[3] Wikipedia notes on quantisation for signal processing `https://en.wikipedia.org/wiki/Quantization_(signal_processing)`

[4] Digital quantisation on Youtube `https://youtu.be/cUdnGkymAWU`

[5] Advantages, disadvantages and application of PCM (pulse code modulation) `https://www.polytechnichub.com/advantages-disadvantages-application-pcm-pulse-code-modulation/`