รายงาน

เรื่อง Detection และ Quantization

วิชา ปฏิบัติการระบบโทรคมนาคม (Communication System Lab)

เสนอ

อาจารย์ ผศ. สิทธิพร เกิดสำอางค์

จัดทำโดย

นายโสภณ สุขสมบูรณ์ รหัสนักศึกษา 6201011631188

นักศึกษาชั้นปีที่3 สาขาวิชาวิศวกรรมไฟฟ้า (โทรคมนาคม)

ปฏิบัติการครั้งที่ 2

วิชา ปฏิบัติการระบบโทรคมนาคม ประจำภาคการศึกษา 2/2564

สาขาวิชาวิศวกรรมไฟฟ้า(โทรคมนาคม) ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

Experiment 4

Detection

Procedure

A. Characteristics of Matched Filters

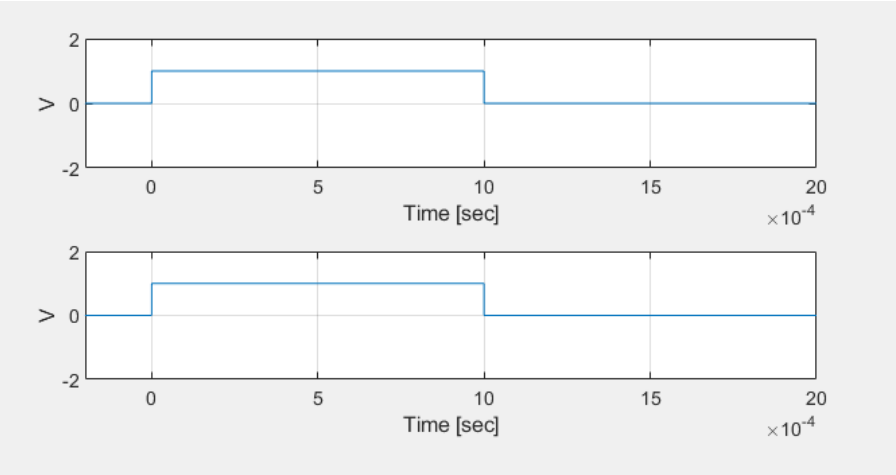
A.1 Generate a rectangular pulse with unit pulse amplitude and 1 msec pulse duration .

```
>> r=wave_gen(1,'polar_nrz',1000);
```

A.2 Display r and the impulse response of a matched filter based on r .

```
>> subplot(311),waveplot(r)
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 90)
>> subplot(312),match('polar_nrz')
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 90)
In match (line 90)
>>
```

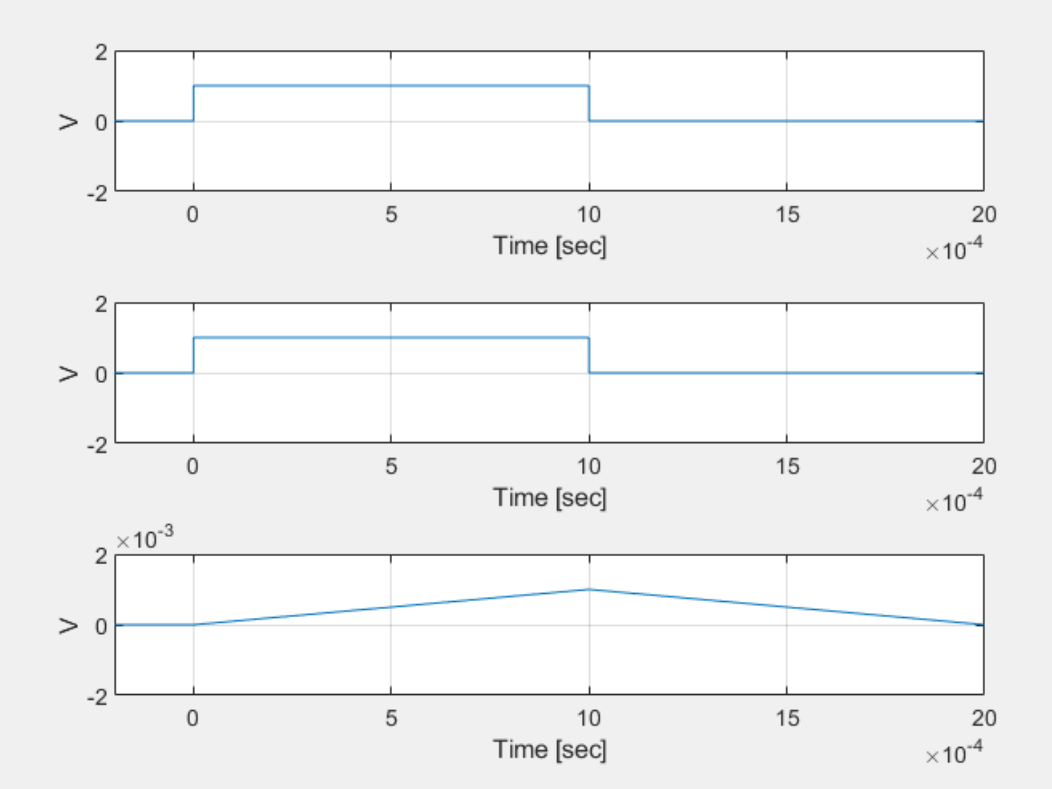
Wave plot



A.3 Observe the matched filter output if r is applied to its input .

```
>> rm=match('polar_nrz',r);
>> subplot(313),waveplot(rm)
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 122)
>>
```

Wave plot



Determine the time when the filter output reached its maximum value . How is this time related to the waveform r ?

Time = $10\ x\ 10^{-4} sec$ *หรือ 1 msec*

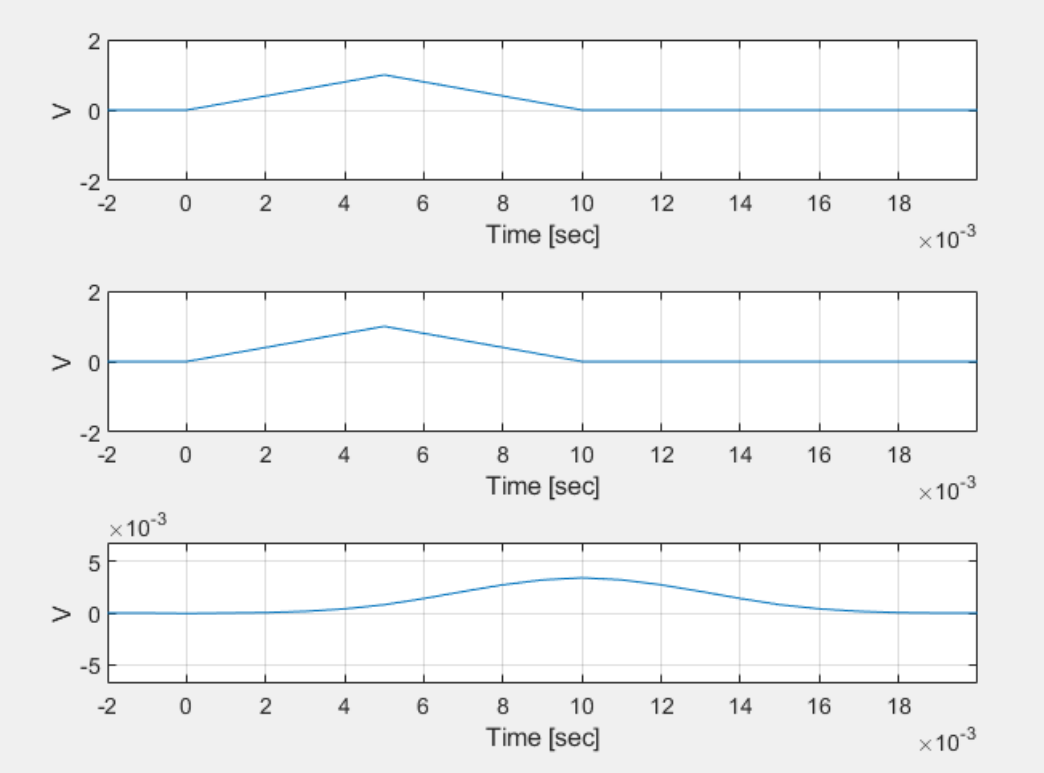
Q4.1 How would you determine the peak amplitude value of the matched filter output directly from the above plots ?

คำนวณค่า Peak Amplitude จากสมการ $Y = y(T) = \int_0^T g(T-\tau)r(\tau)d\tau$

A.4 Repeat Part A.1 – A.3 for a triangular pulse with 10 msec pulse width and unit peak amplitude .

```
>> r= wave_gen(1,'triangle',100);
>> clf; subplot(311),waveplot(r)
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 122)
>> subplot(312),match('triangle')
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 122)
In match (line 90)
>> rm=match('triangle',r);
>> subplot(313),waveplot(rm)
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 122)
>>
```

Wave plot



Q4.2 If the triangular pulse width is changed to 1 msec , determine the peak amplitude of the matched filter output ?

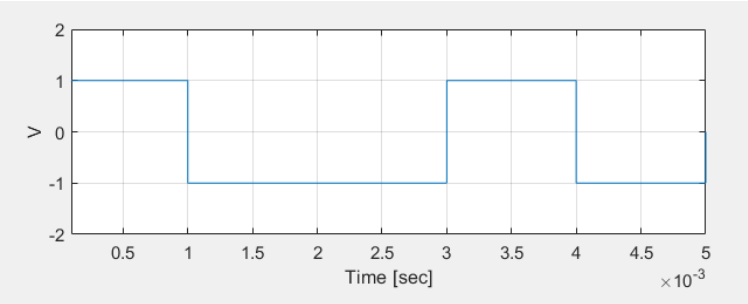
ค่า Peak Amplitude ที่ได้จากการทำ Matched filter เท่ากันกับที่ 1 msec แต่เวลาที่เข้าสู่ Peak สูงสุดต่างกัน

A.5 Repeat parts A.1-A.3 for a Manchester pulse with 10 msec pulse width and unit peak amplitude . Predict the matched filter impulse response and matched filter output . Verify your predictions using MATLAB function.

A.6 Generate a polar NRZ waveform that represents the 5-sample binary sequence [1 0 0 1 0] . The binary data rate Rb is 1 kbps and the pulse amplitude A is 1 V.

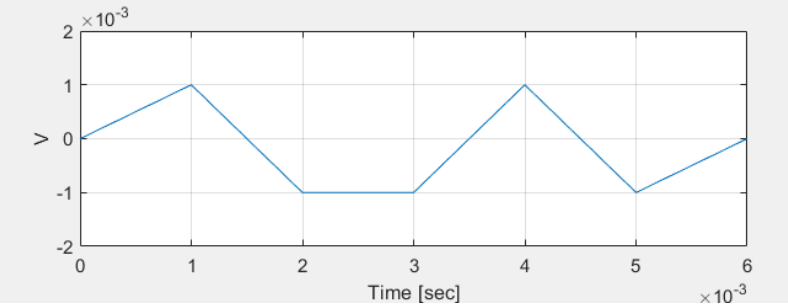
```
>> x5=wave_gen([1 0 0 1 0],'polar_nrz',1000);
>> clf,subplot(211),waveplot(x5)
```

Wave plot



A.7 Apply x5 to a matched filter . Record the matched filter output in Graph Using the amplitude scale shown on the right .

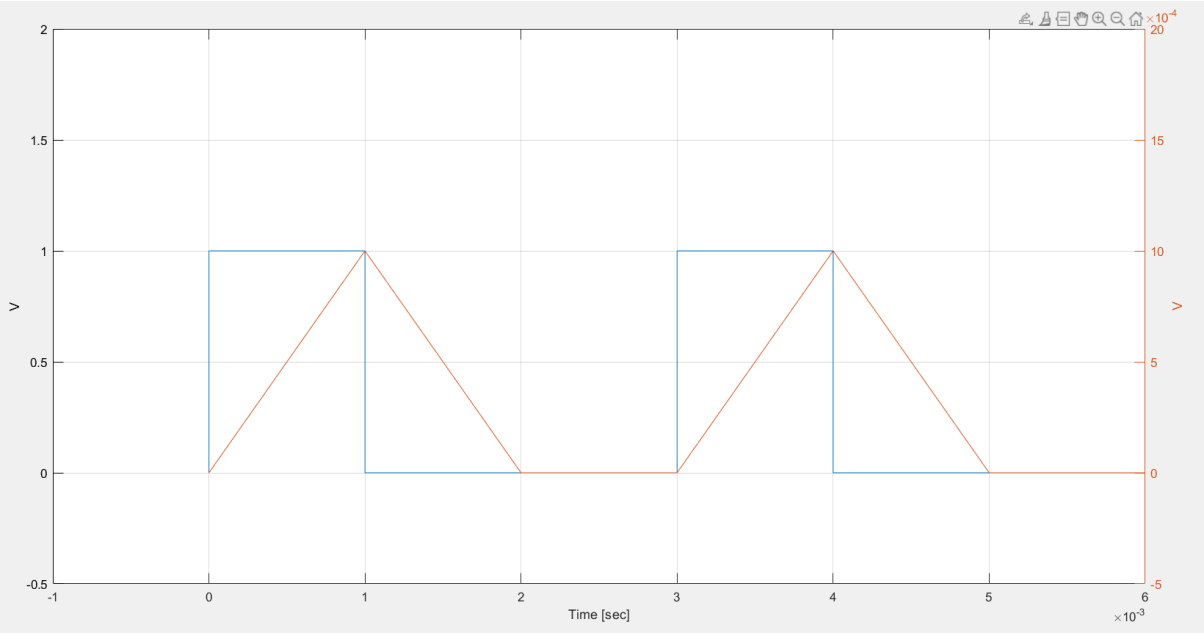
```
>> subplot(212),waveplot(match('polar nrz',x5))
```



Q4.3 Construct the waveform at a matched filter output if the input is a unipolar NRZ waveform that represents the binary sequence [1 0 0 1 0]

```
>> x5=wave_gen([1 0 0 1 0],'unipolar_nrz',1000);

>> clf,waveplot(x5)

>> axis([-0.001 0.006 -0.5 2]) , hold on ;

>> yyaxis right

>> axis([-0.001 0.006 -0.0005 0.002])

>> waveplot(match('unipolar_nrz',x5))
```

Wave plot

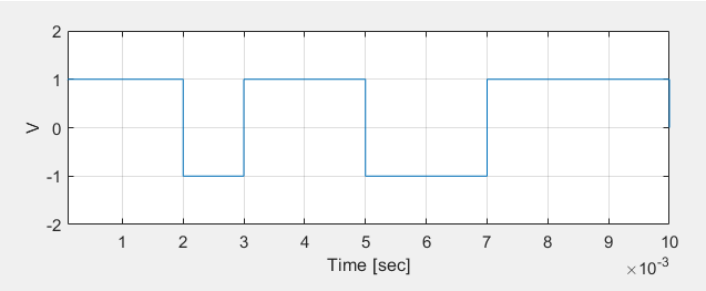


B. Signal Detection

B.1 Generate a 10-sample binary sequence and a waveform that represents this binary sequence in polar NRZ signaling format .

```
>> b10=binary(10);
>> x10=wave_gen(b10,'polar_nrz',1000);
>> clf,subplot(211),waveplot(x10)
```

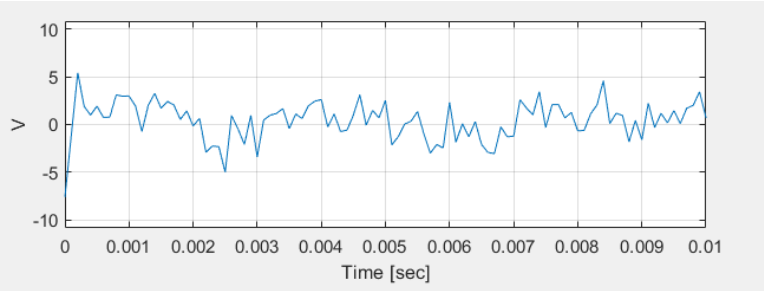
Wave plot



B.2 Apply x10 to a channel with 4.9 kHz bandwidth and AWGN where the noise power is 2 W . Display the channel output waveform y10 :

```
>> y10=channel(x10,1,2,4900);
>> subplot(212),waveplot(y10)
```

Wave plot



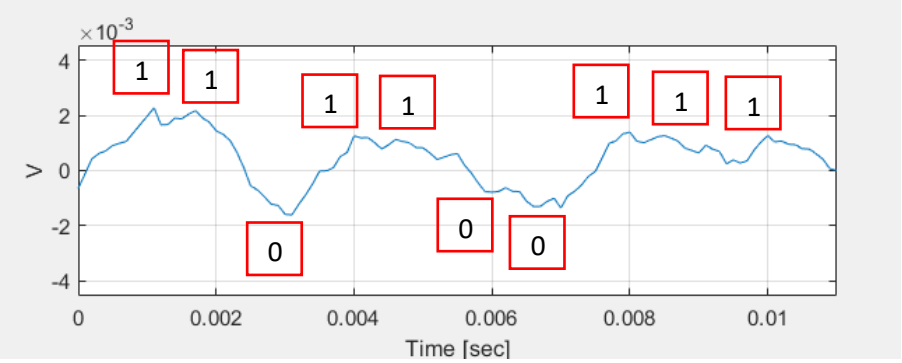
Decode the binary sequence from the waveform y10 :

Binary Sequence : 1 1 0 1 1 0 0 1 1 1

B.3 Apply y10 to a matched filter . Display the output waveform z10 :

```
>> z10 = match('polar_nrz',y10);
>> subplot(212),waveplot(z10)
```

Wave plot



Let T_b be the binary data period. Sample the output of the matched filter at kT_b, $k = 1, \ldots, 10$ and apply the following decision rule:

$$\widehat{b_k} = \begin{cases} 0, & \text{if} \quad \texttt{z10}(kT_b) \leq 0; \\ 1, & \text{if} \quad \texttt{z10}(kT_b) > 0; \end{cases}$$

where $\widehat{b_k}$ is the estimated value of the k*th* element of the binary sequence `b10`. Apply this decision rule on the matched filter output `z10`:

b10 = 1 1 0 1 1 0 0 1 1 1

Compare your estimate with the original sequence b10 :

1x10 double

	1	2	3	4	5	6	7	8	9	10	1
1	1	1	0	1	1	0	0	1	1	1	
2											

Q 4.4 Comment on whether it easier to decode the transmitted binary sequence directly from the channel output y10 or from the matched filter output y10 or from the matched filter output z10 .if sampling instants other than those specified above are used, the probability of making a decoding error will be larger why ?

การถอดรหัสโดยใช้ Matched filter เป็นวิธีการที่ง่ายกว่า binary sequence from channel เนื่องจากหากสังเกตคร่าวๆจาก wave plot การทำด้วย Matched filter ทำได้ง่ายกว่า ความComplex ของกราฟลดลง สามารถแยก sequence ได้ง่ายกว่าการถอดรหัสจาก channel output โดยตรง

C. Matched-Filter Receiver

C.1 Generate a 2000-sample binary sequence b and a polar NRZ waveform based on b :

```
>> b=binary(2000);
>> x=wave_gen(b,'polar_nrz');
>>
```

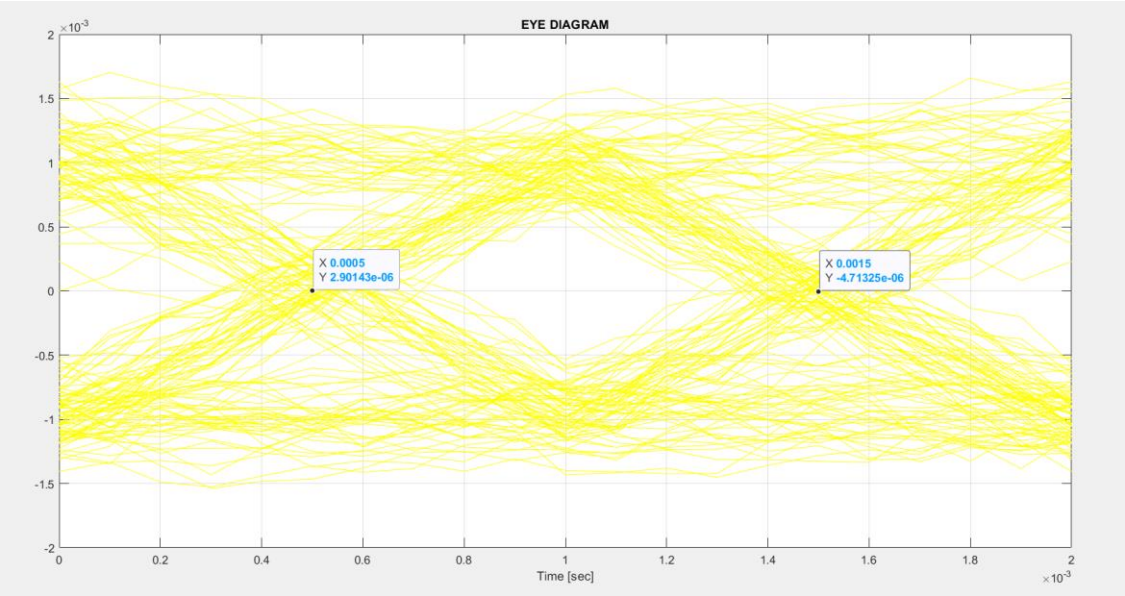
Apply x to a channel with 4.9 kHz bandwidth and AWGN where the noise power $\sigma_n{}^2$ is 0.5 W . Let y be the channel output waveform .

```
>> y=channel(x,1,0.5,4900);
>>
```

C.2 Apply y to a matched filter. Display the eye diagram of the matched filter output z .

```
>> z=match('polar_nrz',y);
>> clf,eye_diag(z);
>>
```

wave plot



From the eye diagram, determine the optimum sampling instants and threshold value `v_th` for the detector to decode the transmitted binary sequence `b`. Sampling instants for the matched filter output are measured with respect to the time origin. For example, if the binary data period is T_b and the `sampling_instant` parameter is set to t_i, then the detector will sample the signal at t_i, $t_i + T_b$, $t_i + 2T_b$, ... etc.

V_th= 0

Sampling_instant = 0.0015-0.0005 = 0.001

Use `v_th` and `sampling_instant` in the detector which will operate on the matched filter output. Record the resulting probability of bit error P_e (also referred to as bit error rate (BER)) in Table 4.1.

$\sigma_n{}^2[W]$	P_e - empirical	P_e – theoretical
0.5	0.000000	0.000051
1.0	0.003500	0.000102
1.5	0.004000	0.000153
2.0	0.015000	0.000204

C.3 Repeat C.1–C.2 for channel noise power of 1, 1.5, and 2 W without displaying the eye diagram of the matched filter output `z`. Record P_e results in Table 4.1. **Remark:** In Experiment 3 you have observed that the optimum sampling instants and the threshold value are independent of channel noise power. Therefore, you can use the optimum sampling instants determined in part C.2 to decode the matched filter output for different channel noise power levels.

C.4 If different sampling instants other than the optimum values are used, the resulting BER will be larger. You can observe this by decoding the binary sequence using values for the `sampling_instant` parameter that are 0.9 and 0.5 times the optimal value used in part C.3.

Sampling_instant	0.001	0.9	0.5
$\sigma_n{}^2[W]$	P_e - empirical	P_e - empirical	P_e - empirical
0.5	0.000000	0.0005	0.275
1.0	0.003500	0.003	0.27
1.5	0.004000	0.0105	0.266
2.0	0.015000	0.0275	0.266

Q4.5 Evaluate theoretical probability of bit error values for all cases considered above and record in Table 4.1. Note that the PSD function of a white noise process can be determined as:

$$S_n(f) = \frac{N_o}{2} = \frac{\sigma_n^2}{2 \times \text{system bandwidth}},$$

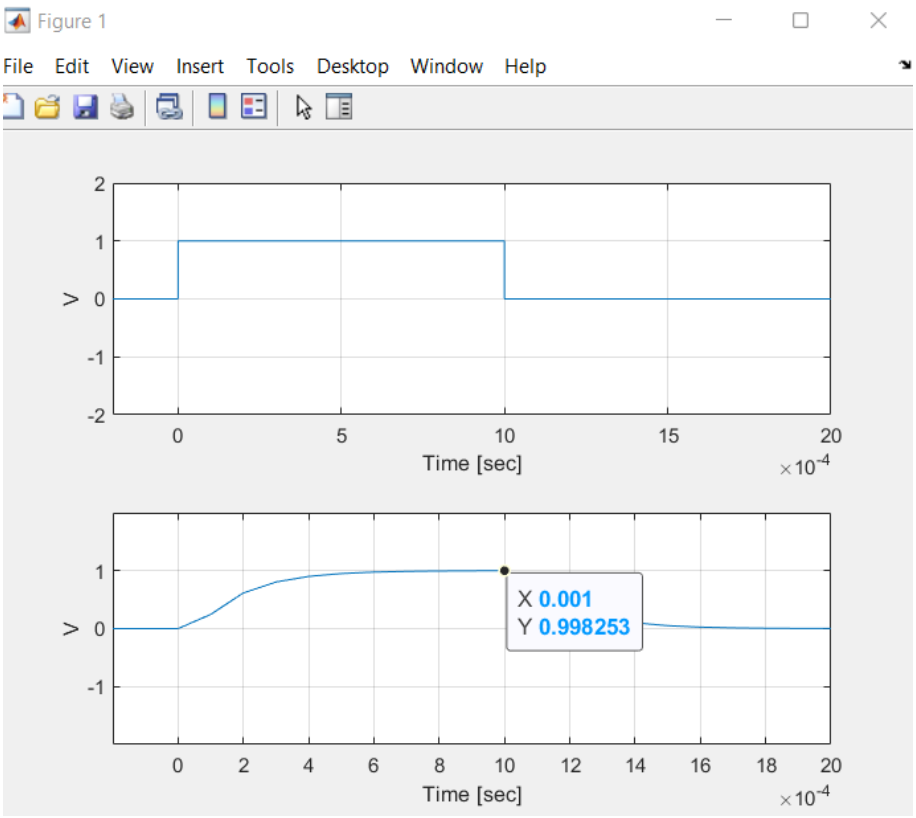
where the system bandwidth in this experiment is 5 kHz.

D . Low-Pass Filter Receiver

D.1 Apply a rectangular pulse to a first-order RC-filter of 1 kHz bandwidth. Display the filter output and measure the peak amplitude A_r:

```
>> r=wave_gen(1,'unipolar_nrz'); r_lpf=rc(1000,r)
>> subplot(211);waveplot(r)
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 90)
>> subplot(212);waveplot(r_lpf);
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 122)
>>
```

Wave plot



Ar = 0.998253 V

D.2 Generate 2,000 samples from a zero-mean white noise sequence of 0.5 W power. Apply the noise sequence to the RC-filter. Record the rms value of the output noise power.

```
>> n=gauss(0,0.5,2000);
>> meansq(rc(1000,n))

ans =

    0.1333

fx >>
```

$$\sigma_n{}^2 = 0.1333\ W$$

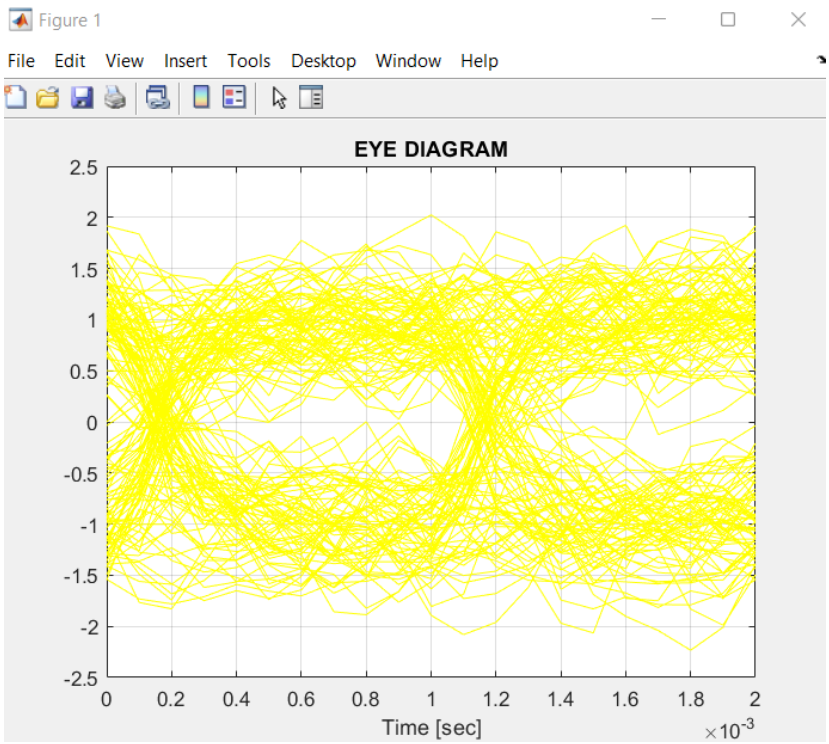
Q4.6 From the results in parts D.1 and D.2, determine the ratio A_r/σ_n, where A_r is the peak signal amplitude measured in D.1 and σ_n is the rms value of the output noise. If y in part C.1 is applied to a receiver which uses the above RC-filter, determine the resulting BER.

$$\frac{A_r}{\sigma_n} = \frac{0.998253}{\sqrt{0.1333}} = 2.73417$$

D.3 Regenerate y from part C.1. Apply y to the RC-filter. Display the eye diagram of the output waveform z_lpf.

```
>> b=binary(2000);
>> x=wave_gen(b,'polar_nrz');
>> y=channel(x,1,0.5,4900);
>> z_lpf=rc(1000,y);
>> clf,eye_diag(z_lpf);
fx >>
```

Wave plot



D.4 From the eye diagram, determine the optimum sampling instant and threshold value. Decode the binary sequence form `z_lpf`.

```
≫ detect(z_lpf,v_th,sampling_instant,b);
```

Compare the resulting BER with the BER evaluated in step C.2.

D.4

```
>> detect(z_lpf,0,0.001,b);
Probability of bit error (BER) =    0.002500.
>>
```

C.2

```
>> detect(z,0,0.001,b);
Probability of bit error (BER) =    0.000000.
>>
```

D.5 Repeat part D.4 for the channel noise power of 1, 1.5, and 2 W. Record results in Table 4.2.

$\sigma_n{}^2[W]$	P_e	
	BW = 1.0 kHz	BW=0.5 kHz
0.5	0.0035	0
1.0	0.0315	0.011
1.5	0.065	0.022
2.0	0.085	0.053

D.6 Repeat parts D.3 – D.5 for a first-order RC-filter with 500 Hz bandwidth. Record the resulting BER values in Table 4.2.

```
≫ z_lpf = rc(500,y);
≫ eye_diag(z_lpf)
≫ detect(z_lpf,v_th,sampling_instant,b);
```

Q4.7 Explain why the BER resulting from a low-pass filter of 500 Hz bandwidth is smaller than the BER resulting from a low-pass filter of 1 kHz bandwidth. Will the BER be further decreased if a low-pass filter of 100 Hz bandwidth is used?

เพราะเป็นความถี่ที่ให้ผลลัพธ์ใกล้เคียงกับการทำ Matched filter ที่ช่วง BW 100 Hz ค่า BE ไม่ได้ลดลงจากค่าเดิมที่ช่วง BW 500 Hz

E . Implementation of Matched Filter Structure

Integrate-and-dump filter is a practical circuit implementation of a matched filter.

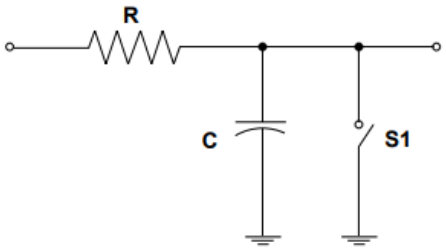
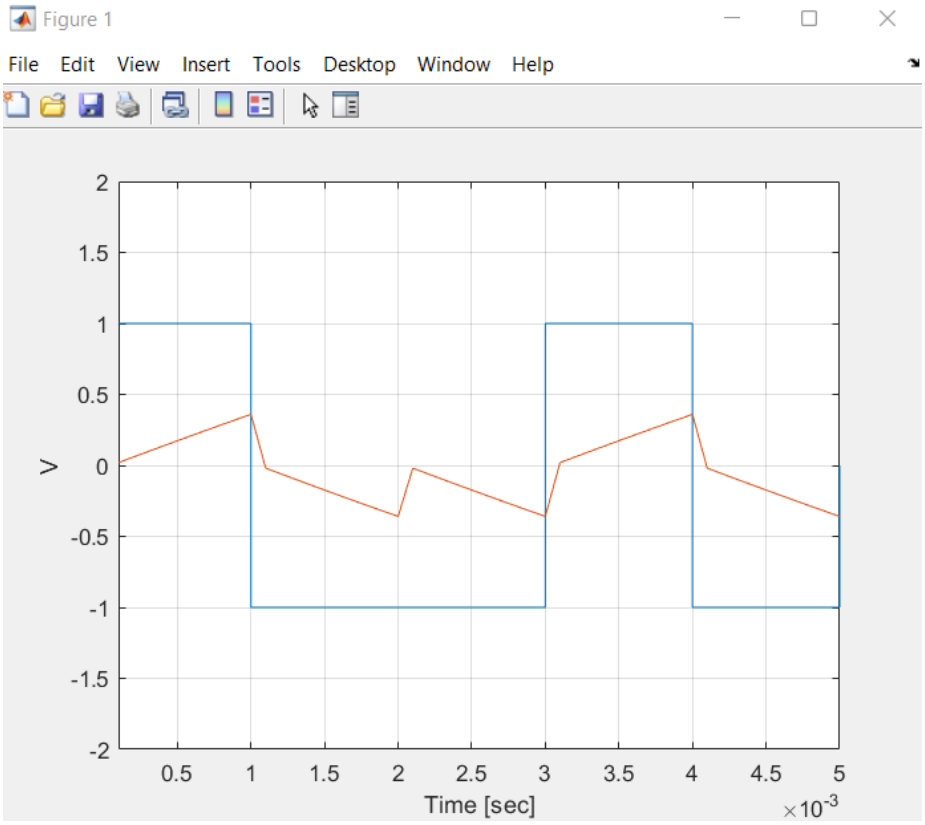


Fig 4.1 Matched filter implementation

In the circuit, the switch S1 is closed periodically for a short time to discharge the capacitor. The RC network with a very large time constant acts as an integrator.

E.1 To understand the operation of the filter, apply `x5` generated in part A.6 to the filter and observe both input and output waveforms.

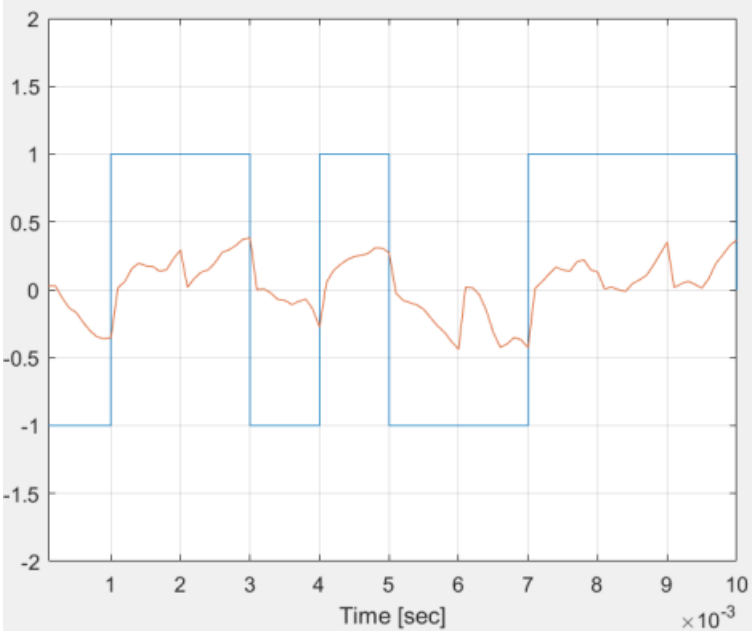
```
≫  y5 = int_dump(x5);
≫  clf; waveplot(x5); hold on; waveplot(5*y5)
```



Q4.8 In Graph 4.2 indicate when the capacitor is charging and discharging. When is the switch S1 closed? Can you specify the optimum sampling instants without studying the eye diagram?

เมื่อ Capacitor เริ่มชาร์จประจุไฟฟ้าจะเป็นช่วง oc หรือ Open Circuit (สวิตช์เปิด) ซึ่งกราฟจะค่อยๆสูงขึ้น และเมื่อ Capacitor คลายประจุ จะเป็นช่วง sc หรือ Short Circuit (สวิตช์ปิด) ซึ่งจะทำให้กราฟค่อยๆลดลง

E.2 Repeat parts D.3 – D.4, but use the integrate-and-dump filter instead of the low-pass filter. Compare the resulting BER with those resulting from a low-pass filter and a matched filter.



Experiment 5

Quantization

Procedure

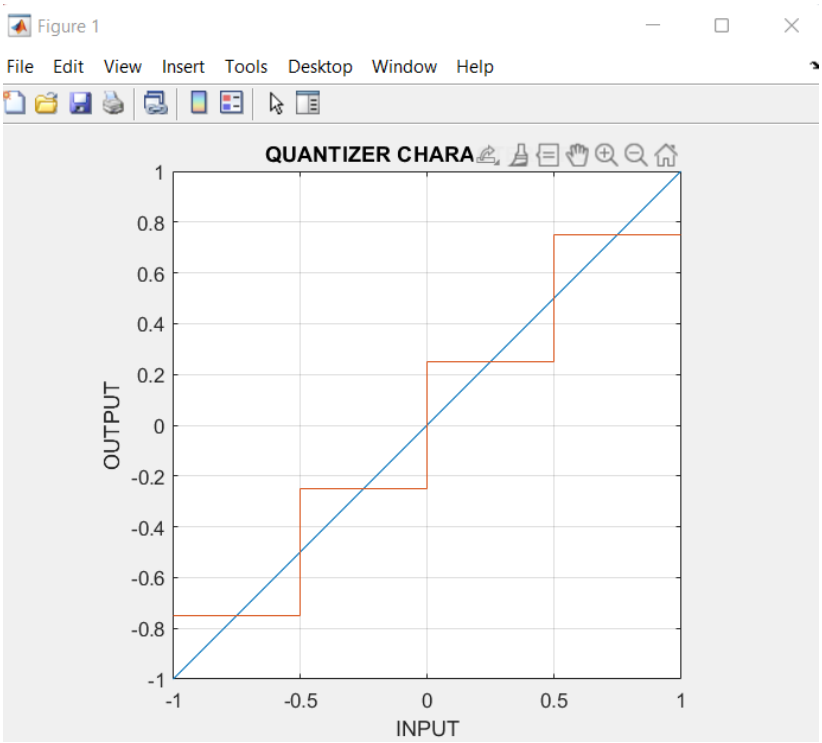
A. Uniform Quantization

After a message has been sampled, the next step is to quantize the samples. Because an analog signal has a continuous range of amplitudes, pulses in the sampled signal will have an infinite number of amplitude levels. It is not possible to assign unique, finite length code words to an infinite number of amplitudes. Quantization consists of approximating the sampled signal by a signal made up of discrete amplitudes.

A.1 Display the input-to-output mapping characteristic of a 2-bit uniform quantizer:

```
>> quant_ch(2,'uniform')
>>
```

Wave plot



A.2 Let x be the analog pulse amplitude, and x_q be the quantizer output, if x is the input. Assume that pulse amplitude x is normalized such that $|x| \leq 1$. From Graph 5.1 derive the quantizer mapping rule:

Quantization Interval	Xq	Binary Code
-1 to -0.5	-0.75	00
-0.5 to 0	-0.25	01
0 to 0.5	0.25	10
0.5 to 1	0.75	11

Observe the length of each interval that appears in the first column of Table 5.1. How is the quantization interval length related to the number of quantization levels of a uniform quantizer? The entries in the $2nd$ column are referred to as quantization levels; only these values appear at the quantizer output.

A.3 Generate the 8 element sampled sequence x where each element of x represents the analog pulse amplitude at that sampling instant:

```
>> x=[0.8,0.6,0.2,-0.4,0.1,-0.9,-0.3,0.7];
>>
```

Apply quantization rule described in Table 5.1 to sequence **x**:

Element	X	Xq	Binary Code
1	0.8	0.7500	11
2	0.6	0.7500	11
3	0.2	0.2500	10
4	-0.4	-0.2500	01
5	0.1	0.2500	10
6	-0.9	-0.7500	00
7	-0.3	-0.2500	01
8	0.7	0.7500	11

Compare results obtained from applying the quantizer mapping rule with the output from the MATLAB function *quantize*:

```
>> xq=quantize(x,2)

xq =

  Columns 1 through 4

    0.7500    0.7500    0.2500   -0.2500

  Columns 5 through 8

    0.2500   -0.7500   -0.2500    0.7500
```

A.4 Devise a rule which assigns a unique 2-bit binary symbol to each quantization level. Use a natural code[1] which assigns the binary symbol

"00" to the most negative quantization level (-0.75 in this example). As you move from the most negative to the most positive quantization level, numeric value of the binary symbol is incremented. This procedure is called *source coding*. In Table 5.1, fill in the 3*rd* column with 2-bit natural code. Apply this coding procedure to the quantized sequence `xq` and enter the binary codes into the 3*rd* column of Table 5.2. You may check your answer with the MATLAB function *bin_enc* (natural binary encoder):

```
>> xbin=bin_enc(xq,2)

xbin =

     1     1
     1     1
     1     0
     0     1
     1     0
     0     0
     0     1
     1     1

fx >>
```

Q5.1 Is it possible to code pulse amplitudes in sequences `x` and `xq` with 2-bit binary symbols such that each pulse amplitude remains uniquely identifiable?

ได้ ถ้า Amplitude อยู่ในช่วงของการทำ Quantization

A.5 One of the most common methods of converting an analog signal into a stream of binary digits is *pulse code modulation* (PCM). In PCM, you perform the following sequence of operations:

- sampling;
- quantization;
- source coding;
- conversion into a serial data stream.

Thus, if the source coded variable `xbin` is converted into a serial data, you in fact, generate PCM data representing the sequence `x`:

```
>> par2ser(xbin)

ans =

  Columns 1 through 8

     1     1     1     1     0     1     1     0

  Columns 9 through 16

     0     1     0     0     1     0     1     1

fx >>
```

B. Distortion Induced by Quantization

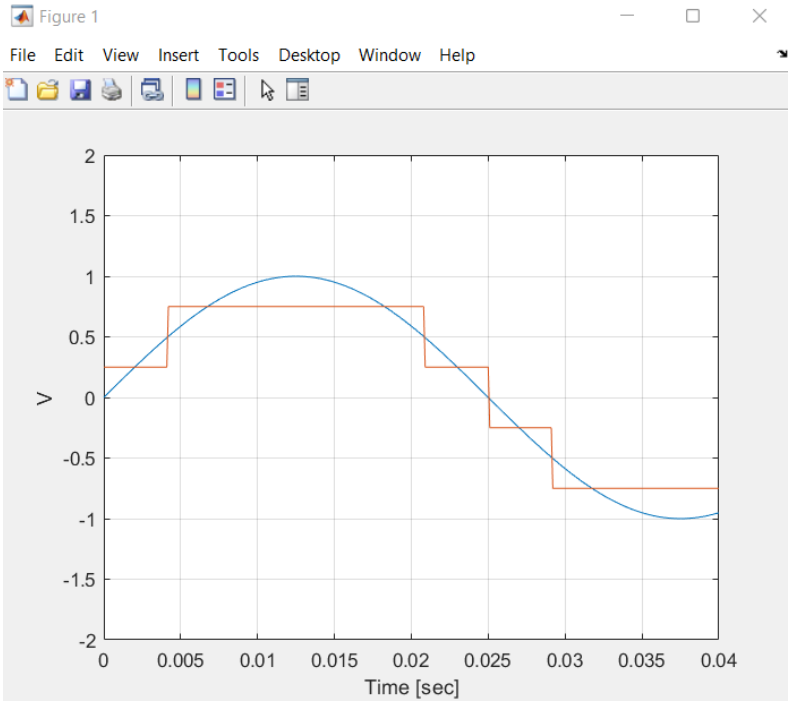
B.1 Quantization generates an approximation to the original sequence. To observe this point generate samples from a sinusoid of unit amplitude:

```
>> x=sin(2*pi*20*[1:400]/SAMPLING_FREQ);
fx >>
```

Default sampling frequency is set to 10 kHz by the variable `sampling_freq`. Use a 2-bit uniform quantizer and compare the quantized waveform with the original sequence `x`:

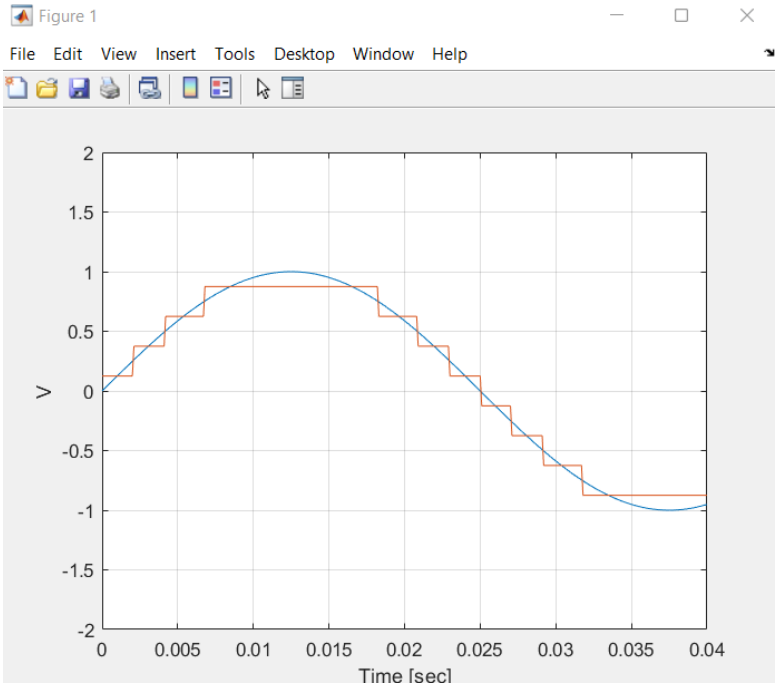
```
>> clf,waveplot(x),hold on,waveplot(quantize(x,2)
```

Wave plot



Repeat using 3-, 4- and 5-bit uniform quantizers.

3-bit



4-bit

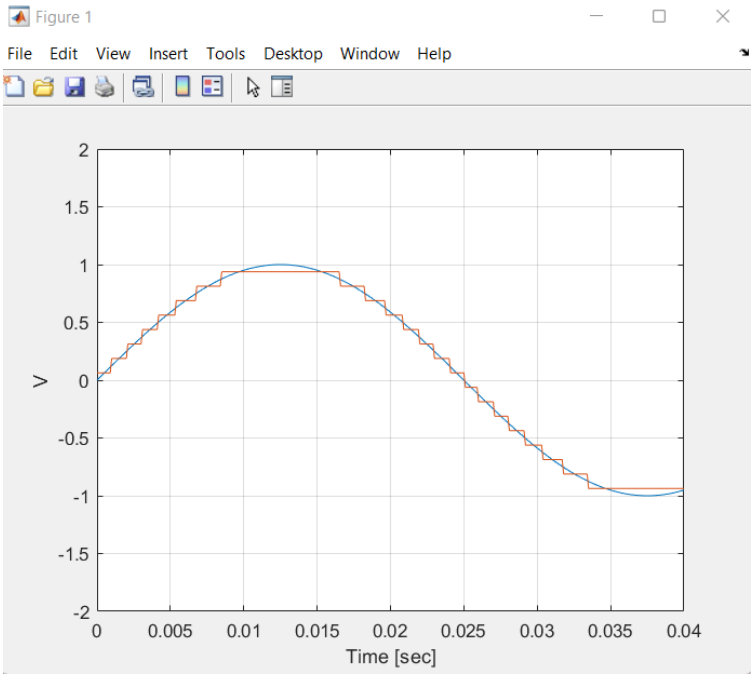


Figure 1
File Edit View Insert Tools Desktop Window Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
V
0 0.005 0.01 0.015 0.02 0.025 0.03 0.035 0.04
Time [sec]

5-bit

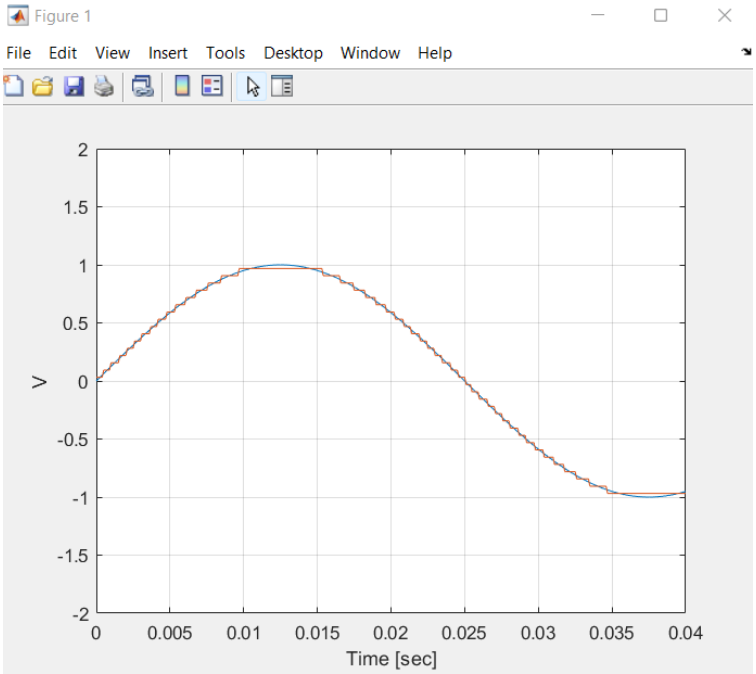


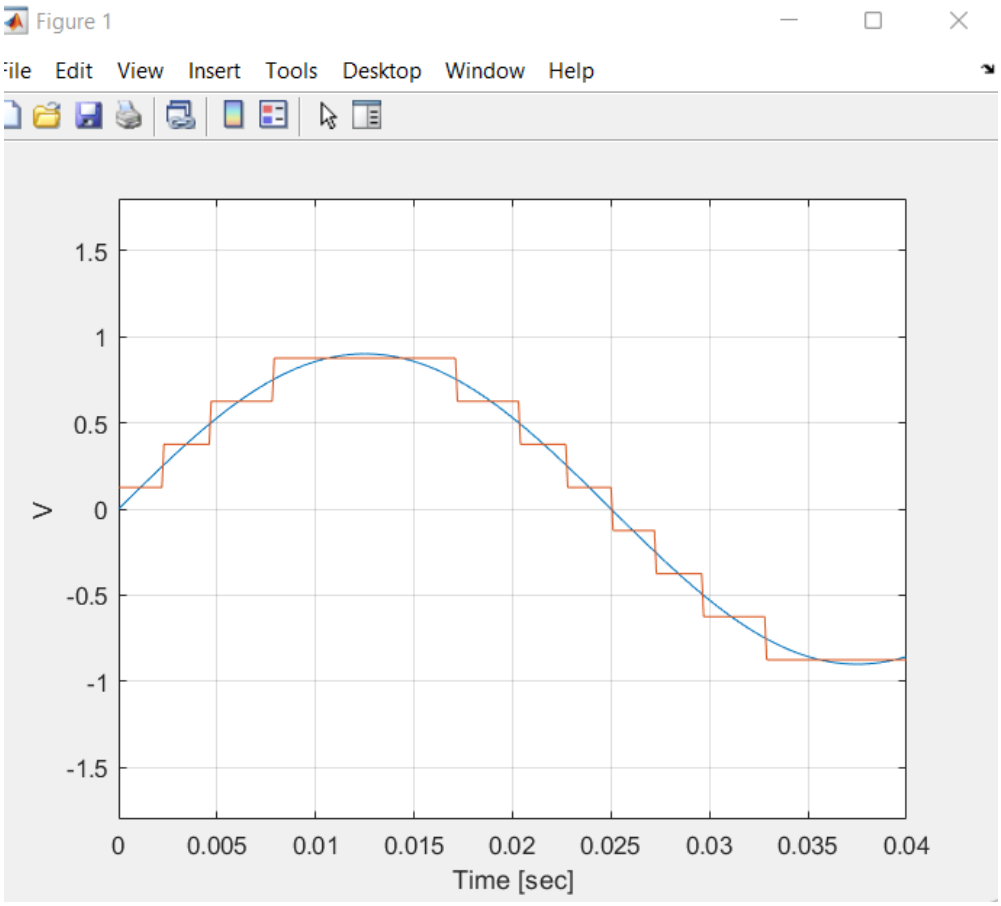
Figure 1
File Edit View Insert Tools Desktop Window Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
V
0 0.005 0.01 0.015 0.02 0.025 0.03 0.035 0.04
Time [sec]

Since the quantized sequence is only an approximation to the original, quantization produces *distortion*. There are two types of distortion associated with a quantizer: overload (or clipping) distortion and quantization noise. Overload distortion occurs when the input signal exceeds the quantizer's input range — in this experiment $[-1, 1]$. Once the quantizer range has been exceeded, quantizer output will remain at its maximum or minimum value until the input falls within the quantizer's input range.

B.2 Set the amplitude of the sinusoid **x** to 0.9 and display **x** and its 3-bit quantized version:

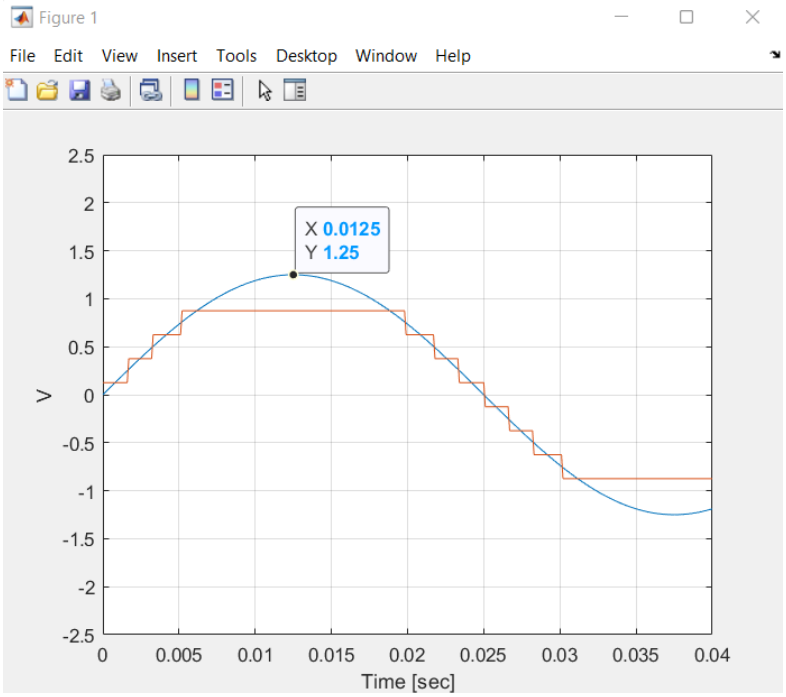
```
>> a=0.9;
>> clf
>> waveplot(a*x),hold on,waveplot(quantize(a*x,3)
```

Wave plot

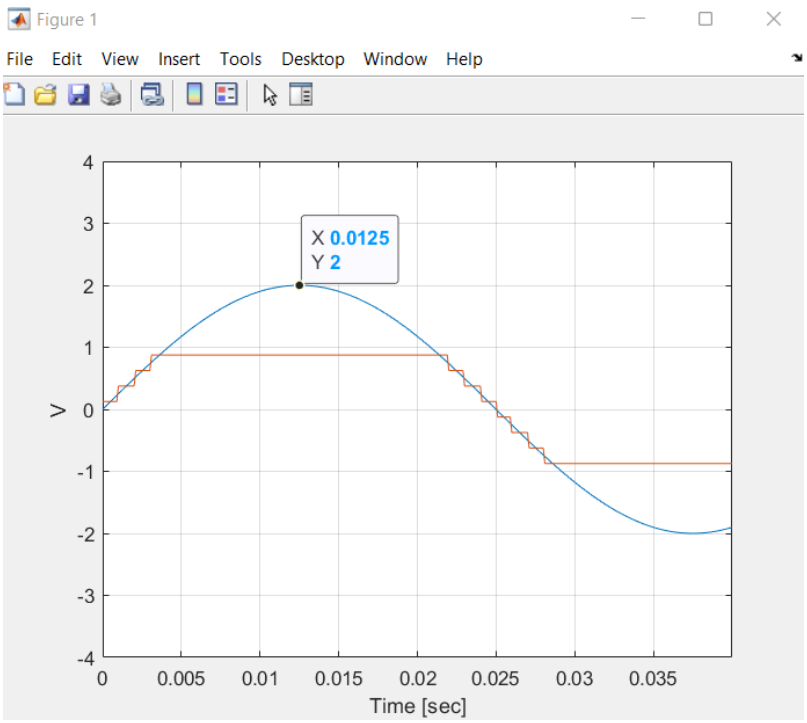


Observe that, since $\max|\mathbf{x}| = 0.9 < 1$, no clipping occurs. Now increase the signal amplitude and repeat the above step for $\mathtt{a} = 1.25$, 2 and 5.

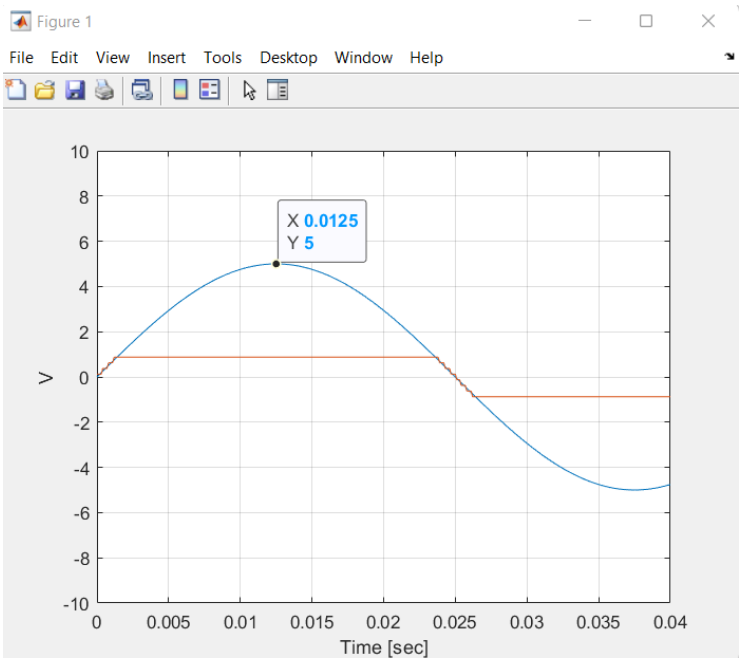
a=1.25



a=2



a=5



Q5.2 Determine the percentage of samples in `x` that are clipped by the quantizer when the amplitude is set to 0.9, 2 and 5. For the 3-bit uniform quantizer what are the minimum and maximum quantization levels?

ถ้า a = 0.9 ; 2.78% , a=2 ; 56.25% และ a=5 ; 82.5%

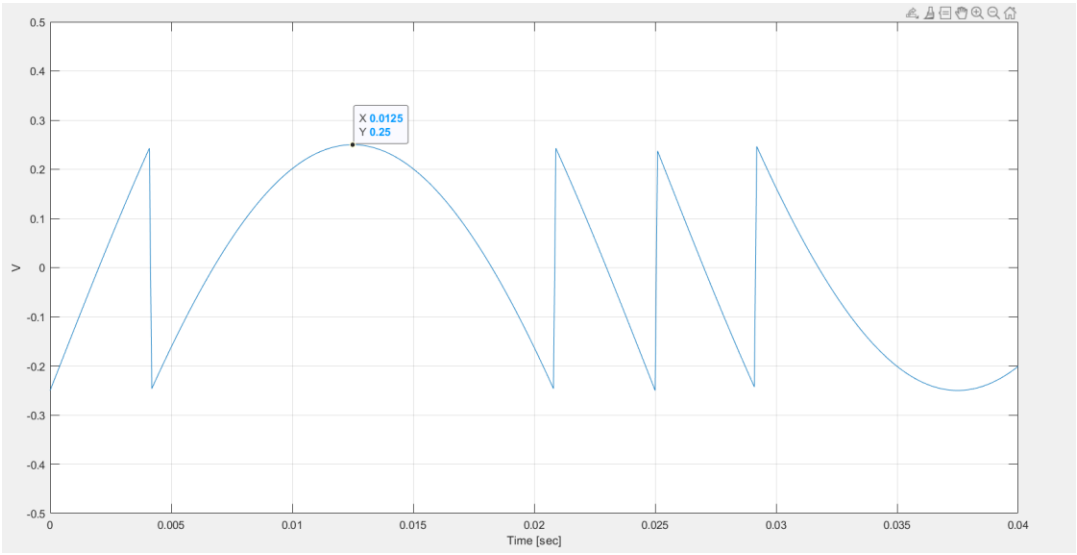
To avoid clipping a quantizer is matched to the input signal. (This feature is also available to the MATLAB function *quantize*, if it is called with the *scaling* option. For more information type `help quantize`.) The second type of distortion is due to quantization noise which arises because of the difference between the input amplitude and the quantized sample amplitude.

B.3 Use samples from the sinusoid `x` to observe quantization error `x − xq`:

where N=2

```
>> clf
>> xq=quantize(x,2);xe=x-xq;
>> waveplot(xe)
```

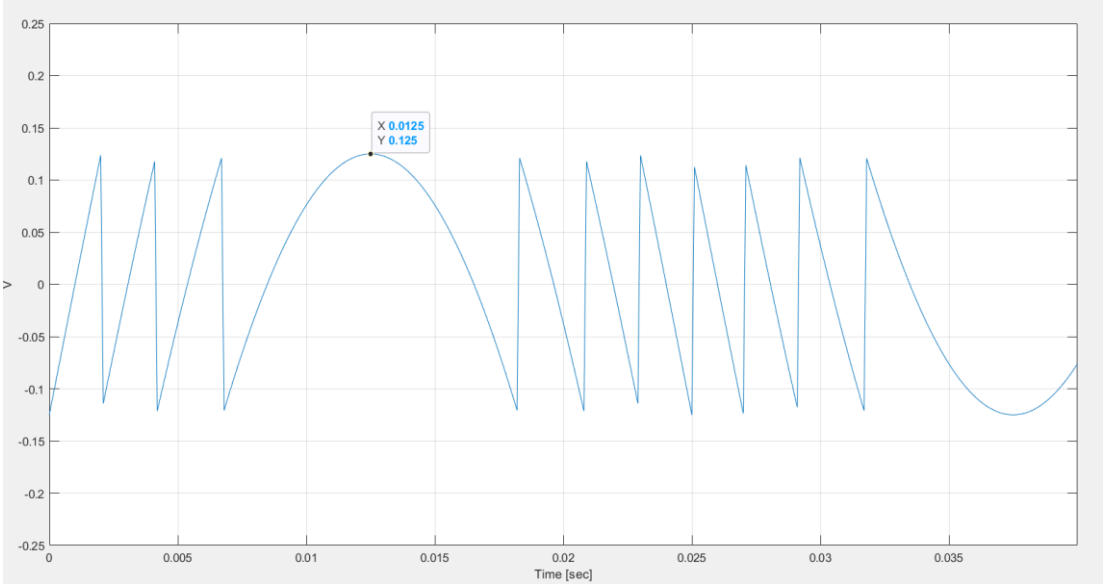
Wave plot



Where N=3

```
>> clf
>> xq=quantize(x,3);xe=x-xq;
>> waveplot(xe)
```

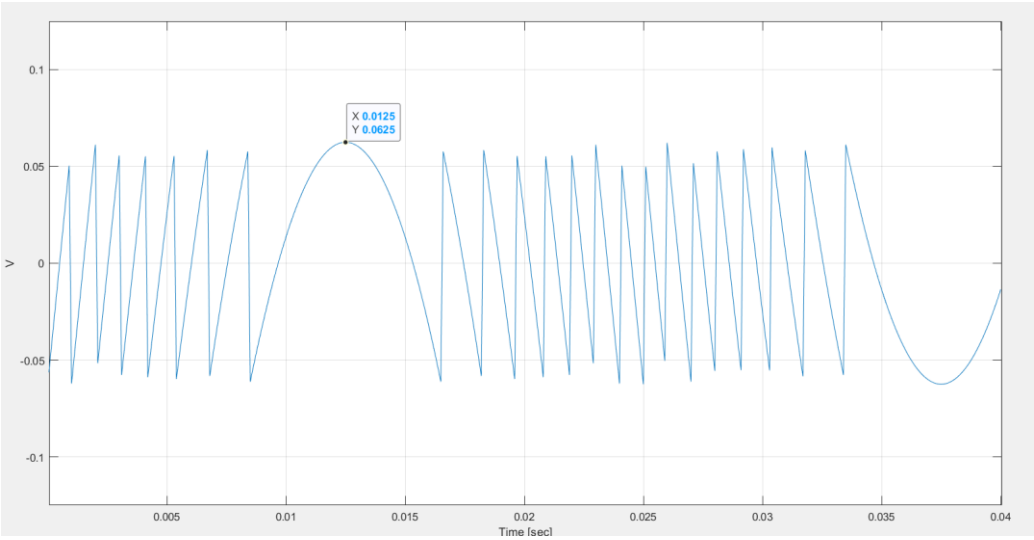
Wave plot



Where N=4

```
>> clf
>> xq=quantize(x,4);xe=x-xq;
>> waveplot(xe)
```

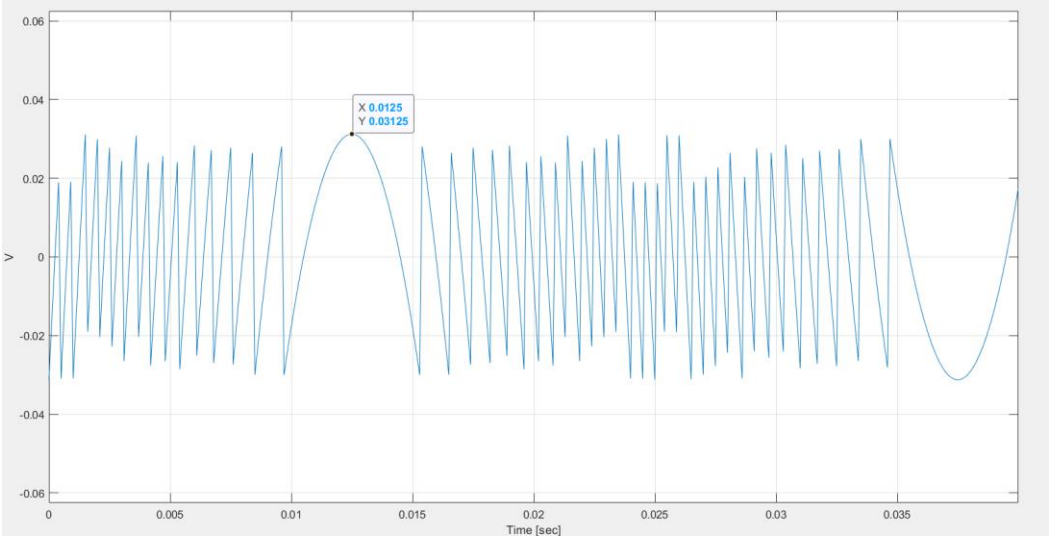
Wave plot



Where N=5

```
>> clf
>> xq=quantize(x,5);xe=x-xq;
>> waveplot(xe)
```

Wave plot



where N = 2, 3, 4 and 5. For each choice of N, record the maximum absolute value of the quantization error into Table 5.3.

N	2	3	4	5
Max |xe|	0.25	0.125	0.0625	0.03125

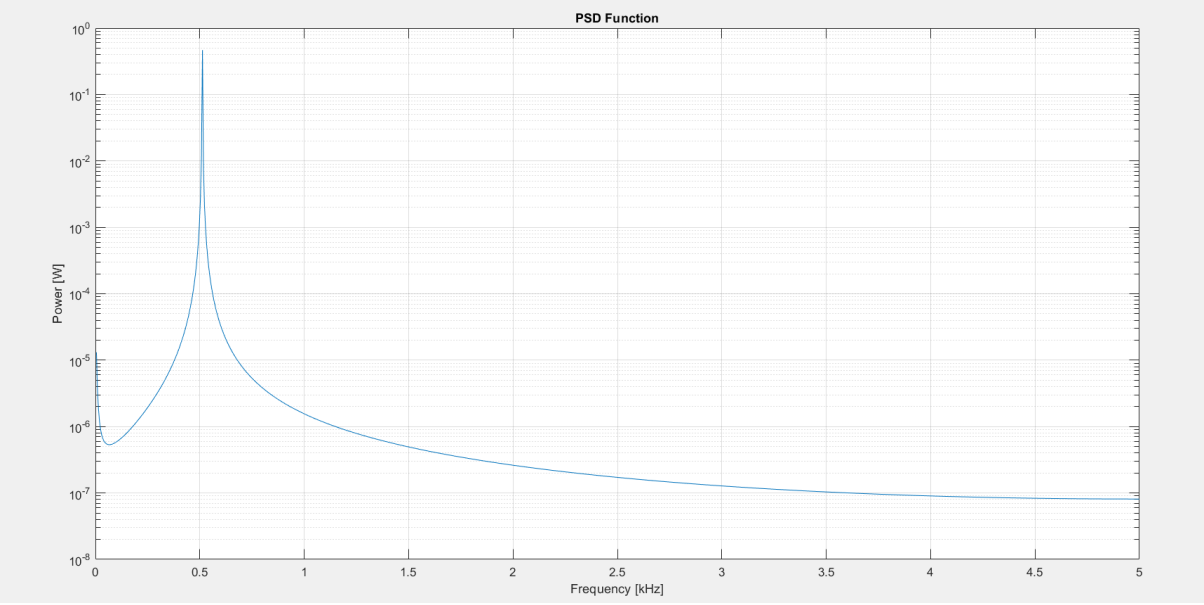
Q5.3 Derive a formula for the maximum absolute quantization error due to a N-bit uniform quantization as a function of N. Under what conditions is the quantization error reaches this maximum?

$$Max|xe| = \frac{1}{2^N}$$

B.4 Generate the following sequence and display its PSD function:

```
>> x=sin(2*pi*512*[1:2048]/SAMPLING_FREQ);
>> psd(x)
>>
```

Wave plot

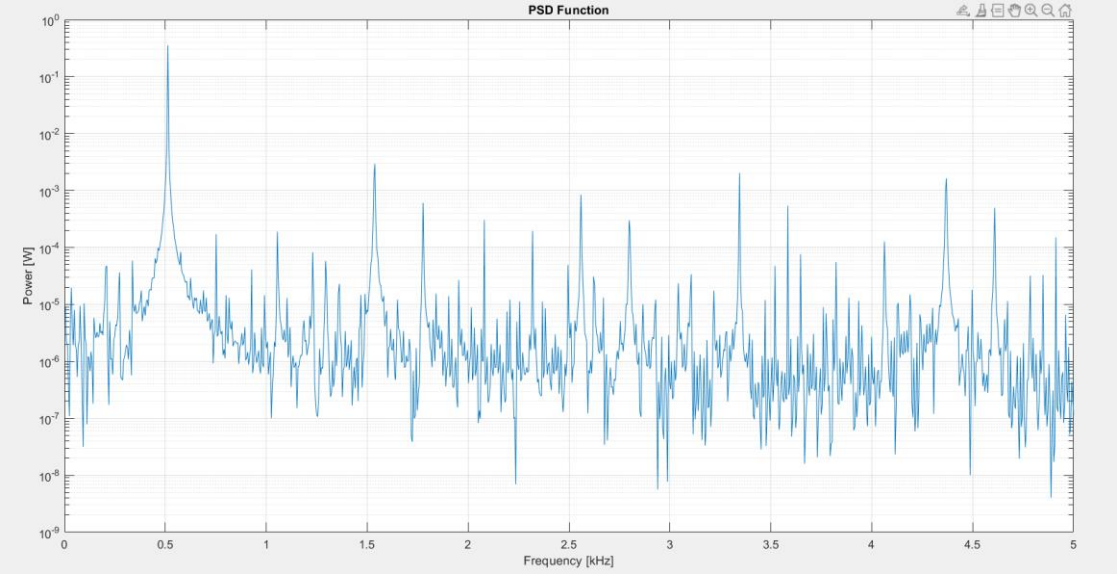


For N = 2, 3, 4, 5, and 6 display the PSD function of the quantized waveform xq and evaluate the corresponding quantization error power in dB. Enter results into Table 5.4.

Where N=2

```
>> xq=quantize(x,2);
>> psd(xq)
>> sq2=10*log10(var(x-xq))

sq2 =

  -15.8187
```

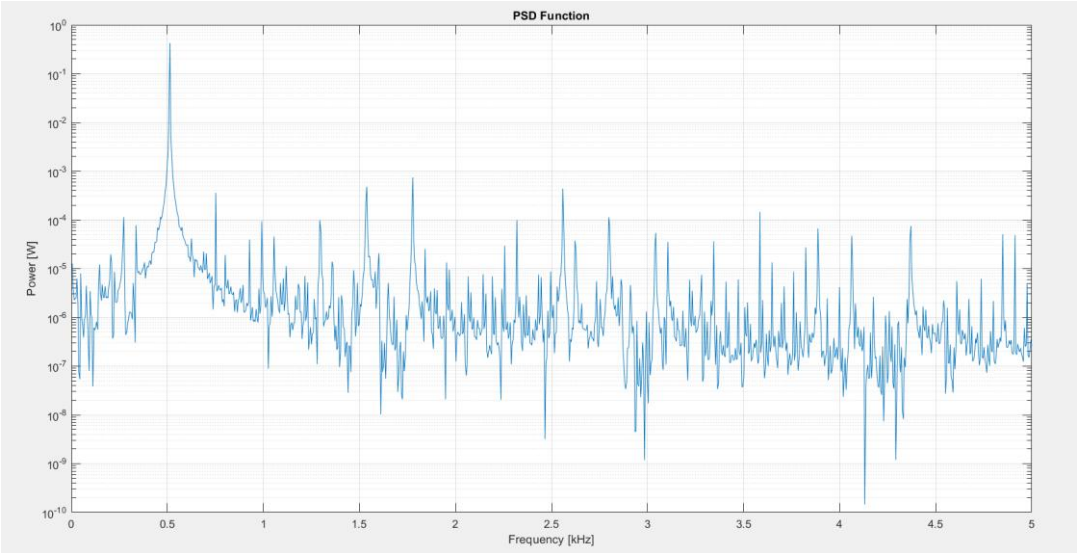
Wave plot



Where N=3

```
>> xq=quantize(x,3);
>> psd(xq)
>> sq2=10*log10(var(x-xq))

sq2 =

  -22.1046

fx >>
```

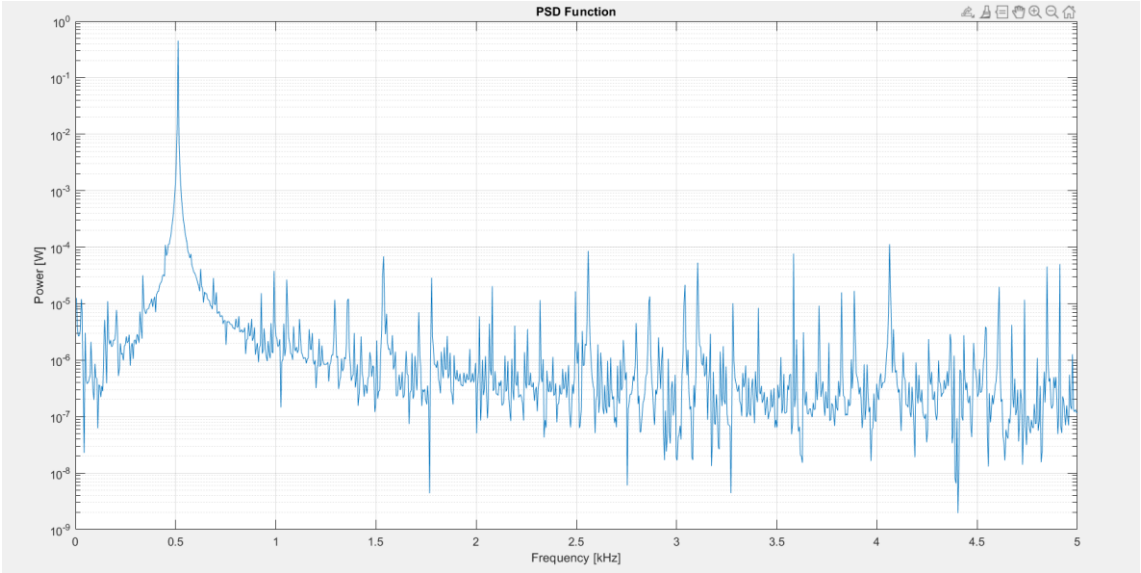
Wave plot



Where N=4

```
>> xq=quantize(x,4);
>> psd(xq)
>> sq2=10*log10(var(x-xq))

sq2 =

  -28.3320
```

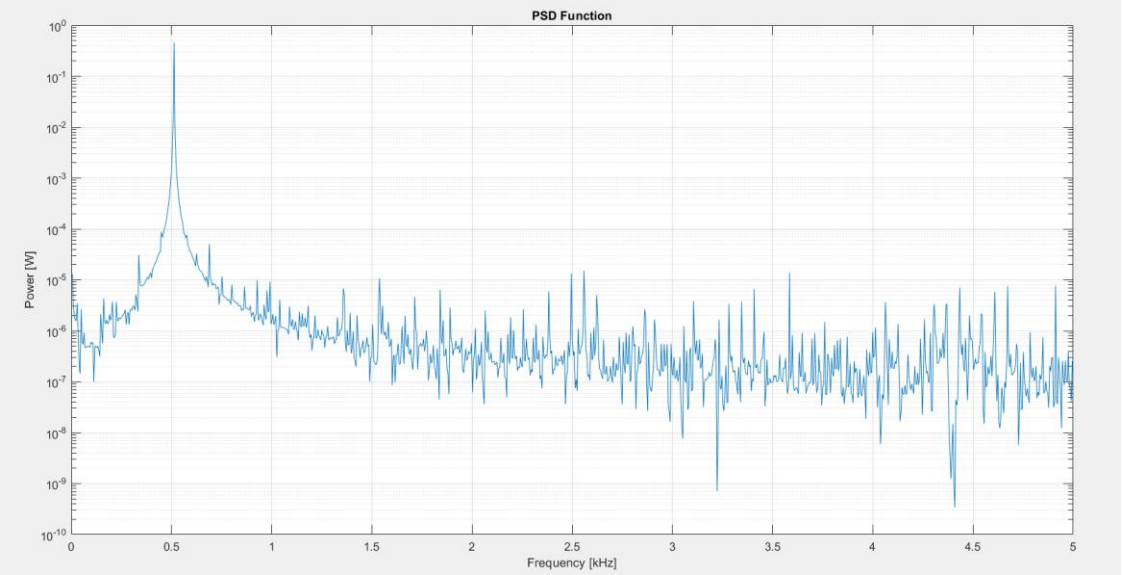
Wave plot



Where N=5

```
>> xq=quantize(x,5);
>> psd(xq)
>> sq2=10*log10(var(x-xq))

sq2 =

  -34.4960

fx >>
```

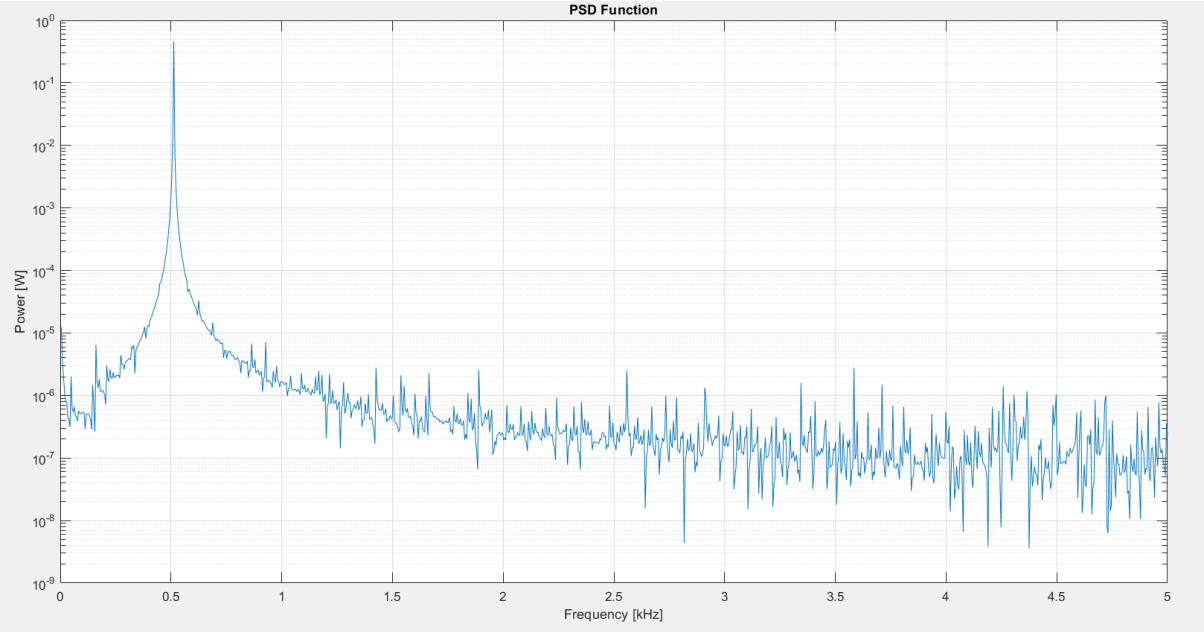
Wave plot



Where N=6

```
>> xq=quantize(x,6);
>> psd(xq)
>> sq2=10*log10(var(x-xq))

sq2 =

  -40.6210

fx >>
```

Wave plot



N	2	3	4	5	6
${\sigma_q}^2$	-15.8187	-22.1046	-28.3320	-34.4960	-40.6210

Under the assumption that the quantization error amplitude is uniformly distributed over $[-q/2,\ q/2]$ where q is the quantization step size of an N-bit uniform quantizer, noise power is given by:

$$\sigma_q^2 = -(4.77 + 6.02N) \quad \text{dB}.$$

Compare results in Table 5.4 with those obtained from the formula.

${\sigma_q}^2 = -(4.77 + 6.02x2) = -16.81\ dB$

${\sigma_q}^2 = -(4.77 + 6.02x3) = -22.83\ dB$

${\sigma_q}^2 = -(4.77 + 6.02x4) = -28.85\ dB$

${\sigma_q}^2 = -(4.77 + 6.02x5) = -34.87\ dB$

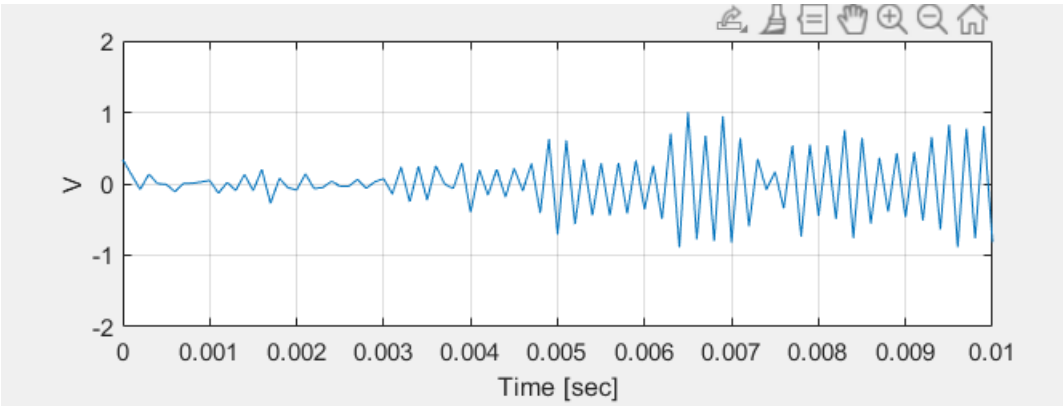
${\sigma_q}^2 = -(4.77 + 6.02x3) = -40.89\ dB$

C . Non-uniform Quantization

C.1 Generate 100 samples representing voiced segment of a typical speech signal and display the sequence:

```
>> s=speech(100);
>> subplot(211),waveplot(s)
```

Wave plot

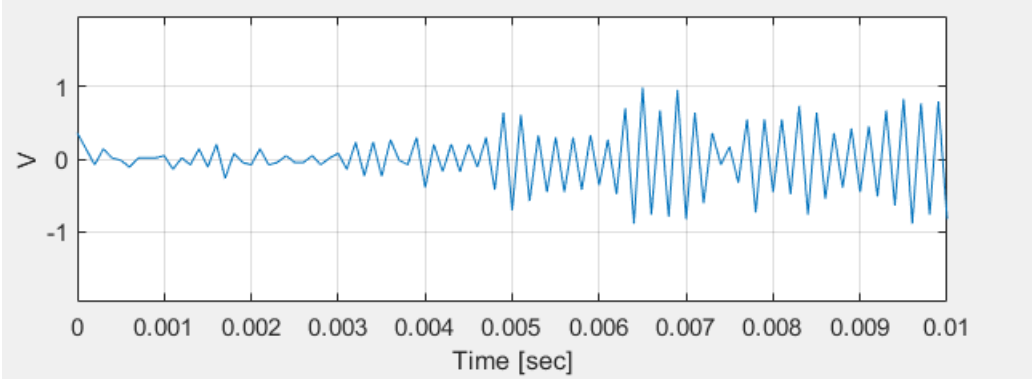


Observe that the signal amplitude is restricted to $[-1, 1]$ and there are samples in `s` with amplitude ± 1. Thus, `s` spans the entire input range of the quantizer.

C.2 Quantize `s` using a 6-bit uniform quantizer and display the quantized waveform:

```
>> sq=quantize(s,6);
>> subplot(212),waveplot(sq)
```

Wave plot



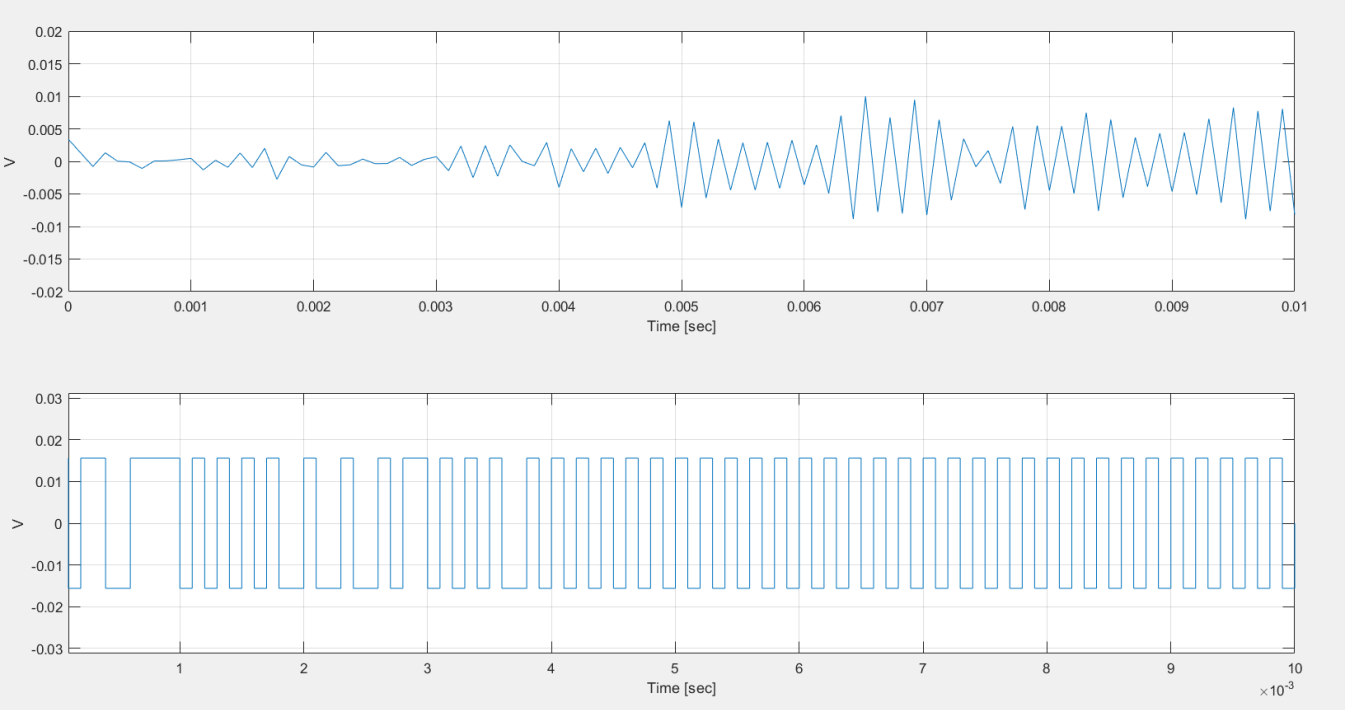
The quantized waveform `sq` closely resembles the original sequence `s`. Now, scale `s` by a factor `K` such that $\max|K\,s| \leq 0.01$.

```
>> x=normalize(s,0.01);
>>
```

As you display `x`, you will observe that `x` is an attenuated version of `s`. Quantize `x` using the same 6-bit uniform quantizer and display the quantized signal `xq`:

```
>> xq=quantize(x,6);
>> subplot(211),waveplot(x),subplot(212),waveplot(xq)
```

Wave plot



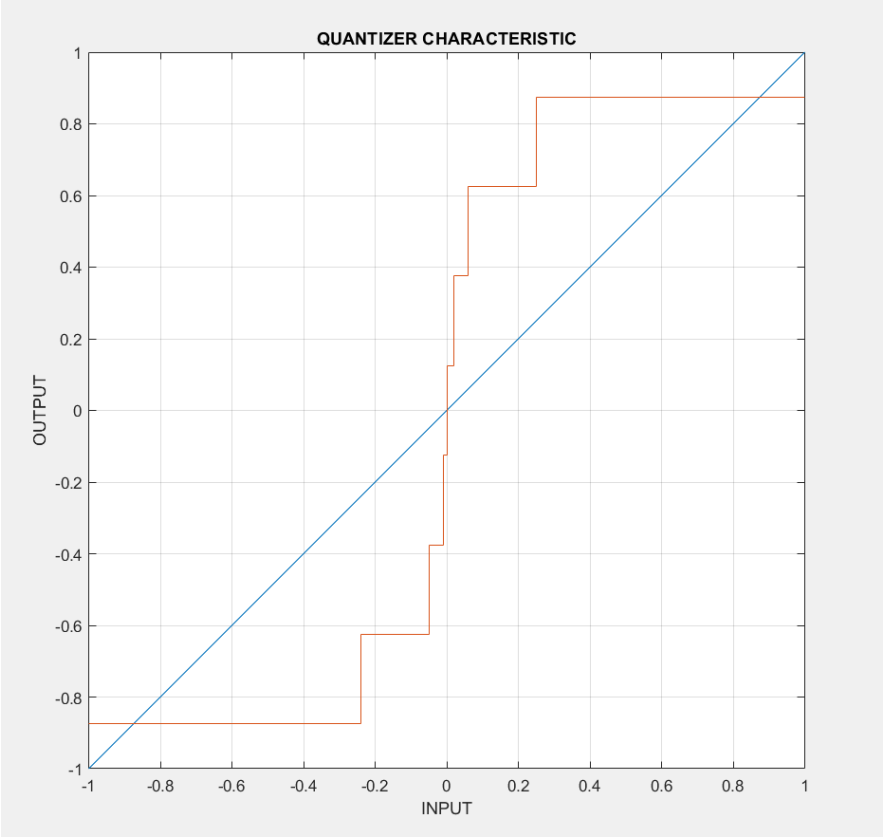
Can you explain why `xq` differs so much from `s` or `sq`?

ค่า x ที่นำมาทำ Quantize เพื่อให้ได้ค่า xq ถูก normalize ด้วยค่า k =0.01 นั่นเอง

C.3 Non-uniform quantization: Display the quantizer characteristic of a μ-law companding 3-bit quantizer:

```
>> clf,quant_ch(3,'mu_law')
>>
```

Wave plot



Q5.4 Consider a sampled signal with pulse amplitudes restricted to the interval $[-0.25, 0.25]$. If the signal is quantized, how many quantization levels are *actually* used by a μ-law companding 3-bit quantizer? By a uniform 3-bit quantizer?

ในช่วง [-0.25,0.25] การทำ Uniform มี 3 Level แต่การทำ mu_law จะมี 8 Level

C.4 Apply μ-law companding to sampled signals **s** and **x** and then quantize:

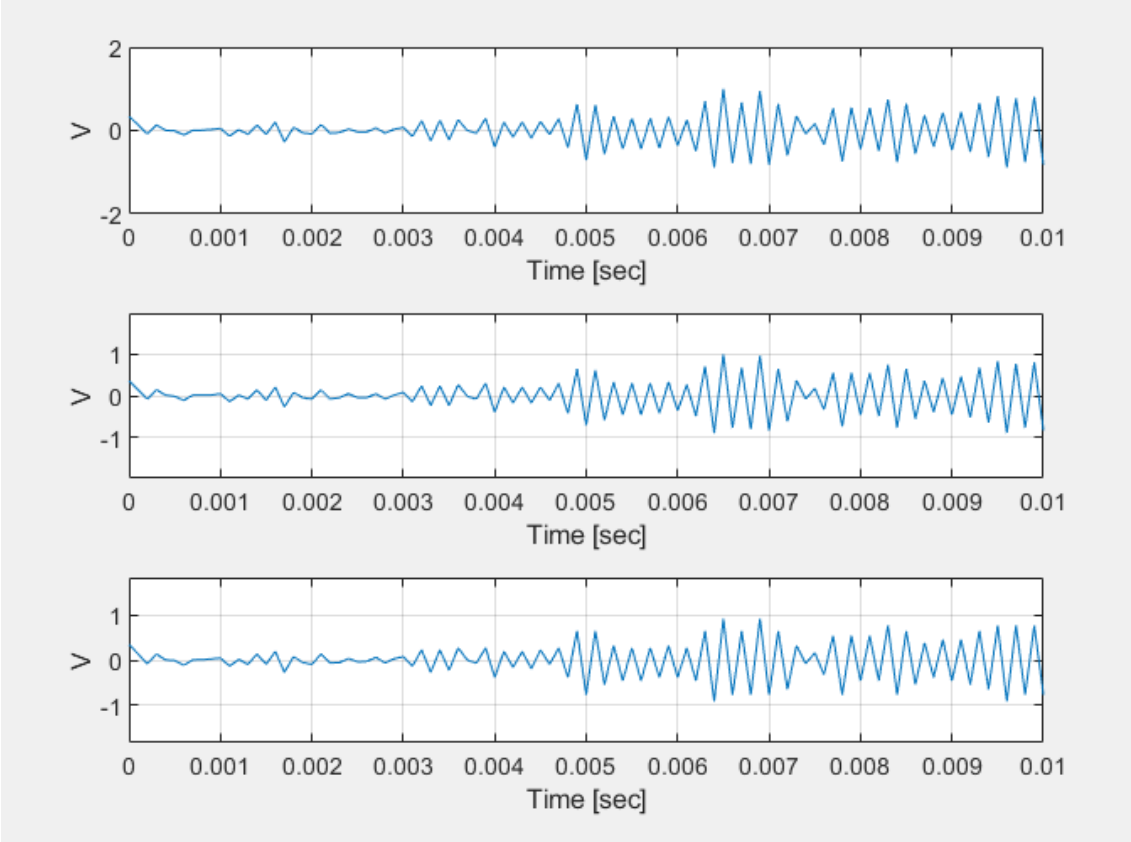
```
>> msq=mu_inv(quantize(mu_law(s),6));
>> mxq=mu_inv(quantize(mu_law(x),6));
>>
```

Since quantized samples must be restored to their original values, non-uniform quantization is equivalent[1] to the following sequence of operations: *compand* — *uniform quantization* — *expand*. Display waveforms

`msq`, `mxq` and compare with original signals `s` and `x`. It is of particular interest to compare `msq` with `sq` and `mxq` with `xq`. First, the large amplitude signal `s`:

```
≫ clf, subplot(311), waveplot(s)
≫ subplot(312), waveplot(sq)
≫ subplot(313), waveplot(msq)
```

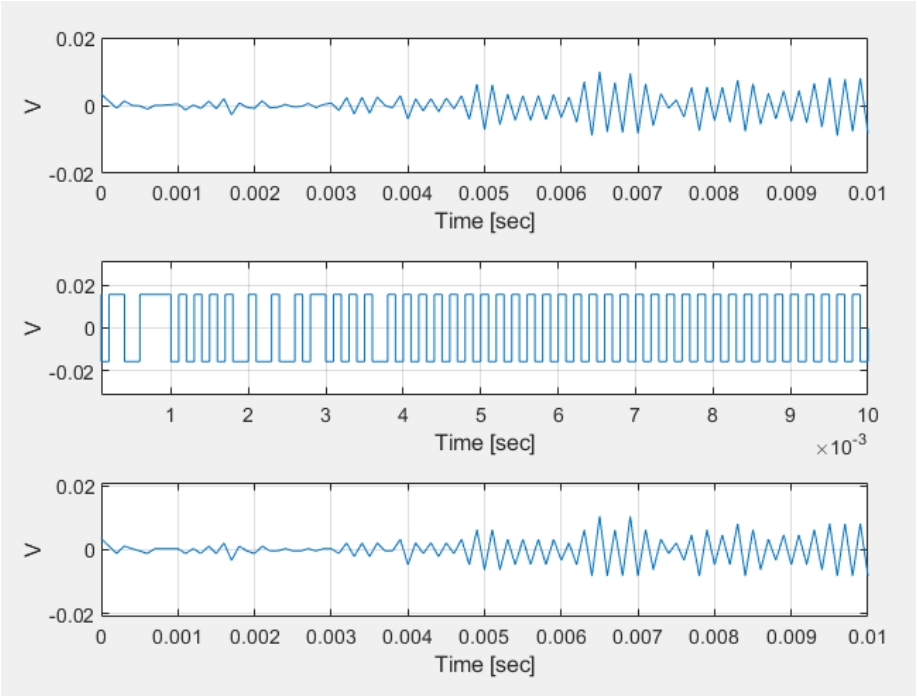
Wave plot



Now for the small amplitude signal `x`:

```
≫ figure(2), subplot(311), waveplot(x)
≫ subplot(312), waveplot(xq)
≫ subplot(313), waveplot(mxq)
```

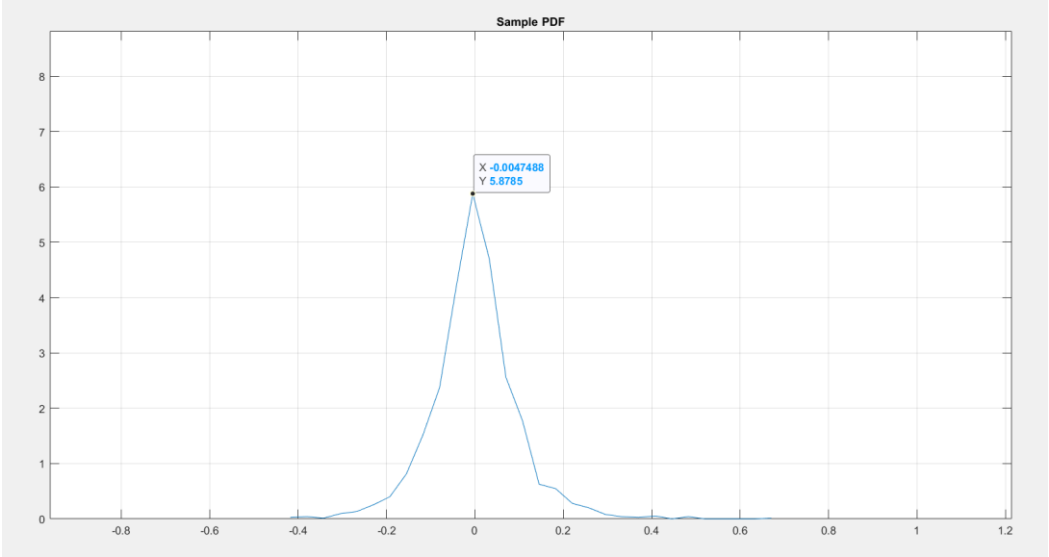
Wave plot



C.5 For a given number of quantization levels, a uniform quantizer is optimum if input sample amplitudes are uniformly distributed over the input range $[-1, 1]$. However, speech signal amplitude distribution is best modelled by a Laplace distribution. Generate the sample pdf of 2,000 samples from a Laplace distribution with variance 0.01:

```
>> close(2),figure(1),clf
>> a=laplace(2000,0.01);
>> pdf(a,30)
```

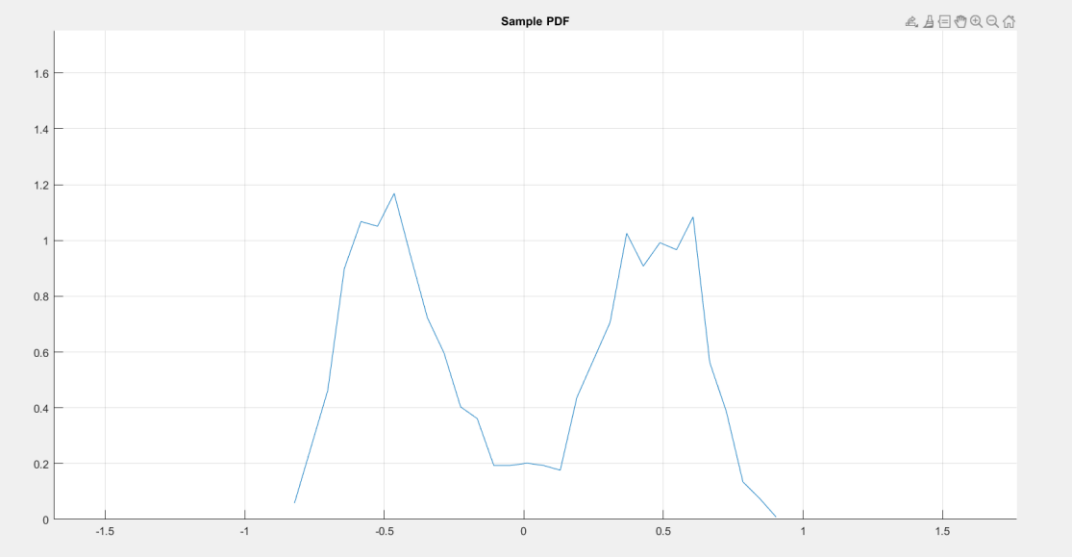
Wave plot



Compand the sequence a by a μ-law compander and display the sample pdf of the resulting sequence:

```
>> b=mu_law(a);
>> hold on , pdf(b,30)
```

Wave plot



The pdf of the companded sequence b resembles a uniform pdf, which results in better utilization of available quantization levels.

C.6 In practise, any quantizer has to operate on signals at varying amplitude levels. Therefore, to assess the quantizer performance, it must be tested with signals at different power levels. One measure of quantizer performance is the signal to quantization noise ratio SQNR. Consider a 8-bit quantizer used in a typical voice grade A/D converter:

Algorithm:

1. Generate a sequence of 1,000 voice signal samples at a specified power level and measure the signal power: σ_s^2.
2. Quantize the sequence using an 8-bit uniform quantizer. Measure the resulting quantization error power: $\sigma_q^2(\text{uniform})$.
3. Quantize the sequence using a μ-law companding 8-bit quantizer. Measure the resulting quantization error power: $\sigma_q^2(\mu\text{-law})$.
4. Measure the $SQNR = \sigma_s^2/\sigma_q^2$ for both quantizer types.

This algorithm is available in MATLAB function *exp5_c6*

Where power = 0.0001

```
>> exp5_c6(0.0001)

  SIGMAs^2    SQNR(un)   SQNR(mu-law)
-------------------------------------

ans =

  -39.5046    13.3967    35.8250

fx >>
```

Where power = 0.001

```
>> exp5_c6(0.001)

  SIGMAs^2    SQNR(un)   SQNR(mu-law)
-------------------------------------

ans =

  -29.9521    22.9103    37.1779

fx >>
```

Where power = 0.01

```
>> exp5_c6(0.01)

  SIGMAs^2    SQNR(un)   SQNR(mu-law)
-------------------------------------

ans =

  -20.0733    32.8674    37.3474

fx >>
```

Where power = 0.1

```
>> exp5_c6(0.1)

  SIGMAs^2    SQNR(un)   SQNR(mu-law)
-------------------------------------

ans =

  -10.1082    42.8029    36.5238

fx >>
```

Where power = 1

```
>> exp5_c6(1)

  SIGMAs^2    SQNR(un)   SQNR(mu-law)
-------------------------------------

ans =

   -3.9651    47.3530    34.7150

fx >>
```

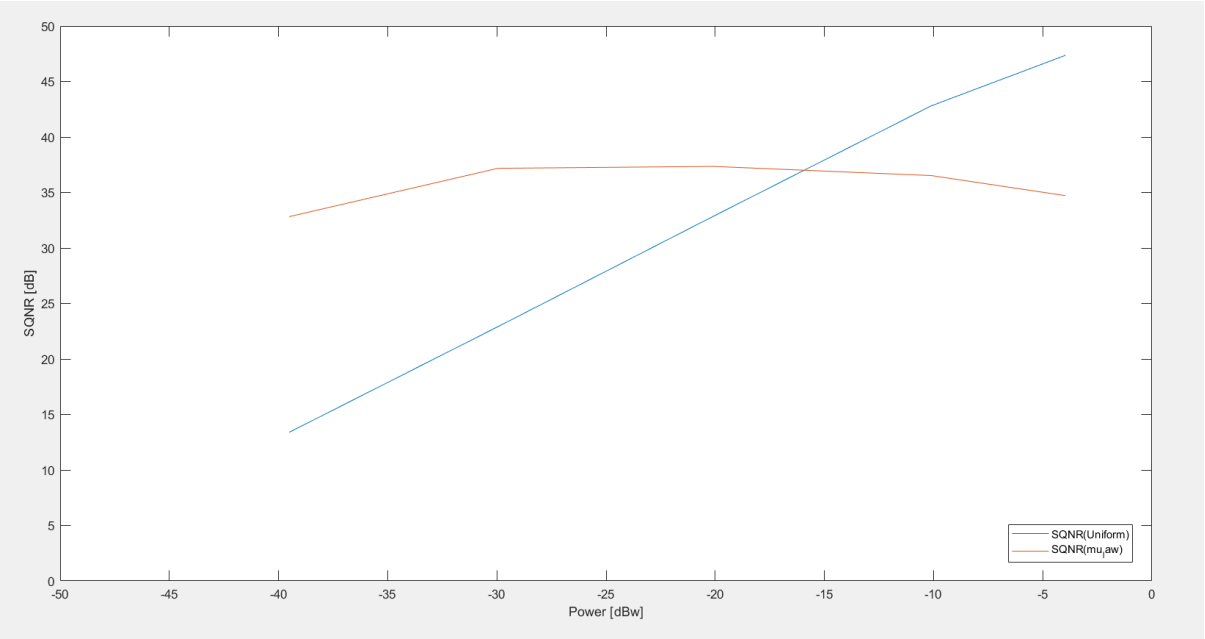
where power = [.0001, .001, .01, .1, 1]. The output is in the form:

$$[\ \sigma_s^2, \quad \text{SQNR(uniform)}, \quad \text{SQNR}(\mu\text{-law})\ ],$$

where all quantities are measured in dB. Enter results into Table 5.5. Also plot SQNR vs. σ_s^2 for both quantizer types in Graph 5.2.

$\sigma_s{}^2$	SQNR (Uniform)	SQNR (μ-law)
-39.5046	13.3967	32.8250
-29.9521	22.9103	37.1779
-20.0733	32.8674	37.3474
-10.1082	42.8029	36.5238
-3.9651	47.3530	34.7150

Plot



Q5.5 Based on the results depicted in Graph 5.2 comment on the preferred quantizer type for input signals if:

a. $\sigma_s^2 \in [-40, -10]$ dBW.

b. $\sigma_s^2 \in [-10, 0]$ dBW.

c. $\sigma_s^2 \in [-40, 0]$ dBW.

a. : mu_law , b : Uniform และ c : mu_law