American International University- Bangladesh

Department of Computer Engineering

COE 3201: Data Communication Laboratory

Title: Analog Signal quantization using MATLAB

Abstract:

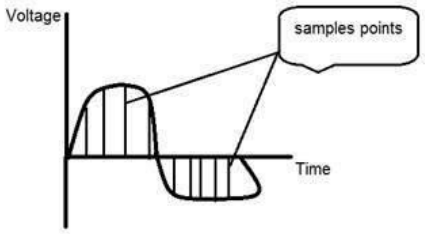
This experiment is designed to-

1.To understand the use of MATLAB for solving communication engineering problems.
2.To develop understanding of Quantization concept and how to implement it in Matlab.

Introduction:

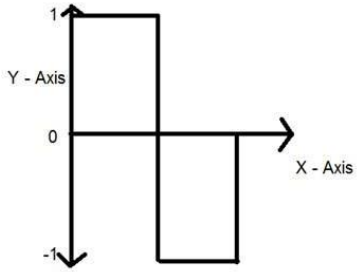
Digitizing a signal: As we have seen in the previous tutorials, that digitizing an analog signal into a digital, requires two basic steps. Sampling and quantization. Sampling is done on x axis. It is the conversion of x axis (infinite values) to digital values.

The below figure shows sampling of a signal.

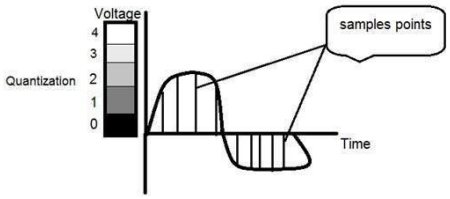


Quantization: Quantization is opposite to sampling. It is done on y axis. When you are quantizing an signal, you are actually dividing a signal into quanta(partitions).

On the x axis of the signal, are the co-ordinate values, and on the y axis, we have amplitudes. So digitizing the amplitudes is known as Quantization.



The signal has been quantified into three different levels (-1,0,1). That means that when we sample a signal, we actually gather a lot of values, and in quantization, we set levels to these values. This can be more clear in the image below.



In the figure shown in sampling, although the samples have been taken, but they were still spanning vertically to a continuous range of gray level values. In the figure shown above, these vertically ranging values have been quantized into 5 different levels or partitions. Ranging from 0 black to 4 white. This level could vary according to the type of signal you want.

Quantization is a non linear transformation which maps elements from a continuous set to a finite set. It is also the second step required by A/D conversion.

Types of quantization:

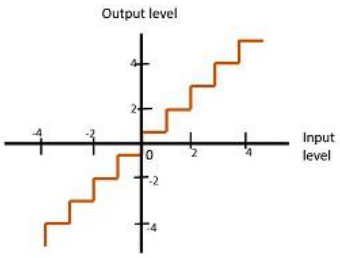


Fig 1 : Mid-Rise type Uniform Quantization

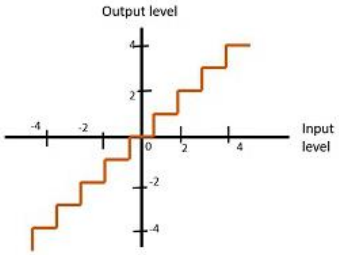


Fig 2 : Mid-Tread type Uniform Quantization

A unipolar quantizer deals with analog signals ranging from 0 volt to a positive reference voltage, and a bipolar quantizer deals with analog signals ranging from a negative reference to a positive reference. The notations and general rules for quantization (step size Δ) are as follows:

$$\Delta = \frac{(x_{\max} - x_{\min})}{L}$$

$$L = 2^m$$

m= number of bits

$$i = round\left(\frac{x - x_{\min}}{\Delta}\right)$$

$$x_q = x_{\min} + i\Delta \quad i = 0, 1, \cdots, L-1$$

where x_{max} and x_{min} are the maximum value and minimum values, respectively, of the analog input signal x. The symbol L denotes the number of quantization levels, where m is the number of bits used in ADC. The symbol Δ is the step size of the quantizer or the ADC resolution. Finally, xq indicates the quantization level, and i is an index corresponding to the binary code.

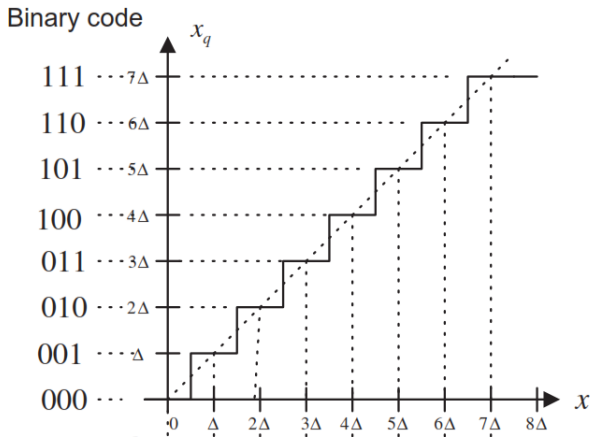
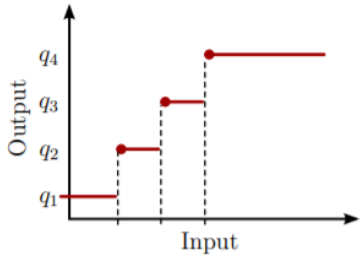


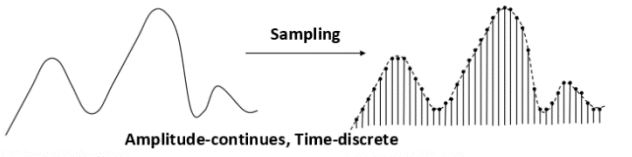
Fig. : Characteristics of the unipolar quantizer.

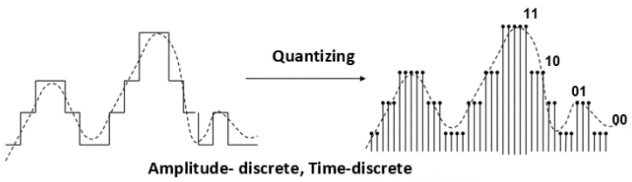
Let Q : R 7→ Q be a quantizer with four possible outputs labeled q1 through q4. The output xq of the quantizer belongs to set Q, and it must therefore be equal to one of the four possible points listed above. Figure shows the functional representation of a four-level quantization scheme.



The Figure 3.1: This is a functional representation of a quantizer where the input is converted to one of four possible values. output of the quantizer in this example is determined according to a nearest neighbor rule and this implies that

$$Q(x) = \begin{cases} q_1 & \text{if } x < \frac{q_1+q_2}{2} \\ q_2 & \text{if } \frac{q_1+q_2}{2} \le x < \frac{q_2+q_3}{2} \\ q_3 & \text{if } \frac{q_2+q_3}{2} \le x < \frac{q_3+q_4}{2} \\ q_4 & \text{if } x \ge \frac{q_3+q_4}{2} \end{cases}.$$





Example of an analog signal quantization in MATLAB:

```
Am=4;  %setting the sine wave amplitude
bit=3;   % setting the number btis
f=1;      %sine wave frequency
fs=30;  % setting sampling frequency
t=0:1/fs:1*pi;  %defining the signal duration for plotting
y=Am*sin(2*pi*f*t);   %sampled sine wave y(30 samples in a cycle)
Nsamples=length(y);   %calculating the total number of samples
quantised_out=zeros(1,Nsamples);   %making an array of size=number of samples midrise
del=2*Am/(2^bit);   %determining the step size
Llow=-Am+del/2;
Lhigh=Am-del/2;

for i=Llow:del:Lhigh   %Iterating from lowest to the highest levels

    for j=1:Nsamples   %taking the whole sampled vector
     if(((i-del/2)<y(j))&&(y(j)<(i+del/2)))       % Quantizing the sampled value
       quantised_out(j)=i;                        % to the quantization level if it

     end                        % lies within the bound of -del/2 and
    end                         % del/2 of the level.
end
stem(t,quantised_out); %plotting wave forms.
hold on;
plot(t,y,'color','r');
```

Another example quantization in MATLAB

```
fs=40e3;% sampling frequency

fc=4e3;% frequency of the signal

t=0:1/fs:0.001;%discrete time

x=0.5*sin(2*pi*fc*t);% discrete signal


%--------Quantization----------%

n=8;

L=(2^n)-1;

delta=(max(x)-min(x))/L;

xq=min(x)+(round((x-min(x))/delta)).*delta;

%--------END----------%
```

```
subplot(2,1,1)
stem(t,x,'R');
subplot(2,1,2);% breaking the window figure to plot both graphs
stem(t,x,'b');% plot of discrete time signaltitle('Discrete time representation')% title of the figure
xlabel('time(s)')% label on the x-axis of the plot
ylabel('X[n]')% label on the y-axis of the plot
subplot(2,1,2);
stairs(t,xq,'b');% the quantized output
title('Quantized Signal')% title of the figure
xlabel('time')% label on the x-axis of the plot
ylabel('amplitude')% label on the y-axis of the plot
```

Performance Task for Lab Report: (your ID = AB-CDEFG-H)

**Generate aN analog signal using the following equation,

$x_1(t) = A_1 \cos(2\pi(CDE*100)t)$

(a) Select the value of the amplitudes as follows: let A_1 = GD and A_2 = AF.

(b) Assuming that a 4-bit ADC channel accepts analog input ranging from 0 to 5 volts, determine

I. the number of quantization levels;
II. the step size of the quantizer or resolution;
III. the quantization level when the analog voltage is 3.2 volts;
IV. Implement it in MATLAB