EAST WEST UNIVERSITY

Department of EEE

***SECTION*: 01**
***COURSE CODE*: EEE307**
***COURSE NAME*: TELECOMUNICATION EENGINEERING**

***COURSE INSTRUCTION'S NAME*: KAMANASHIS SAHA,**
LECTURER, EEE, EWU

PROJECT

***SUBMISSION DATE*: 05 SEPTEMBER, 2023**

SUBMITTED BY (GROUP-04)

- ***KAZI IFTIER RAHMAN (2020-1-80-004)***
- ***ARPON PODDER (2020-1-80-005)***
- ***ROHIT BHOWMICK(2020-1-80-006)***
- ***FARHANA MISU(2020-1-80-027)***

OBJECTIVE

In this lab project, we need to create a staircase approximation of a sinusoidal monotone message signal by applying delta modulation. Delta modulation is the signal conversion technique from analog to digital and digital to analog. This modulation transmits only one bit for one sample. To minimize the total (Granular noise + Slope overload noise) noise it's important to properly select step size and Δ's value based on the sampling frequency and the maximum slope of the message signal.

THOERY

***DELTA MODULATION (DM)*:** Delta modulation is a simple analog-to-digital modulation technique used in signal processing and telecommunications. It is a type of pulse code modulation (PCM) where the analog signal is approximated by a sequence of discrete values, typically binary values (0 or 1), to represent the variations in the analog signal.

***ADAPTIVE DELTA MODULATION (ADM)*:** Adaptive Delta Modulation (ADM) is a variation of the Delta Modulation (DM) technique used in digital signal processing and telecommunications for analog-to-digital conversion. ADM is designed to improve the efficiency and accuracy of delta modulation by dynamically adjusting the step size (delta) based on the characteristics of the input signal.

- ***STEP-01***

  Group No: 04
  Group Members SID last digit: 4, 5, 6, 7
  $X = 4$
  $f_m = 1X = 14kHz = 14000Hz$;
  $Y = round(\frac{4+5+6+7}{4}) = 6$;
  $A_m = 1Y = 16\,V$;

Now,
Message signal,$m(t) = A_m \sin(2\pi f_m t) = 16\sin(2\pi \times 14000t)$

- ***MATLAB CODE***

```
clc
close all
clear

%Task-01

X = 4;
Y = round((4+5+6+7)/4);

Am = 16;  %V (According to manual amiplitude 1Y V)
fm = 14000;  %Hz (According to manual amiplitude 1X kHz)
t=0:1/(fm*100):2/fm; %sec
```

```
mt = Am*sin(2*pi*fm*t); %Message signal

figure(1)
plot(t,mt,'LineWidth',2);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Message Signal');
axistight
```

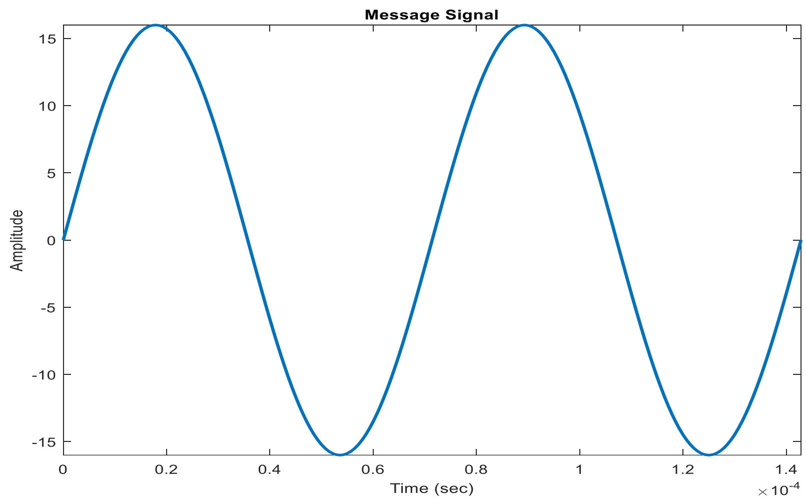

Figure 01: **Message Signal**

❖ ***STEP-02***

The Nyquist frequency is half of the sampling frequency and represents the maximum frequency that can be accurately represented in a sampled signal. In this case, the message signal has a frequency of 14,000 Hz. Therefore, the Nyquist frequency can be calculated as:

Nyquist frequency, f_{NF}= (14,000 Hz) / 2 = 7,000 Hz

The minimum sampling frequency should be at least twice the Nyquist frequency to avoid aliasing and accurately reconstruct the original signal. So, for this message signal, the minimum sampling frequency should be: Minimum sampling frequency, $f_{s_{min}}$ = 2 * Nyquist frequency = 2 * 7,000 Hz = 14,000 Hz

Choose a sampling frequency 10 times the Nyquist frequency and sample the message signal with it.
So, the Minimum sampling frequency, f_s = 10* Nyquist frequency = 10 * 7,000 Hz = 70,000 Hz

✓ ***MATLAB CODE***

```
clc
close all
clear

X = 4;
Y = round((4+5+6+7)/4);

Am = 16;  %V (According to manual amiplitude 1Y V)
fm = 14000;  %Hz (According to manual amiplitude 1X kHz)
t=0:1/(fm*100):2/fm; %sec

mt = Am*sin(2*pi*fm*t); %Message Signal
```

```
f_NF = fm/2; %Hz (Nyquist Frequency)
fs_min = 2*fm;%Hz (Minimum sampling frequency)
fs = 10*f_NF;
Ts = 1/fs;
ts = 0:1/fs:max(t);

ms = Am*sin(2*pi*fm*ts); %Sampled signal

figure(2)
plot(t,mt,'b'); hold on
stem(ts,ms,'r','LineWidth',2);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Message Signal and Sampled signal');
legend('Message signal','Sampled signal');
axistight
```

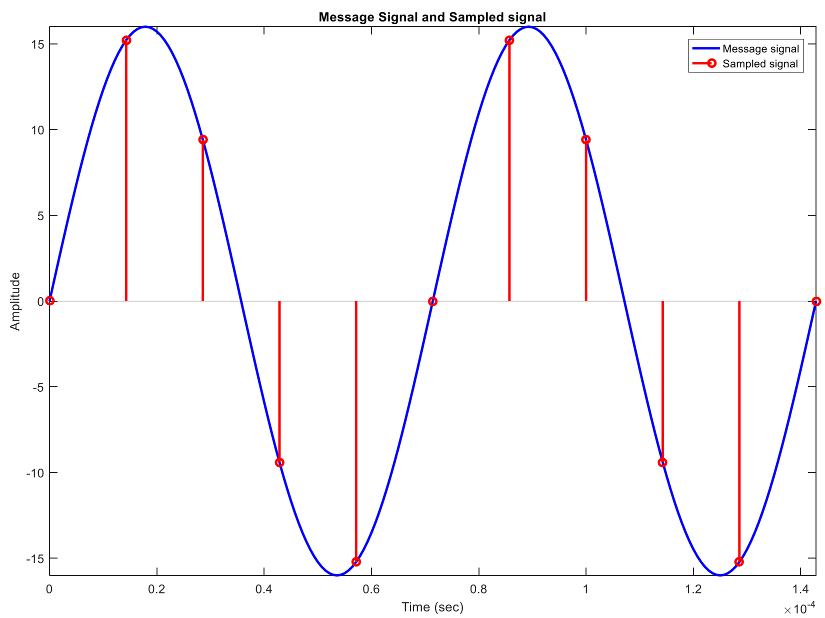

Figure 02: **Message Signal and Sampled Signal**

❖ ***STEP-03***

Determine the value of the step size, δ for which there will be no slope overload noise.

Answer: $\delta = \frac{2\pi f_m A_m}{f_s} = \frac{2\times\pi\times14000\times16}{70,000} = 20.10619$ is step size for no slope overload.

✓ ***MATLAB CODE***

```
clc
close all
clear
```

```
X = 4;
Y = round((4+5+6+7)/4);

Am = 16;  %V (According to manual amiplitude 1Y V)
fm = 14000;  %Hz (According to manual amiplitude 1X kHz)
t=0:1/(fm*100):2/fm; %sec

mt = Am*sin(2*pi*fm*t); %Message Signal

f_NF = fm/2; %Hz (Nyquist Frequency)
fs_min = 2*fm;%Hz (Minimum sampling frequency)
fs = 10*f_NF;

delta = (2*pi*fm*Am)/fs; %step size for which there will be no slope overload noise.
```

- ***STEP-04***

- ***MATLAB CODE***

```
clc
close all
clear

X = 4;
Y = round((4+5+6+7)/4);

Am = 16;  %V (According to manual amiplitude 1Y V)
fm = 14000;  %Hz (According to manual amiplitude 1X kHz)
t=0:1/(fm*100):2/fm; %sec

f_NF = fm/2; %Hz (Nyquist Frequency)
fs_min = 2*fm;%Hz (Minimum sampling frequency)
fs = 10*f_NF;
Ts = 1/fs;
ts = 0:1/fs:max(t);

ms = Am*sin(2*pi*fm*ts);%Sampled signal

delta = (2*pi*fm*Am)/fs; %step size, for which there will be no slope overload noise.

mp = 0;
for i = 1:length(ms)
ifms(i)>= mp
md(i) = mp + delta;
mb(i) = dec2bin(1);
else
md(i) = mp-delta;
mb(i) = dec2bin(0);
end
mp = md(i);
```

```
end

figure(3)
stairs(ts,md,'LineWidth',2);
xlabel('Time(sec)');
ylabel('Amplitude');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta)]);
axis tight
```

COMMAND WINDOW:

```
Command Window
>> mb

mb =

10100110010

fx >>
```

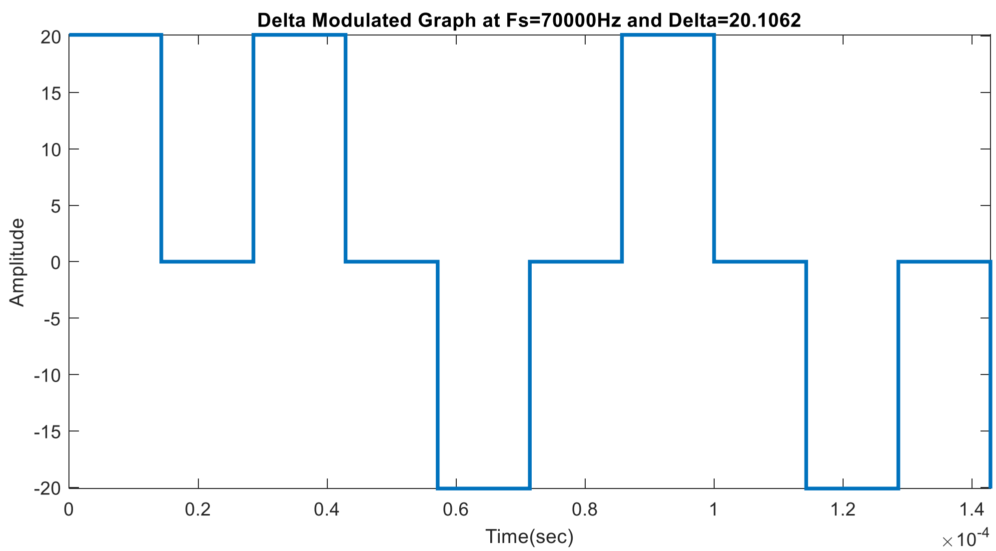

Figure 03: Delta Modulated graph when Fs=70000Hz and delta=20.1062

❖ ***STEP-05***

Max Slope=step size*sampling frequency=$(20.10619 \times 70000) = 1407433.3$
Slope overload noise= Max Slope$-$step size=1407413.194

Granular noise $= \sqrt{\delta/_{12}} = 1.2944$

❖ ***STEP-06***

Change the step size to 0.25δ, 0.5δ, δ, 1.5 δ, 2 δ

✓ ***MATLAB CODE***

```
clc
close all
clear

X = 4;
Y = round((4+5+6+7)/4);

Am = 16;  %V (According to manual amiplitude 1Y V)
fm = 14000;  %Hz (According to manual amiplitude 1X kHz)
t=0:1/(fm*100):2/fm; %sec

mt = Am*sin(2*pi*fm*t); %Message Signal

f_NF = fm/2; %Hz (Nyquist Frequency)
fs_min = 2*fm;%Hz (Minimum sampling frequency)
fs = 10*f_NF;
Ts = 1/fs;
ts = 0:1/fs:max(t);

ms = Am*sin(2*pi*fm*ts);%Sampled signal

delta = (2*pi*fm*Am)/fs; %step size, for which there will be no slope overload noise.

mp = 0;
for i = 1:length(ms)
ifms(i)>= mp
md(i) = mp + delta;
mb(i) = dec2bin(1);
else
md(i) = mp-delta;
mb(i) = dec2bin(0);
end
mp = md(i);
end

%For 0.25*delta
delta_1 = 0.25*delta;

mp_1 = 0;
```

```
for i = 1:length(ms)
ifms(i)>= mp_1
 md_1(i) = mp_1 + delta_1;
  mb_1(i) = dec2bin(1);
else
 md_1(i) = mp_1-delta_1;
 mb_1(i) = dec2bin(0);
end
 mp_1 = md_1(i);
end

figure(4)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_1,'LineWidth',2);
xlabel('Time(sec)');
ylabel('Amplitude');
axistight
legend('Message Signal','SampledSignal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_1)]);

%For 0.5*delta
delta_2 = 0.5*delta;

mp_2 = 0;
for i = 1:length(ms)
ifms(i)>= mp_2
 md_2(i) = mp_2 + delta_2;
  mb_2(i) = dec2bin(1);
else
 md_2(i) = mp_2-delta_2;
 mb_2(i) = dec2bin(0);
end
 mp_2 = md_2(i);
end

figure(5)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_2,'LineWidth',2);
xlabel('Time(sec)');
ylabel('Amplitude');
axistight
legend('Message Signal','SampledSignal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_2)]);

%For delta
delta_3 = delta;

mp_3 = 0;
for i = 1:length(ms)
ifms(i)>= mp_3
 md_3(i) = mp_3 + delta_2;
  mb_3(i) = dec2bin(1);
else
```

```
 md_3(i) = mp_3-delta_3;
 mb_3(i) = dec2bin(0);
end
 mp_3 = md_3(i);
end

figure(6)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_3,'LineWidth',2);
xlabel('Time(sec)');
ylabel('Amplitude');
axistight
legend('Message Signal','SampledSignal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_3)]);

%For 1.5*delta
delta_4 = 1.5*delta;

mp_4 = 0;
for i = 1:length(ms)
ifms(i)>= mp_4
 md_4(i) = mp_4 + delta_4;
  mb_4(i) = dec2bin(1);
else
 md_4(i) = mp_4-delta_4;
 mb_4(i) = dec2bin(0);
end
 mp_4 = md_4(i);
end

figure(7)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_4,'LineWidth',2);
xlabel('Time(sec)');
ylabel('Amplitude');
axistight
legend('Message Signal','SampledSignal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_4)]);

%For 2*delta
delta_5 = 2*delta;

mp_5 = 0;
for i = 1:length(ms)
ifms(i)>= mp_5
 md_5(i) = mp_5 + delta_5;
  mb_5(i) = dec2bin(1);
else
 md_5(i) = mp_5-delta_5;
 mb_5(i) = dec2bin(0);
end
 mp_5 = md_5(i);
end
```

```
figure(8)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_5,'LineWidth',2);
xlabel('Time(sec)');
ylabel('Amplitude');
axistight
legend('Message Signal','SampledSignal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_5)]);
```

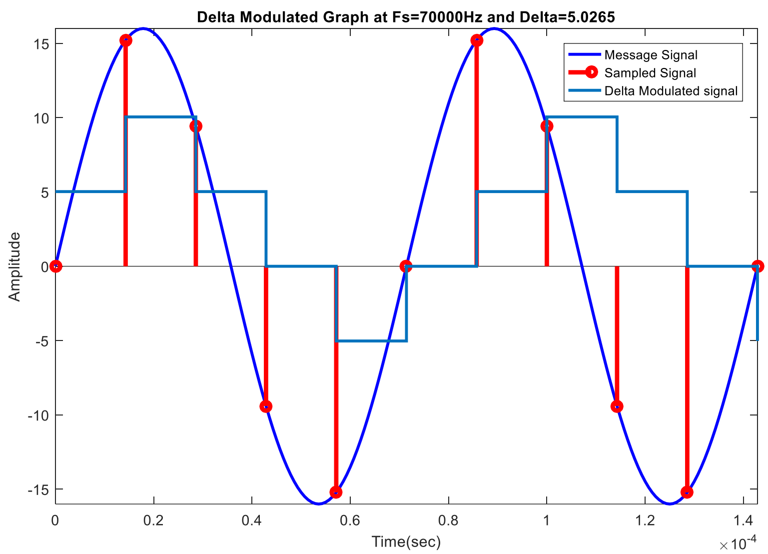

Figure 04: Time V/S Message signal Sample Signal & Delta Modulation Signal when step size, $\delta_1 = 5.02654$

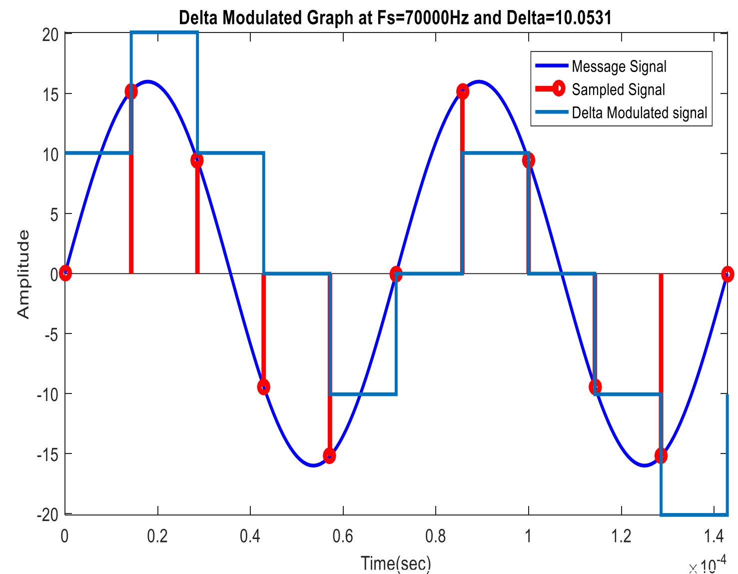

Figure 05: Time V/S Message signal Sample Signal & Delta Modulation Signal when step size, $\delta_2 = 10.0531$

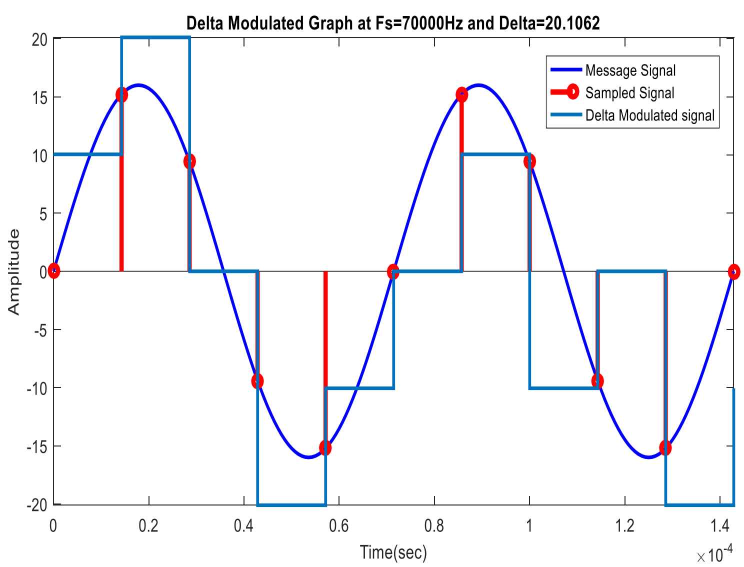

Figure 06: Time V/S Message signal Sample Signal & Delta Modulation Signal when step size, $\delta_3 = 20.10619$

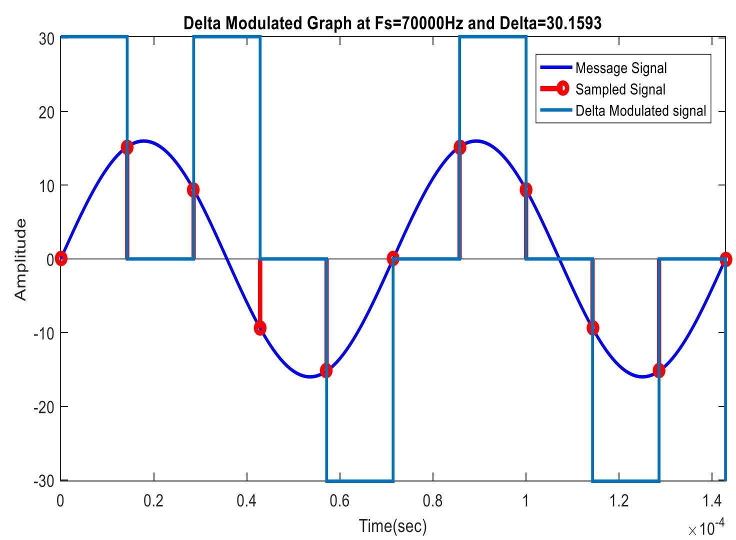

Figure 07: Time V/S Message signal Sample Signal & Delta Modulation Signal when step size, $\delta_4 = 30.1593$

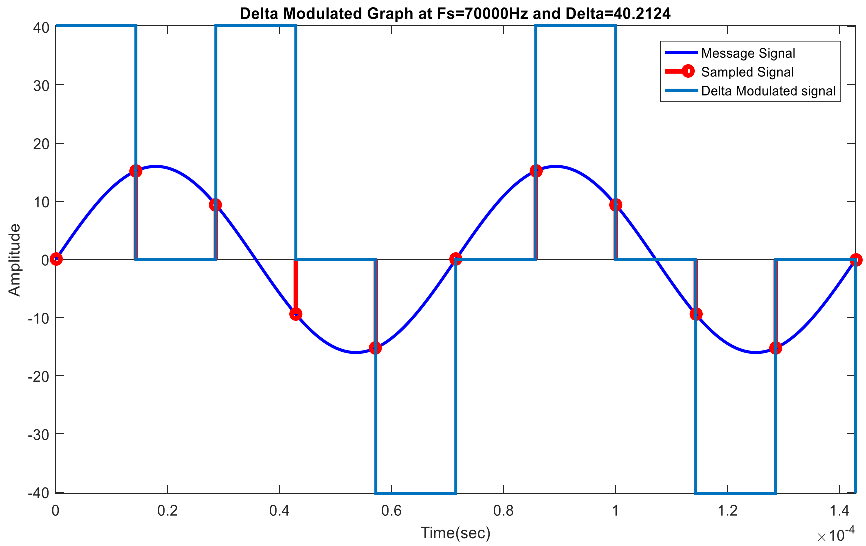

***Figure 08*:** Time V/S Message signal Sample Signal & Delta Modulation Signal when step size, δ_5 = 40.21238

<u>COMMAND WINDOW:</u>

```
Command Window
>> mb_1

mb_1 =

11000111000
```

```
Command Window
>> mb_2

mb_2 =

11000110001

fx >>
```

```
Command Window
>> mb_3

mb_3 =

11001110101

fx >>
```

```
Command Window
>> mb_4

mb_4 =

10100110010

fx >>
```

```
Command Window
>> mb_5

mb_5 =

10100110010

fx >>
```

❖ ***STEP-07***

✓ ***MATLAB CODE***

```
clc
clear
close all

X = 4;
Y = round((4+5+6+7)/4);

Am = 16;  %V (According to manual amiplitude 1Y V)
fm = 14000;  %Hz (According to manual amiplitude 1X kHz)
t = 0:1/(fm*100):2/fm; %sec

mt = Am*sin(2*pi*fm*t); %Message Signal

fs_min = 2*fm;%Hz (Minimum sampling frequency)
f_NF = fm/2; %Hz (Nyquist Frequency)
fs_min = 2*fm;%Hz (Minimum sampling frequency)
fs = 10*f_NF;
Ts = 1/fs;
ts = 0:1/fs:max(t);

ms = Am*sin(2*pi*fm*ts);%Sampled signal

%Delta = 0.25*delta
delta = ((2*pi*fm*Am)/fs)*0.25;

prev_sample = 0; %previous_sample
for k = 1:length(ts)
if ms(k) >= prev_sample
   m_quan_1(k) = prev_sample + delta;
   bo_1(k) = 1;
else
   m_quan_1(k) = prev_sample - delta;
```

```
    bo_1(k) = 0;
end
prev_sample = m_quan_1(k);
end

Quant_lvl = interp1(t,mt,ts);

prev_sample_1 = 0;
j = 0;
for i=1:length(Quant_lvl)
if prev_sample_1 < Quant_lvl(i)
    new_delta(i)= prev_sample_1 + delta;
    bo_2(i)= 1;
    if bo_2(i) == 1 && bo_2(i-j) == 1
        delta = delta+(0.20*delta);
    end
else
    new_delta(i) = prev_sample_1-delta;
    bo_2(i)= 0;
    if bo_2(i)== 0 && bo_2(i-j)== 0
        delta=delta+(0.20*delta);
    end
end

prev_sample_2=new_delta(i);
j=1;
end

figure(9)
plot(t,mt,'b','linewidth',2); hold on
stem(ts,ms,'g','linewidth',2); hold on
stairs(ts,new_delta,'r','linewidth',2); hold on
stairs(ts,m_quan_1,'k','linewidth',2);
xlabel('Time(sec)')
ylabel('Amplitude')
title(['Adaptive Delta Modulated Signal for delta=' num2str(delta)]);
legend('Message Signal','Sampled Message Signal','Adaptive Dm Signal','Dm signal');
axis tight;

%Binary Outputs
bo_1
bo_2
```

COMMAND WINDOW:

```
Command Window

bo_1 =

     1     1     0     0     0     1     1     1     0     0     0


bo_2 =

     0     1     1     0     0     0     1     1     0     0     0
```

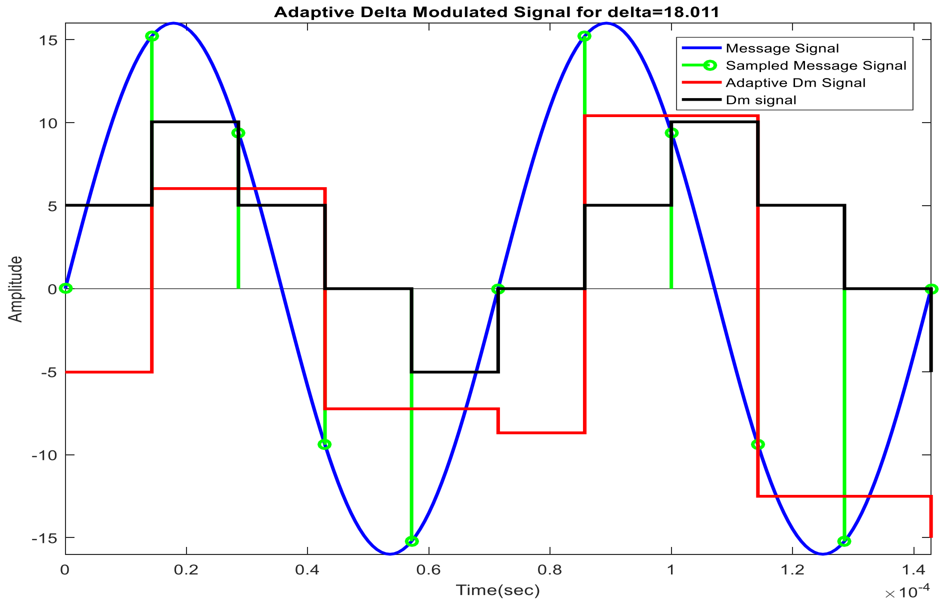

Figure 09: Delta Modulation (Adaptive step size)

❖ *STEP-08*

Adaptive delta modulation is better than delta modulation for this case. Because-

- It adjusts the step size of quantization depending on the input signal's characteristics.
- The reduction in idle noise and slope overload distortion improves the signal to noise ratio compared to ordinary delta modulation.

❖ ***STEP-09***

✓ ***MATLAB CODE***

```
clc
clear
close all

X = 4;
Y = round((4+5+6+7)/4);

Am = 16;  %V (According to manual amiplitude 1Y V)
fm = 14000;  %Hz (According to manual amiplitude 1X kHz)
t = 0:1/(fm*100):2/fm; %sec

mt = Am*sin(2*pi*fm*t); %Message Signal

fs_min = 2*fm;%Hz (Minimum sampling frequency)
f_NF = fm/2; %Hz (Nyquist Frequency)
fs_min = 2*fm;%Hz (Minimum sampling frequency)
fs = 10*f_NF;
Ts = 1/fs;
ts = 0:1/fs:max(t);

ms = Am*sin(2*pi*fm*ts);%Sampled signal
```

```
%Delta = 0.25*delta
delta = ((2*pi*fm*Am)/fs)*0.25;
fs_1 = 150000;
ts_1 = 0:1/fs_1:2/fm;

ms_1 = Am*sin(2*pi*fm*ts_1);

%for 0.25 delta
delta_1 = ((Am*2*pi*fm)/fs_1)*0.25;

prev_sample_3 = 0; %previous_sample

for m = 1:length(ts_1)
if ms_1(m) >= prev_sample_3
   m_quant_2(m) = prev_sample_3 + delta_1;
   bo_3(m) = 1;
else
   m_quant_2(m) = prev_sample_3 - delta_1;
   bo_3(m) = 0;
end
prev_sample_3 = m_quant_2(m);
end

Quant_lvl_1 = interp1(t,mt,ts_1);

prev_sample_4=0;
r = 0;
for q = 1:length(Quant_lvl_1)
if prev_sample_4 < Quant_lvl_1(q)
   new_delta_1(q)= prev_sample_4 + delta_1;
   bo_4(q)= 1;
   if bo_4(q) == 1 && bo_4(q-r) == 1
      delta_1 = delta_1+(0.20*delta_1);
   end
else
   new_delta_1(q)=prev_sample_4-delta_1;
   bo_4(q)= 0;
   if bo_4(q)== 0 && bo_4(q-r)== 0
      delta_1=delta_1+(0.20*delta_1);
   end
end
if ((bo_4(q)== 1  && bo_4(q-r)== 0) || (bo_4(q)== 0) && bo_4(q-r)== 1)
   delta_1 = 6.4;
end
prev_sample_4=new_delta_1(q);
r=1;
end

figure(10)
plot(t,mt,'b','linewidth',2); hold on
stem(ts_1,ms_1,'g','linewidth',2); hold on
stairs(ts_1,new_delta_1,'r','linewidth',2); hold on
stairs(ts_1,m_quant_2,'k','linewidth',2);
xlabel('Time')
ylabel('Amplitude')
title(['Adaptive Delta Modulated Signal for delta=' num2str(delta_1)])
legend('Message Signal','Sampled Message Signal','Adaptive Dm Signal','Dm signal');
```

```
axis tight;

%Binary Outputs
bo_3
bo_4
```

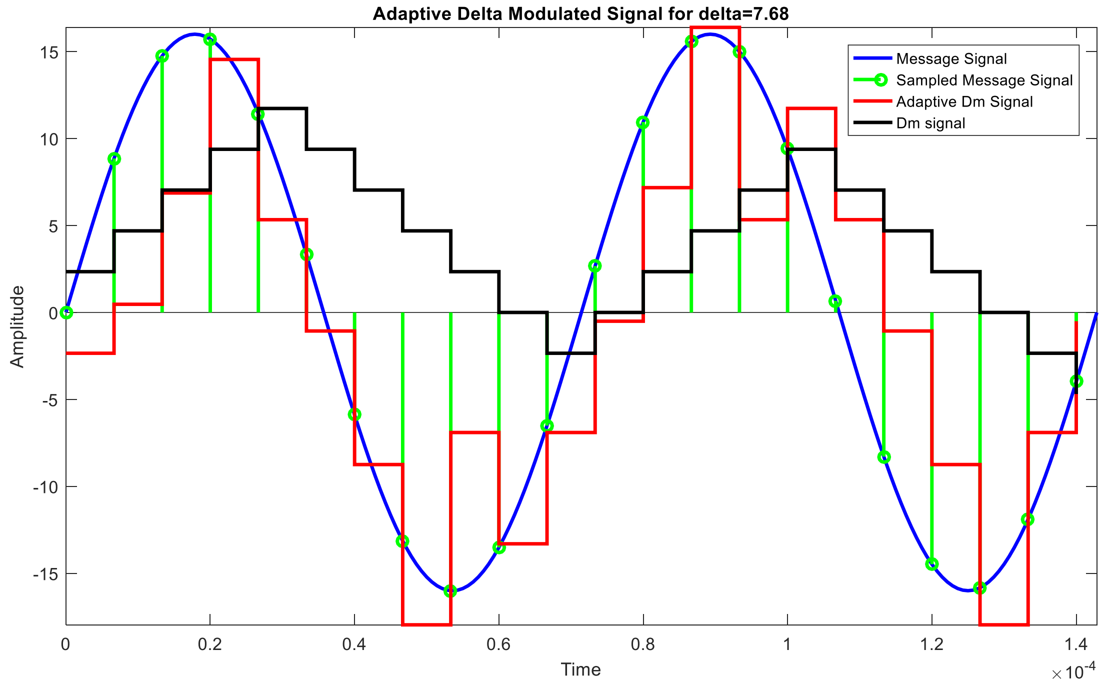

***Figure 10*:** Delta Modulation (Adaptive step size) for finest value

Comment: As we change the sample frequency to 150000, the sample time, sample signal and delta values change.

CONCLUSION

In this project, to test the impact of modulated signal we used delta modulation with fixed step size and also did adaptive delta modulation. We decreased and increased the value of step size for no slope overload noise to see the impact of the modulated signal. Then we do adaptive delta modulation and compare it with delta modulation with fixed step size. We changed all the necessary parameters in the adaptive delta modulation for the best staircase approximation.

APPENDIX

Matlab code for Step (01-06)

```
%Group-04
%Kazi Iftier Rahman (2020-1-80-004)
%Arpon Podder (2020-1-80-005)
%Rohit Bhowmick (2020-1-80-006)
%Farhana Misu(2020-1-80-027)

clc
close all
clear

%Step:01

X = 4;
Y = round((4+5+6+7)/4);

Am = 16;  %V (According to manual amiplitude Am = 1Y V)
fm = 14000;  %Hz (According to manual amiplitude fm = 1X kHz)
t=0:1/(fm*100):2/fm; %sec

mt = Am*sin(2*pi*fm*t); %Message Signal

figure(1)
plot(t,mt,'LineWidth',2);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Message Signal');
axis tight

%Step:02

f_NF = fm/2; %Hz (Nyquist Frequency)
fs_min = 2*fm;%Hz (Minimum sampling frequency)
fs = 10*f_NF;
Ts = 1/fs;
ts = 0:1/fs:max(t);

ms = Am*sin(2*pi*fm*ts);%Sampled signal

figure(2)
plot(t,mt,'b','linewidth',2); hold on
stem(ts,ms,'r','LineWidth',2);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Message Signal and Sampled signal');
legend('Message signal','Sampled signal');
axis tight

%Step:03

delta = (2*pi*fm*Am)/fs; %step size, for which there will be no slope overload noise.

%Step:04
```

```
mp = 0;
for i = 1:length(ms)
 if ms(i)>= mp
 md(i) = mp + delta;
  mb(i) = dec2bin(1);
 else
 md(i) = mp-delta;
 mb(i) = dec2bin(0);
 end
 mp = md(i);
end

figure(3)
stairs(ts,md,'LineWidth',2);
xlabel('Time(sec)');
ylabel('Amplitude');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta)]);
axis tight

%Step:05

Slope_Max = delta*fs;
Noise_SO = Slope_Max-delta;
Noise_Granular = sqrt(delta/12);

%Step:06
delta_1 = 0.25*delta;

mp_1 = 0;
for i = 1:length(ms)
 if ms(i)>= mp_1
 md_1(i) = mp_1 + delta_1;
  mb_1(i) = dec2bin(1);
 else
 md_1(i) = mp_1-delta_1;
 mb_1(i) = dec2bin(0);
 end
 mp_1 = md_1(i);
end

figure(4)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_1,'LineWidth',2);
axis tight
legend('Message Signal','Sampled Signal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_1)]);

delta_2 = 0.5*delta;

mp_2 = 0;
for i = 1:length(ms)
 if ms(i)>= mp_2
 md_2(i) = mp_2 + delta_2;
  mb_2(i) = dec2bin(1);
 else
 md_2(i) = mp_2-delta_2;
 mb_2(i) = dec2bin(0);
```

```
 end
 mp_2 = md_2(i);
end

figure(5)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_2,'LineWidth',2);
axis tight
legend('Message Signal','Sampled Signal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_2)]);

delta_3 = delta;

mp_3 = 0;
for i = 1:length(ms)
 if ms(i)>= mp_3
 md_3(i) = mp_3 + delta_2;
  mb_3(i) = dec2bin(1);
 else
 md_3(i) = mp_3-delta_3;
 mb_3(i) = dec2bin(0);
 end
 mp_3 = md_3(i);
end

figure(6)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_3,'LineWidth',2);
axis tight
legend('Message Signal','Sampled Signal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_3)]);


delta_4 = 1.5*delta;

mp_4 = 0;
for i = 1:length(ms)
 if ms(i)>= mp_4
 md_4(i) = mp_4 + delta_4;
  mb_4(i) = dec2bin(1);
 else
 md_4(i) = mp_4-delta_4;
 mb_4(i) = dec2bin(0);
 end
 mp_4 = md_4(i);
end

figure(7)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_4,'LineWidth',2);
axis tight
legend('Message Signal','Sampled Signal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_4)]);

delta_5 = 2*delta;
```

```
mp_5 = 0;
for i = 1:length(ms)
 if ms(i)>= mp_5
 md_5(i) = mp_5 + delta_5;
  mb_5(i) = dec2bin(1);
 else
 md_5(i) = mp_5-delta_5;
 mb_5(i) = dec2bin(0);
 end
 mp_5 = md_5(i);
end

figure(8)
plot(t,mt,'b','LineWidth',2); hold on
stem(ts,ms,'r','LineWidth',3); hold on
stairs(ts,md_5,'LineWidth',2);
axis tight
legend('Message Signal','Sampled Signal','Delta Modulated signal');
title(['Delta Modulated Graph at Fs=' num2str(fs) 'Hz and Delta=' num2str(delta_5)]);

%Binary Outputs
mb
mb_1
mb_2
mb_3
mb_4
mb_5
```

Matlab code for Step (07-09)

```
%Group-04
%Kazi Iftier Rahman (2020-1-80-004)
%Arpon Podder (2020-1-80-005)
%Rohit Bhowmick (2020-1-80-006)
%Farhana Misu(2020-1-80-027)

%Step:07

clc
clear
close all

X = 4;
Y = round((4+5+6+7)/4);

Am = 16;  %V (According to manual amiplitude 1Y V)
fm = 14000;  %Hz (According to manual amiplitude 1X kHz)
t = 0:1/(fm*100):2/fm; %sec

mt = Am*sin(2*pi*fm*t); %Message Signal

fs_min = 2*fm;%Hz (Minimum sampling frequency)
f_NF = fm/2; %Hz (Nyquist Frequency)
```

```
fs_min = 2*fm;%Hz (Minimum sampling frequency)
fs = 10*f_NF;
Ts = 1/fs;
ts = 0:1/fs:max(t);

ms = Am*sin(2*pi*fm*ts);%Sampled signal

%Delta = 0.25*delta
delta = ((2*pi*fm*Am)/fs)*0.25;

prev_sample = 0; %previous_sample
for k = 1:length(ts)
if ms(k) >= prev_sample
    m_quan_1(k) = prev_sample + delta;
    bo_1(k) = 1;
else
    m_quan_1(k) = prev_sample - delta;
    bo_1(k) = 0;
end
prev_sample = m_quan_1(k);
end

Quant_lvl = interp1(t,mt,ts);

prev_sample_1 = 0;
j = 0;
for i=1:length(Quant_lvl)
if prev_sample_1 < Quant_lvl(i)
    new_delta(i)= prev_sample_1 + delta;
    bo_2(i)= 1;
    if bo_2(i) == 1 && bo_2(i-j) == 1
        delta = delta+(0.20*delta);
    end
else
    new_delta(i) = prev_sample_1-delta;
    bo_2(i)= 0;
    if bo_2(i)== 0 && bo_2(i-j)== 0
        delta=delta+(0.20*delta);
    end
end

prev_sample_2=new_delta(i);
j=1;
end

figure(9)
plot(t,mt,'b','linewidth',2); hold on
stem(ts,ms,'g','linewidth',2); hold on
stairs(ts,new_delta,'r','linewidth',2); hold on
stairs(ts,m_quan_1,'k','linewidth',2);
xlabel('Time(sec)')
ylabel('Amplitude')
title(['Adaptive Delta Modulated Signal for delta=' num2str(delta)]);
legend('Message Signal','Sampled Message Signal','Adaptive Dm Signal','Dm signal');
axis tight;

%Step:08
Slope_Max = delta*fs;
```

```
Noise_SO = Slope_Max-delta;
Noise_Granular = sqrt(delta/12);

%Step:09

fs_1 = 150000;
ts_1 = 0:1/fs_1:2/fm;

ms_1 = Am*sin(2*pi*fm*ts_1);

%for 0.25 delta
delta_1 = ((Am*2*pi*fm)/fs_1)*0.25;

prev_sample_3 = 0; %previous_sample

for m = 1:length(ts_1)
if ms_1(m) >= prev_sample_3
    m_quant_2(m) = prev_sample_3 + delta_1;
    bo_3(m) = 1;
else
    m_quant_2(m) = prev_sample_3 - delta_1;
    bo_3(m) = 0;
end
prev_sample_3 = m_quant_2(m);
end

Quant_lvl_1 = interp1(t,mt,ts_1);

prev_sample_4=0;
r = 0;
for q = 1:length(Quant_lvl_1)
if prev_sample_4 < Quant_lvl_1(q)
    new_delta_1(q)= prev_sample_4 + delta_1;
    bo_4(q)= 1;
    if bo_4(q) == 1 && bo_4(q-r) == 1
        delta_1 = delta_1+(0.20*delta_1);
    end
else
    new_delta_1(q)=prev_sample_4-delta_1;
    bo_4(q)= 0;
    if bo_4(q)== 0 && bo_4(q-r)== 0
        delta_1=delta_1+(0.20*delta_1);
    end
end
if ((bo_4(q)== 1  && bo_4(q-r)== 0) || (bo_4(q)== 0) && bo_4(q-r)== 1)
    delta_1 = 6.4;
end
prev_sample_4=new_delta_1(q);
r=1;
end

figure(10)
plot(t,mt,'b','linewidth',2); hold on
stem(ts_1,ms_1,'g','linewidth',2); hold on
stairs(ts_1,new_delta_1,'r','linewidth',2); hold on
stairs(ts_1,m_quant_2,'k','linewidth',2);
xlabel('Time')
ylabel('Amplitude')
```

```
title(['Adaptive Delta Modulated Signal for delta=' num2str(delta_1)])
legend('Message Signal','Sampled Message Signal','Adaptive Dm Signal','Dm signal');
axis tight;

%Binary Outputs
bo_1
bo_2
bo_3
bo_4
```