Bangladesh University of Engineering and Technology

BUET

Department of Electrical and Electronic Engineering

Course No: EEE212

Course Tittle : Numerical Technique Analysis

Project Name : **Wireless transmission of signal by ASK and FSK Modulation**

Prepared For:

Mr. Hamidur Rahman	Shahed Ahmed	Shamiur Salehin Akash
Associate Professor (EEE)	Lecturer of EEE	Lecturer of EEE (PT)
BUET, Dhaka.	BUET, Dhaka.	BUET, Dhaka.

Submitted By :

Name : Md Abu Sayed Chowdhury &

Md Sharif Uddin

ID : 1906076 &

1906077

Lab Group : 10

Section : B

Level: 2 **Term**: 1 **Date Of Submission** : 24-02-2022

Table of Contents:

List of Illustrators:

1.Introduction:

Pulse-Code Modulation is a method used to digitally represent sampled analogue signal. Most signal of practical interest such as speech, biological signals and communication signals are analogue. These signals may have to processed for different purposes. Digital signal processing of an analogue signal is preferable to processing the signals directly in the analog domain because of its flexibility in reconfiguration, better accuracy and control, better storing capability and cost effectiveness. Pulse Code modulation is a digital representation of an analog signal that takes samples of the amplitude of the analog signal at regular intervals. As we cannot transmit analogue signal, the signal has to transmit by converting it into a digital signal by an analog to digital converter After transmitting this signal (DAC) converts the digital signal into the original signal. The main objective is the simulation of PCM scheme for wireless transmission of signal, signals on MATLAB and observing the effect of adding Additive White Gaussian Noise (AWGN) for ASK, FSK modulation scheme.

2.Theory:

Modulation can be defined as the process of changing the carrier signal parameters by the instant values of the message signal. The transmission of message signal can be done mainly for communication and the most common digital modulation – technique is PCM of Pulse Code Modulation. In PCM, the message signal can be signified thorough a series of code impulses. It is a method that is used to convert an analog signal into a digital signal so that a modified analog signal can be transmitted through digital communication network. PCM is in binary form, so there will be only two possible states high and low (0 and 1). We can also get back our analog signal by demodulation. The Pulse Code modulation process is done in three steps-sampling, quantization and coding. The basic elements of PCM mainly include the transmitter section and receiver section.

2.1.ASK:

The short form of Amplitude Shift Keying is referred as ASK. It is the digital modulation technique. In this technique, amplitude of the RF carrier is varied in accordance with baseband digital input signal. The figure depicts operation of ASK modulation. As shown in the figure,

binary 1 will be represented by carrier signal with some amplitude while binary 0 will be represented by carrier of zero amplitude (i.e., no carrier).

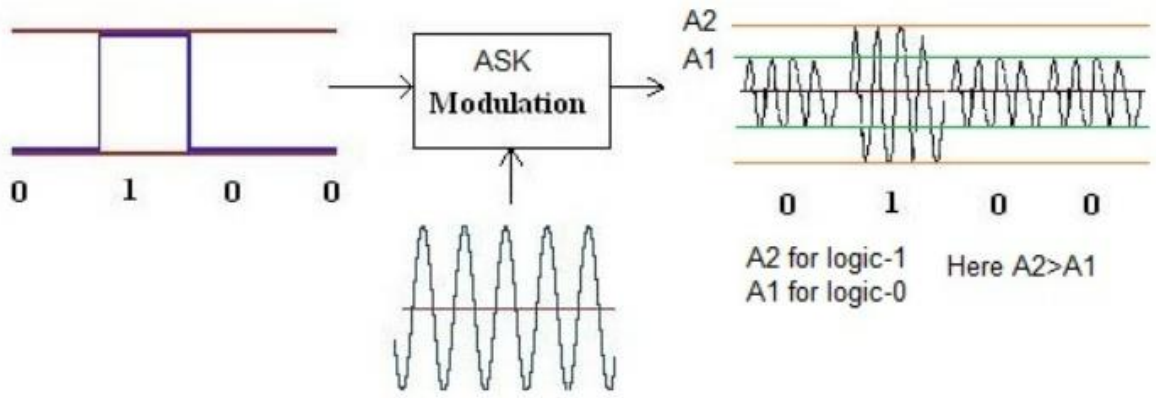


Fig 1: ASK Modulation

ASK modulation can be represented by following equation:
$s(t) = A2 * \cos(2*\pi*fc*t)$ for Binary Logic-1
$s(t) = A1 * \cos(2*\pi*fc*t)$ for Binary Logic-0
Here A2>A1

Often in ASK modulation, binary-1 is represented by carrier with amplitude-A2 and binary-0 is represented by carrier with amplitude-A1. Here A2 is greater in magnitude compare to A1.

2.2.FSK:

The short form of Frequency Shift Keying is referred as FSK. It is also digital modulation technique. In this technique, frequency of the RF carrier is varied in accordance with baseband digital input. The figure depicts the FSK modulation. As shown, binary 1 and 0 is represented by two different carrier frequencies. Figure depicts that binary 1 is represented by high frequency 'f1' and binary 0 is represented by low frequency 'f2'.

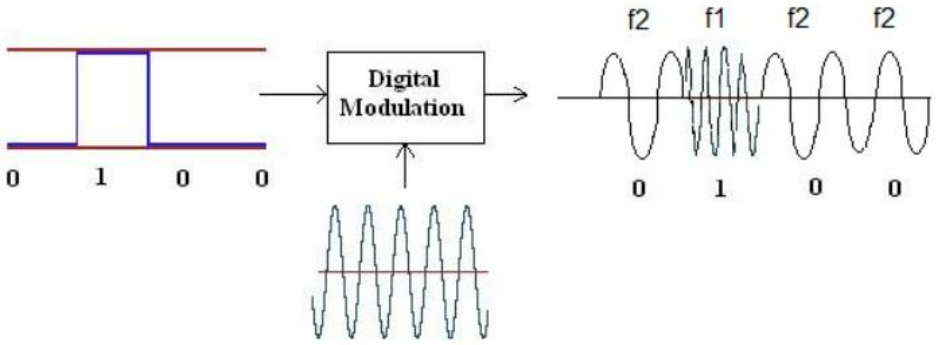


Fig 2 : FSK Modulation

Binary FSK can be represented by following equation:
s(t) = A* cos(2*π*f1*t) for Binary 1
s(t) = A* cos(2*π*f2*t) for Binary 0

3.Algorithm:

3.1. Analog to digital signal

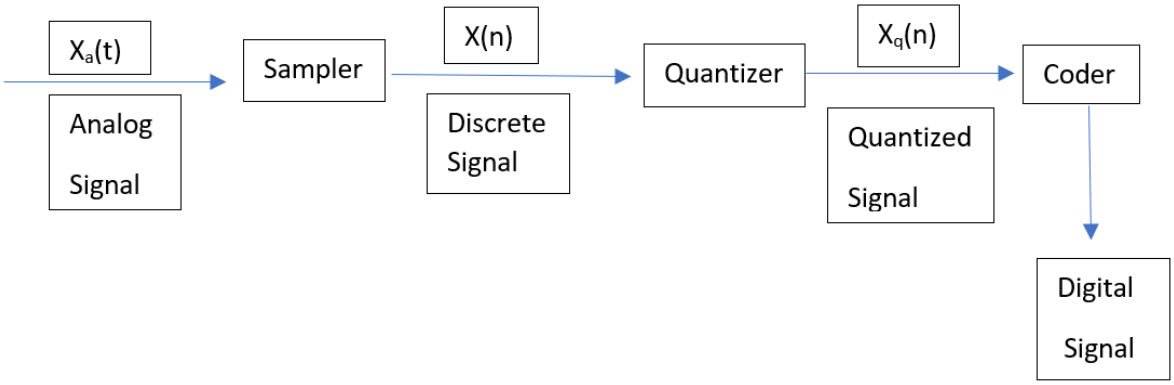


Fig3 : Analog to digital signal

a) Sampling:

By performing a sampling, a continuous time signal is converted into a discrete time signal by taking samples at discrete time instants as shown:

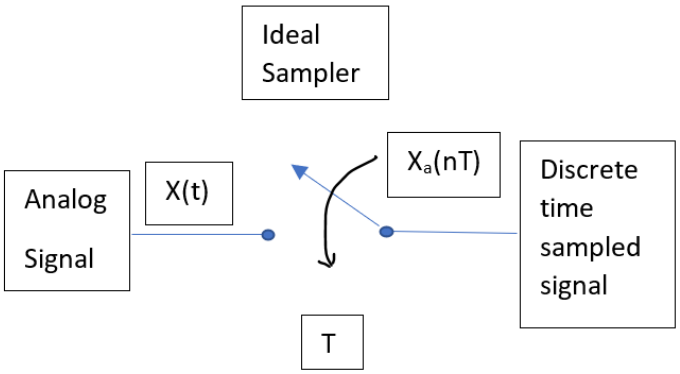
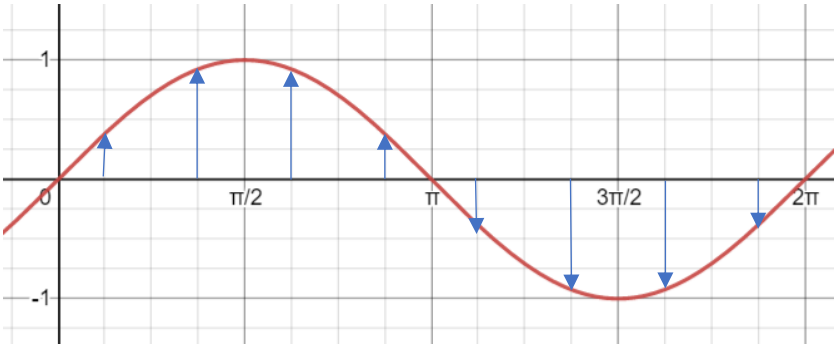


Fig 4 : Sampling of an analog signal

The analog signal is sampled once every T second, resulting in a sampled data sequence. The most important parameter in a sampling process is the sampling frequency, f_s defined as $T=(1/f_s)$, Where T denotes the sampling period. For a sinusoidal signal-sampling process is given below given below



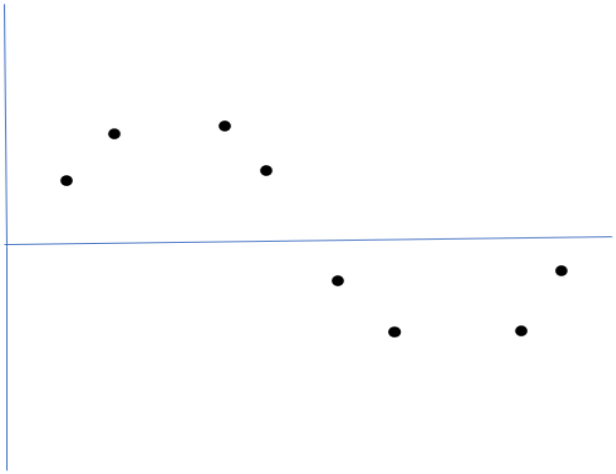


Fig 5 : Sampling in Signal

3.2. Quantization:

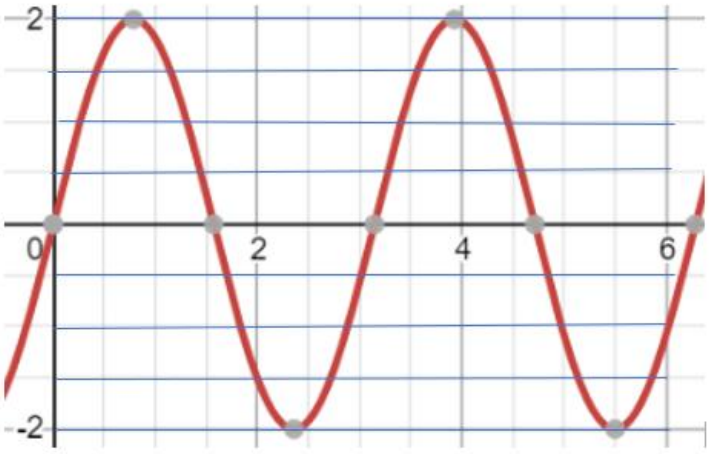


Fig 6 : Quantization

Quantization is the process of taking a continuous voltage signal and mapping it to a discrete number of voltage levels. The number of quantization levels is usually a power of 2 ($N=2^n$). Where, n is the number of quantization bits. The distance Δ, between two successive quantization levels is called the quantization levels is called the quantization step size or resolution. If X_{min} and X_{max} represent the minimum and maximum value of X(n) and L, the number of quantization levels, then

$$\Delta=\frac{Xmax-Xmin}{L-1}$$

If X_{max} is positive and X_{min} is negative and if b is the number of bits, then,

$$\Delta=\frac{2*V}{2^b-1}$$

The quantization error is given by $e(n)=X_q(n)-X(n)$

3.3.Quantized Signal to Binary to Digital:

Here we first make the quantized signal to a 4 bit binary signal. The 4 bits are a column matrix. So we make it a row to vector in order to make it a digital signal.

3.4. Modulation:

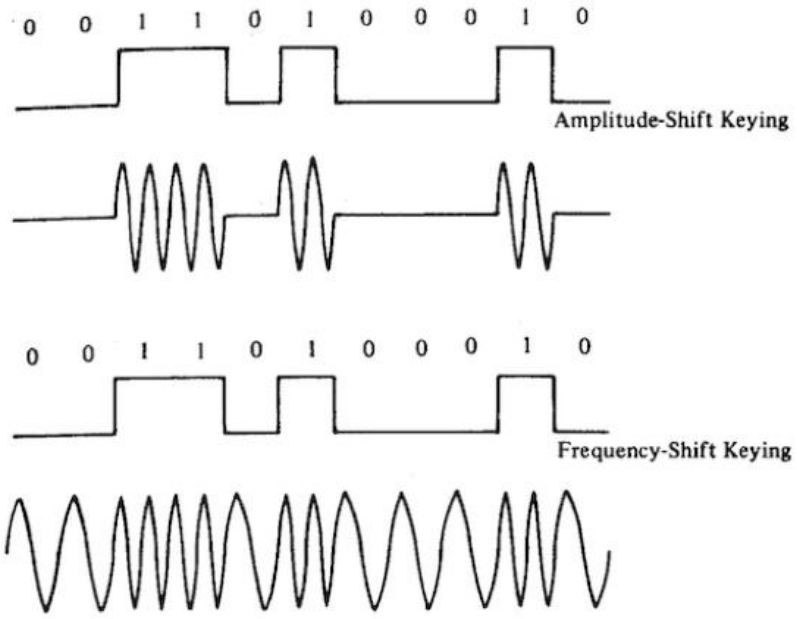
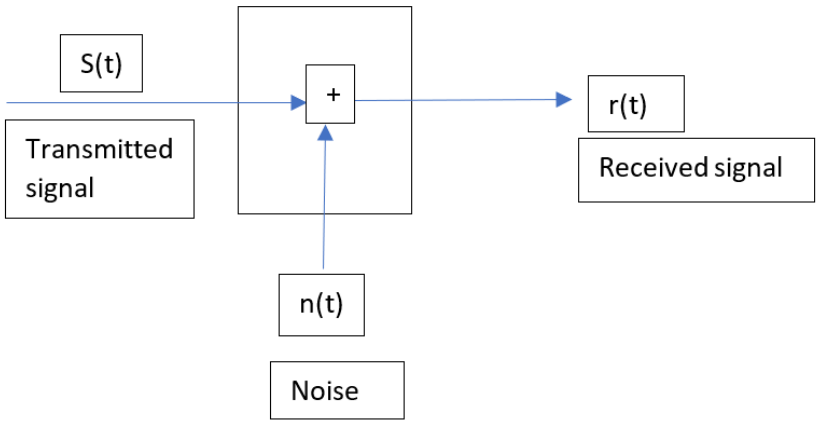


Fig 7 : ASK and PSK (Source: Internet)

i. ASK: Here we multiply the Digital signal with a carrier signal.

ii. FSK: Here we substitute the 0 portion with low frequency signal and 1 portion with high frequency signal.

3.5. Noise in Channel: Every channel has some noise. Here we used Additive White Gaussian Noise.



3.6. Noise cancellation:

Fig 8 : Noise Cancellation

We first rectified the signal and then passed the signal through an integrator in order to smooth the signal

3.7. Reconstruction:

Here, Digital to Analog converter is activated. Then at the receiver end, a pulse code modulator decodes the binary signal back into pulses with the same quantum levels as those in the modulator.

4. Flow Chart:

The block diagram given below describes the whole process of PCM.

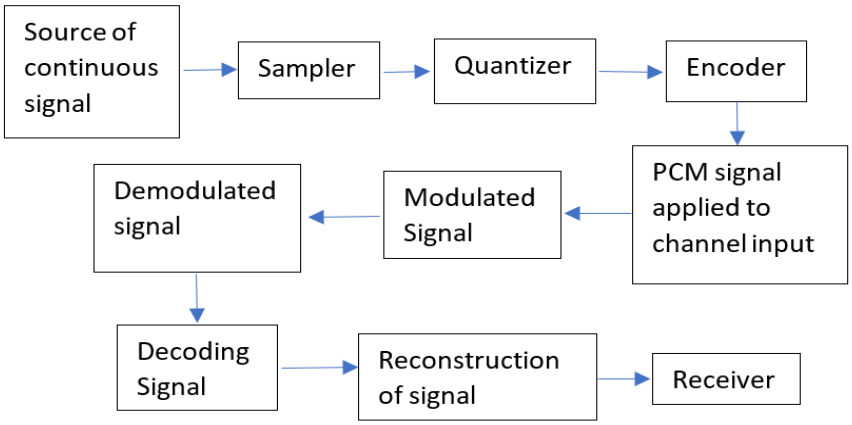


Fig 9 : Block Diagram of PCM

5.Code:

5.1. ASK:

```
%% part 1 : Input function and Sampling
value=app.GraphDropDown.Value;
ax0=app.UIAxes;
app.UIAxes.FontSize=20;
```

```
s=app.StartingtimeEditField.Value;
e=app.EndingtimeEditField.Value;
f=str2func(app.FunctionEditField.Value);
p=app.SamplingperiodEditField.Value;
t=s:p:e;
y=f(t);
%Nyquist theorem
temp=(p/100);
tempt=0:0.1:20*2*pi;
tempy=f(tempt);
ac=xcorr(tempy,tempy);
[~,locs]=findpeaks(ac);
d=mean(diff(locs)*0.1)/2;
if(p>d)
app.Label.Text=strcat('Warning :Sampling period should be less than',num2str(d));
app.Label.FontColor='r';
return;
else
app.Label.Text='';
end
%
if strcmp(value,'Actual Signal')
plot(app.UIAxes,t,y);
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='displacement';
app.UIAxes.Title.String='Actual Signal';
app.UIAxes.XLim=[s e];
app.UIAxes.YLim=[min(y) max(y)];
app.Label_2.Text='';
end
if strcmp(value,'Actual Signal')
return
end
%% Part2: Analogue to binary signal
% Quantization (4 bits)
b=4;
vmax=max(y);
vmin=min(y);
del=(vmax-vmin)/(2^b-1);
lev=[vmin:del:vmax];
for i=1:length(y)
for j=1:length(lev)
if abs(y(i)-lev(j))<del/2
xq(i)=lev(j); %quantization level
xd(i)=j-1; %index of level
break;
end
end
end
%% binary siganl
xdig=de2bi(xd,'left-msb');
dig=[];
```

```
for i=1:length(xdig) %digital value
dig=[dig xdig(i,:)];
end
if strcmp(value,'Digitized Signal')
stem(app.UIAxes,dig);axis tight %discrete signal plot
app.UIAxes.XLabel.String='n';
app.UIAxes.YLabel.String='displacement';
app.UIAxes.Title.String='Digitized Signal';
app.UIAxes.XLim=[0 length(dig)];
app.UIAxes.YLim=[-0.2 1.2];
app.Label_2.Text='';
return;
end
%% part 3 : ASK & Noise
% ASK
m=dig; %copying the dig array to m
Tb=1;
Fc=10;
t=[0:0.01:Tb];
c= sqrt(2/Tb)*sin(2*pi*Fc*t); %carrier signal
n=length(dig);
t1=0;
t2=Tb;
for i=1:length(dig)
t=[t1:0.01: t2];
if m(i)>0.5
m(i)=1;
ms=ones(1, length(t));
else
m(i)=0;
ms=zeros(1,length(t));
end
message(i,: )=ms;
ask(i,: )=c.*ms;
t1=t1+Tb;
t2=t2+Tb;
if strcmp(value,'Message Signal')
plot(app.UIAxes,t,message(i,:),'b');axis tight; %discrete signal plot
hold(ax0,'on');
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='m(t)';
app.UIAxes.Title.String='Digital Message Signal of the Binary Bits';
app.UIAxes.YLim=[-0.2 1.2];
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
if strcmp(value,'ASK Modulated Signal')
plot(app.UIAxes,t,ask(i,:),'b');axis tight;
hold(ax0,'on');%discrete signal plot
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='a(t)';
app.UIAxes.Title.String='ASK Modulation of the Digital Message Signal';
```

```
app.UIAxes.YLim=[-2 2];
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
end
hold(ax0,'off');
if strcmp(value,'Message Signal') || strcmp(value,'ASK Modulated Signal')
return
end
%% noise
t1=0;
t2=Tb;
for i=1:length(dig)
ask(i,:)=awgn(ask(i,:),5); %signal to noise ratio=5
t=[t1:0.01:t2];
if strcmp(value,'ASK Modulated Signal With Noise')
plot(app.UIAxes,t,ask(i,:),'b');axis tight; %discrete signal plot
hold(ax0,'on');
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='a(t) with noise';
app.UIAxes.Title.String='ASK Modulation of the Digital Message Signal';
app.UIAxes.XLim=[0 n];
app.UIAxes.YLim=[-5 5];
app.Label_2.Text='';
end
t1=t1+Tb;
t2=t2+Tb;
end
hold(ax0,'off');
if strcmp(value,'ASK Modulated Signal With Noise')
return
end
%% part 4: (ASK demodulation)
%rectified signal
t1=0;
t2=Tb;
for i=1:length(dig)
t=[t1:0.01:t2];
for j=1:length(t)
if ask (i,j)>0
rect(i,j)=ask(i,j);
else
rect(i,j)=0;
end
end
t1=t1+Tb;
t2=t2+Tb;
if strcmp(value,'Rectified Signal')
plot(app.UIAxes,t,rect(i,:),'b');axis tight; %discrete signal plot
hold(ax0,'on');
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='r(t)';
```

```
app.UIAxes.Title.String='Rectified Signal';
app.UIAxes.YLim=[0 5];
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
end
hold(ax0,'off');
if strcmp(value,'Rectified Signal')
return;
end
%% smoothing
t1=0;
t2=Tb;
for i=1:length(dig)
t=[t1:0.01:t2];
smoothed (i,:)=smooth(rect(i,:),500); % smoothing function
t1=t1+Tb;
t2=t2+Tb;
if strcmp(value,'Smooth Signal')
plot(app.UIAxes,t,smoothed(i,:),'b');axis tight; %discrete signal plot
hold(ax0,'on');
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='s(t)';
app.UIAxes.Title.String='Smooth Signal';
app.UIAxes.YLim=[-0.2 3];
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
end
hold(ax0,'off');
if strcmp(value,'Smooth Signal')
return;
end
%% Demodulated binary data
t1=0;
t2=Tb;
for i=1:length(dig)
t=[t1:.01:t2];
decision_val=sum( smoothed(i,:))/length(t); %average value
average(i,:)=decision_val;
t1=t1+Tb;
t2=t2+Tb;
end
max_avg= max(average);
min_avg=min(average);
avg=((max_avg)+(min_avg))/2;
for i=1:length(dig)
if average(i,:) < avg
x(i)=0;
else
x(i)=1;
end
```

```
end
if strcmp(value,'ASK Demodulated Signal')
stem(app.UIAxes,x,'b');hold on;axis tight; %discrete signal plot
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='x(n)';
app.UIAxes.Title.String='ASK Demodulated Binary Data(decoded signal)';
app.UIAxes.YLim=[min(x)-0.2 max(x)+0.2];
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
hold(ax0,'off');
if strcmp(value,'ASK Demodulated Signal')
return;
            end
            %% bit error ratio
            E=x-dig;
            BER=0;
            for i=1:length(y)
                if E(i)~=0
                    BER=BER+1;     %bit error ratio
                end
            end
            BE_Ratio=BER/length(E);
                        %% part 5 (decoding)
            % binary to decimal
            b=4;
            j=1;
            samp=length(x)/b;
            for i=1:b:length(x)
                dig_row(j,:)=x(1,i:i+3);
                j=j+1;
            end
            for i=1:size(dig_row,1)
                lev(i)=bi2de(dig_row(i,:),'left-msb');
            end
```

```
%%  reconstract signal
levvalue=lev*del+min(y);
if strcmp(value,'Decoded Quantized Signal')
    stem(app.UIAxes,levvalue,'b');axis tight;  %discrete signal plot
    app.UIAxes.XLabel.String='Index';
    app.UIAxes.YLabel.String='Displacement';
    app.UIAxes.Title.String='Decoded Quantized Signal';
    app.UIAxes.XLim=[0 n/4];
    app.UIAxes.YLim=[min(levvalue)-0.2 max(levvalue)+0.2];
    app.Label_2.Text=strcat('Bit error ratio: ',num2str(BE_Ratio));
    return;
end
td=linspace(0,10,length(levvalue));
t=0:0.001:10;
xre=interp1(td,levvalue,t,'spline');
if strcmp(value,'Reconstructed Signal')
    plot(app.UIAxes,t,xre);axis tight;  %discrete signal plot
    app.UIAxes.XLabel.String='time';
    app.UIAxes.YLabel.String='Displacement';
    app.UIAxes.Title.String='Reconstructed Signal';
    app.UIAxes.XLim=[0 10];
    app.UIAxes.YLim=[min(levvalue)-0.2 max(levvalue)+0.2];
    app.Label_2.Text=strcat('Bit Error Ratio: ',num2str(BE_Ratio));
    return;
end
```

5.2. FSK:

FSK has the same code , except ASK part is replaced by the code below:

```
%% part 1 : Input function and Sampling
value=app.GraphDropDown.Value;
ax0=app.UIAxes;
app.UIAxes.FontSize=20;
%input
s=app.StartingtimeEditField.Value;
e=app.EndingtimeEditField.Value;
p=app.SamplingperiodEditField.Value;
f=str2func(app.FunctionEditField.Value);
%sampling
t=s:p:e;
y=f(t);
%Nyquist theorem
temp=(p/100);
tempt=0:0.1:20*2*pi;
tempy=f(tempt);
ac=xcorr(tempy,tempy);
[~,locs]=findpeaks(ac);
```

```
d=mean(diff(locs)*0.1)/2;
if(p>d)
app.Label.Text=strcat('Warning :Sampling period should be less than',num2str(d));
app.Label.FontColor='r';
return;
else
app.Label.Text='';
end
%
if strcmp(value,'Actual Signal')
plot(app.UIAxes,t,y);
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='displacement';
app.UIAxes.Title.String='Actual Signal';
app.UIAxes.XLim=[s e];
app.UIAxes.YLim=[min(y) max(y)];
app.Label_2.Text='';
end
if strcmp(value,'Actual Signal')
return
end
%% Part2: Analogue to binary signal
% Quantization (4 bits)
b=4;
vmax=max(y);
vmin=min(y);
del=(vmax-vmin)/(2^b-1);
lev=[vmin:del:vmax];
for i=1:length(y)
for j=1:length(lev)
if abs(y(i)-lev(j))<del/2
xq(i)=lev(j); %quantization level
xd(i)=j-1; %index of level
break;
end
end
end
%% binary siganl
xdig=de2bi(xd,'left-msb');
dig=[];
for i=1:length(xdig) %digital value
dig=[dig xdig(i,:)];
end
if strcmp(value,'Digitized Signal')
stem(app.UIAxes,dig);axis tight %discrete signal plot
app.UIAxes.XLabel.String='n';
app.UIAxes.YLabel.String='displacement';
app.UIAxes.Title.String='Digitized Signal';
app.UIAxes.XLim=[0 length(dig)];
app.UIAxes.YLim=[-0.2 1.2];
app.Label_2.Text='';
return;
```

```
end
%% FSK
n=length(dig);
m=dig; %copying the dig array to m
Tb=1;
fc1=1000;
fc2=2;
t=[0:0.01:Tb];
c1= sqrt(2/Tb)*sin(2*pi*fc1*t); %carrier signal 1 (1000hz)
c2= sqrt(2/Tb)*sin(2*pi*fc2*t); %carrier signal 2 (2 hz)
t1=0;
t2=Tb;
for i=1:length(dig)
t=[t1:0.01: t2];
if m(i)>0.5
m(i)=1;
m_s=ones(1, length(t));
else
m(i)=0;
m_s=zeros(1,length(t));
end
message (i,:)=m_s; %tidy
if m(i)==1
fsk_signal (i,:)=c1;
else
fsk_signal (i,:)=c2;
end
t1=t1+Tb;
t2=t2+Tb;
if strcmp(value,'Message Signal')
plot(app.UIAxes,t,message(i,:),'b');
hold(ax0,'on');axis tight; %discrete signal plot
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='m(t)';
app.UIAxes.Title.String='Digital Message Signal of the Binary Bits';
app.UIAxes.YLim=[-2 2];
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
if strcmp(value,'FSK Modulated Signal')
plot(app.UIAxes,t,fsk_signal(i,:),'b');
hold(ax0,'on');axis tight; %discrete signal plot
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='a(t)';
app.UIAxes.Title.String='FSK Modulation of the Digital Message Signal';
%app.UIAxes.YLim=[-2 2];
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
end
hold(ax0,'off');
if strcmp(value,'Message Signal') || strcmp(value,'FSK Modulated Signal')
```

```
return
end
%% noise
t1=0;
t2=Tb;
for i=1:length(dig)
fsk_signal(i,:)=awgn(fsk_signal(i,:),5); %signal to noise ratio=5
t=[t1:0.01:t2];
if strcmp(value,'FSK Modulated Signal With Noise')
plot(app.UIAxes,t,fsk_signal(i,:),'b');axis tight; %discrete signal plot
hold(ax0,'on');
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='a(t) with noise';
app.UIAxes.Title.String='FSK Modulation Signal With Noise';
app.UIAxes.XLim=[0 n];
app.UIAxes.YLim=[-5 5];
app.Label_2.Text='';
end
t1=t1+Tb;
t2=t2+Tb;
end
hold(ax0,'off');
if strcmp(value,'FSK Modulated Signal With Noise')
return
end
%% part 4: (FSK demodulation)
%rectified signal
t1=0;
t2=Tb;
for i=1:length(dig)
t=[t1:0.01:t2];
for j=1:length(t)
if fsk_signal (i,j)>0
rect(i,j)=fsk_signal(i,j);
else
rect(i,j)=0;
end
end
t1=t1+Tb;
t2=t2+Tb;
if strcmp(value,'Rectified Signal')
plot(app.UIAxes,t,rect(i,:),'b');axis tight; %discrete signal plot
hold(ax0,'on');
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='r(t)';
app.UIAxes.Title.String='Rectified Signal';
app.UIAxes.YLim=[0 5];
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
end
hold(ax0,'off');
```

```
if strcmp(value,'Rectified Signal')
return;
end
%% smoothing
t1=0;
t2=Tb;
for i=1:length(dig)
t=[t1:0.01:t2];
smoothed (i,:)=smooth(rect(i,:),500); % smoothing function
t1=t1+Tb;
t2=t2+Tb;
if strcmp(value,'Smooth Signal')
plot(app.UIAxes,t,smoothed(i,:),'b');axis tight; %discrete signal plot
hold(ax0,'on');
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='s(t)';
app.UIAxes.Title.String='Smooth Signal';
app.UIAxes.YLim=[-0.2 3];
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
end
hold(ax0,'off');
if strcmp(value,'Smooth Signal')
return;
end
%% Demodulated binary data
t1=0;
t2=Tb;
for i=1:length(dig)
t=[t1:.01:t2];
decision_val=sum( smoothed(i,:))/length(t); %average value
average(i,:)=decision_val;
t1=t1+Tb;
t2=t2+Tb;
end
max_avg= max(average);
min_avg=min(average);
avg=((max_avg)+(min_avg))/2;
for i=1:length(dig)
if average(i,:) < avg
x(i)=0;
else
x(i)=1;
end
end
if strcmp(value,'FSK Demodulated Signal')
stem(app.UIAxes,x,'b');hold on;axis tight; %discrete signal plot
app.UIAxes.XLabel.String='time';
app.UIAxes.YLabel.String='x(n)';
app.UIAxes.Title.String='FSK Demodulated Binary Data(decoded signal)';
app.UIAxes.YLim=[min(x)-0.2 max(x)+0.2];
```

```
app.UIAxes.XLim=[0 n];
app.Label_2.Text='';
end
hold(ax0,'off');
            if strcmp(value,'FSK Demodulated Signal')
                return;
            end
             %% bit error ratio
            E=x-dig;
            BER=0;
            for i=1:length(y)
                if E(i)~=0
                    BER=BER+1;     %bit error ratio
                end
            end
            BE_Ratio=BER/length(E);
                        %% part 5 (decoding)
            % binary to decimal
            b=4;
            j=1;
            samp=length(x)/b;
            for i=1:b:length(x)
                dig_row(j,:)=x(1,i:i+3);
                j=j+1;
            end
            for i=1:size(dig_row,1)
                lev(i)=bi2de(dig_row(i,:),'left-msb');
            end
        %%  reconstracted signal
        levvalue=lev*del+min(y);
        if strcmp(value,'Decoded Quantized Signal')
            stem(app.UIAxes,levvalue);axis tight;  %discrete signal plot
            app.UIAxes.XLabel.String='Index';
            app.UIAxes.YLabel.String='Displacement';
            app.UIAxes.Title.String='Decoded Quantized Signal';
            app.UIAxes.XLim=[0 n/4];
            app.UIAxes.YLim=[min(levvalue)-0.2 max(levvalue)+0.2];
            app.Label_2.Text=strcat('Bit error ratio: ',num2str(BE_Ratio));
            return;
        end
        td=linspace(0,10,length(levvalue));
        t=0:0.001:10;
        xre=interp1(td,levvalue,t,'spline');
       if strcmp(value,'Reconstructed Signal')
            plot(app.UIAxes,t,xre);axis tight;  %discrete signal plot
            app.UIAxes.XLabel.String='time';
            app.UIAxes.YLabel.String='Displacement';
            app.UIAxes.Title.String='Reconstructed Signal';
            app.UIAxes.XLim=[0 10];
            app.UIAxes.YLim=[min(levvalue)-0.2 max(levvalue)+0.2];
            app.Label_2.Text=strcat('Bit error ratio: ',num2str(BE_Ratio));
            return;
       end
```

6. App Design:

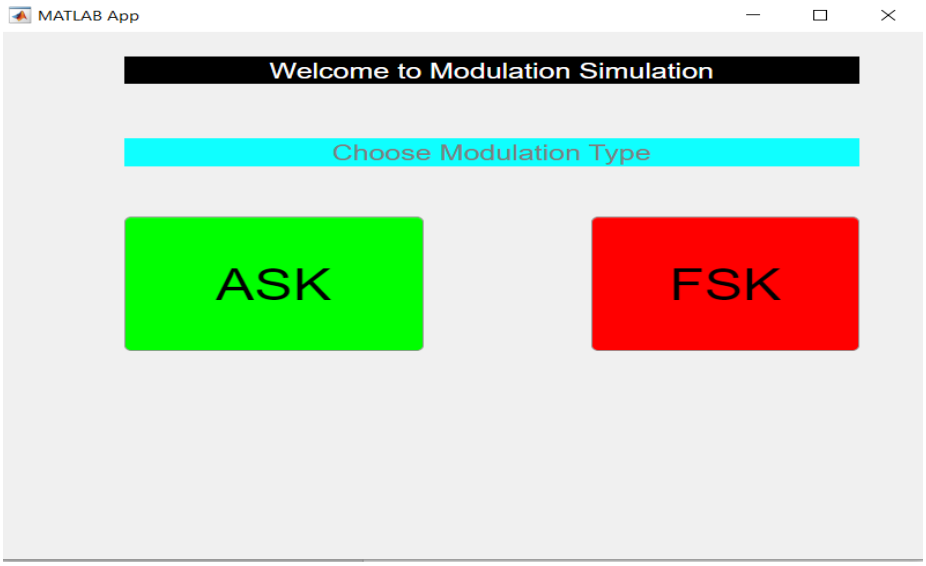
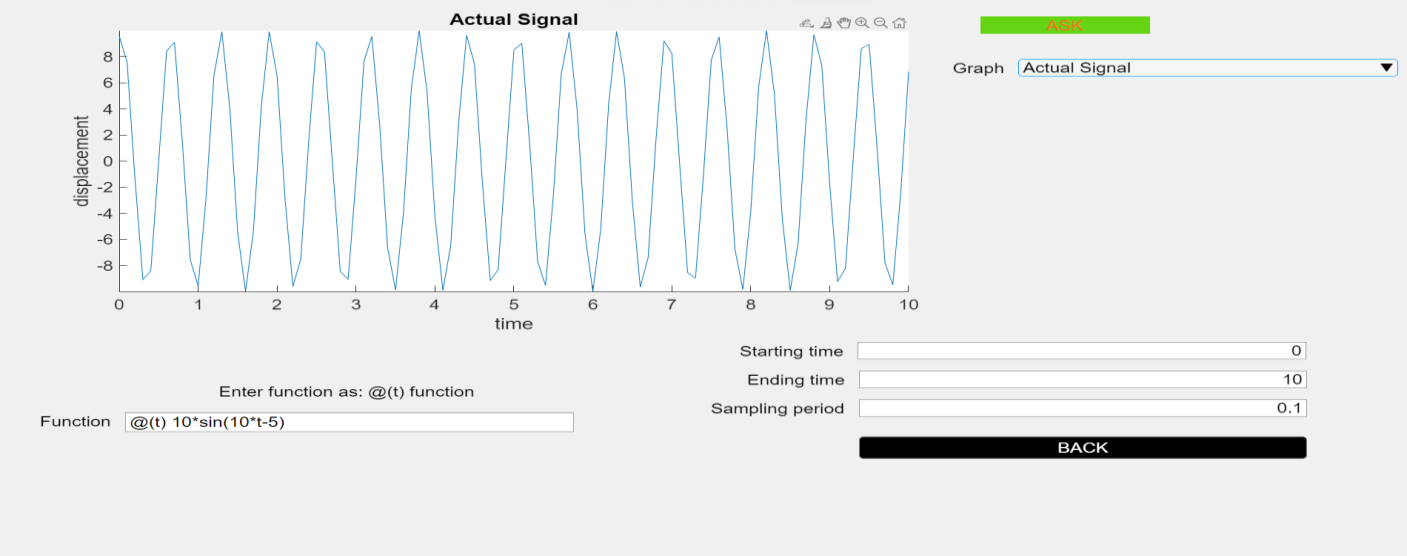
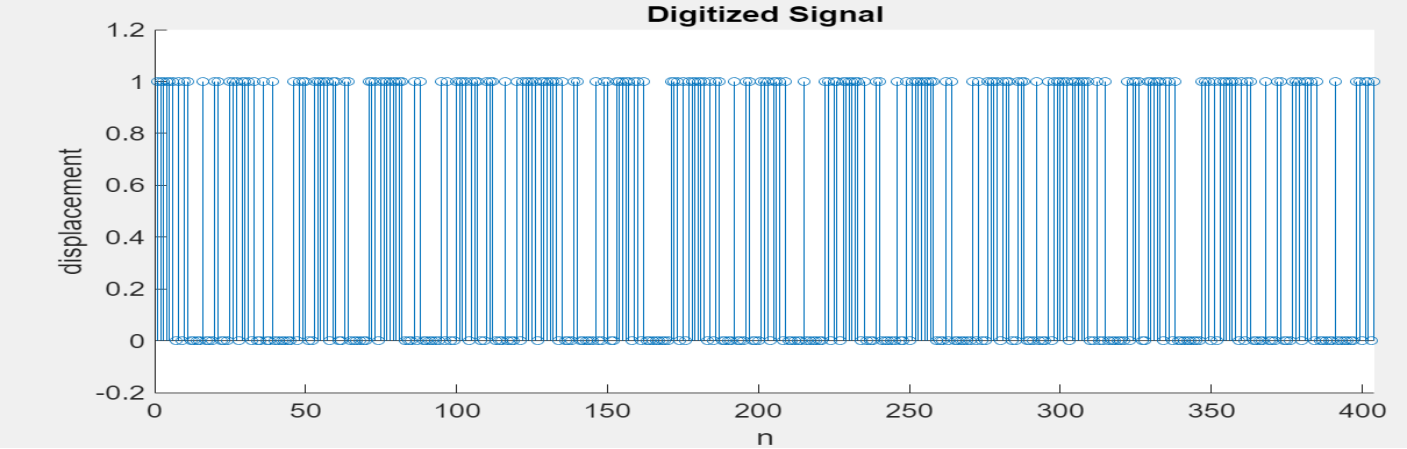


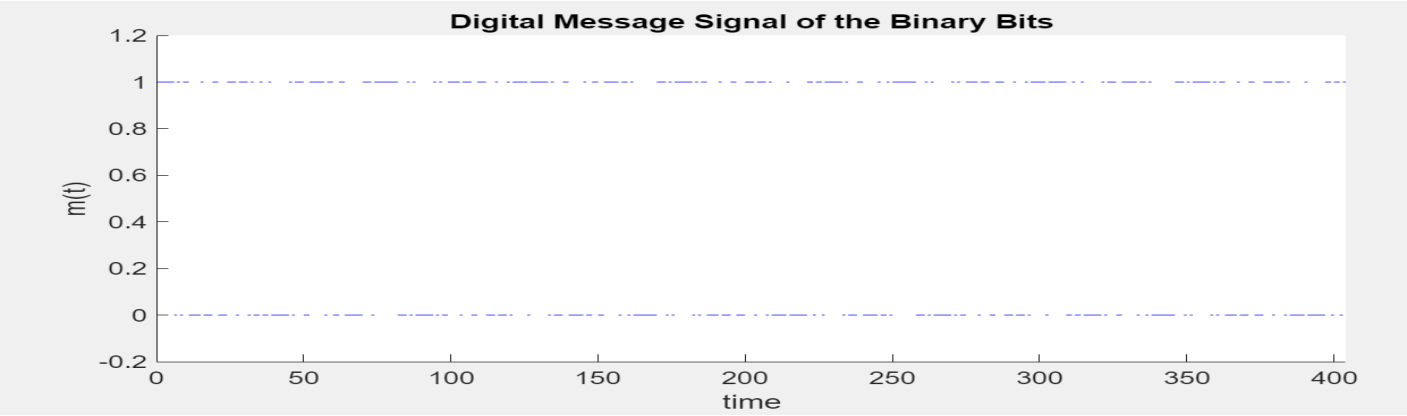
Fig 10 : Front View of the App

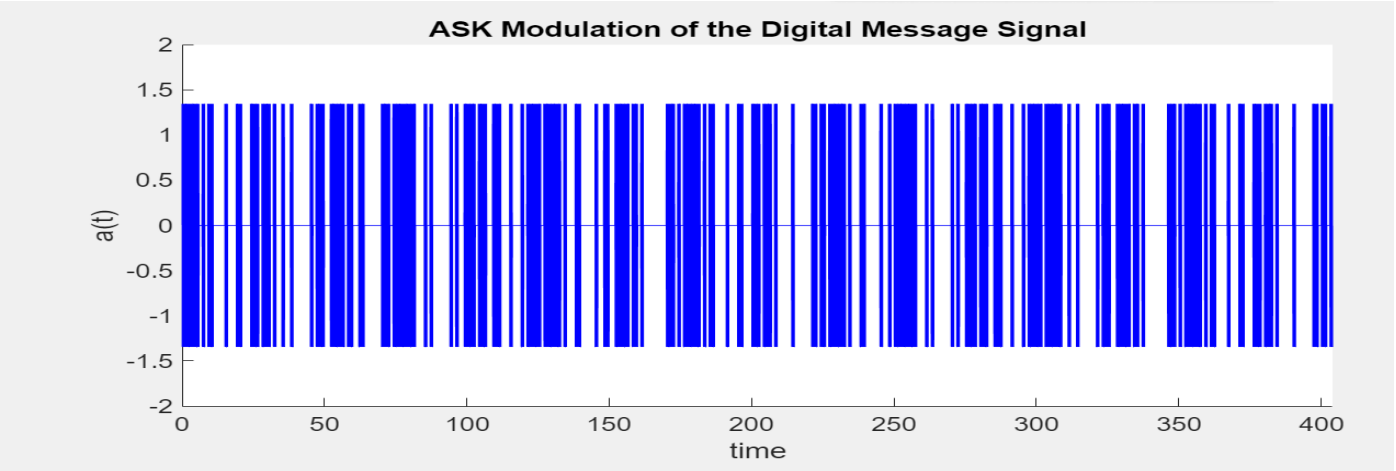
7.Test Cases:

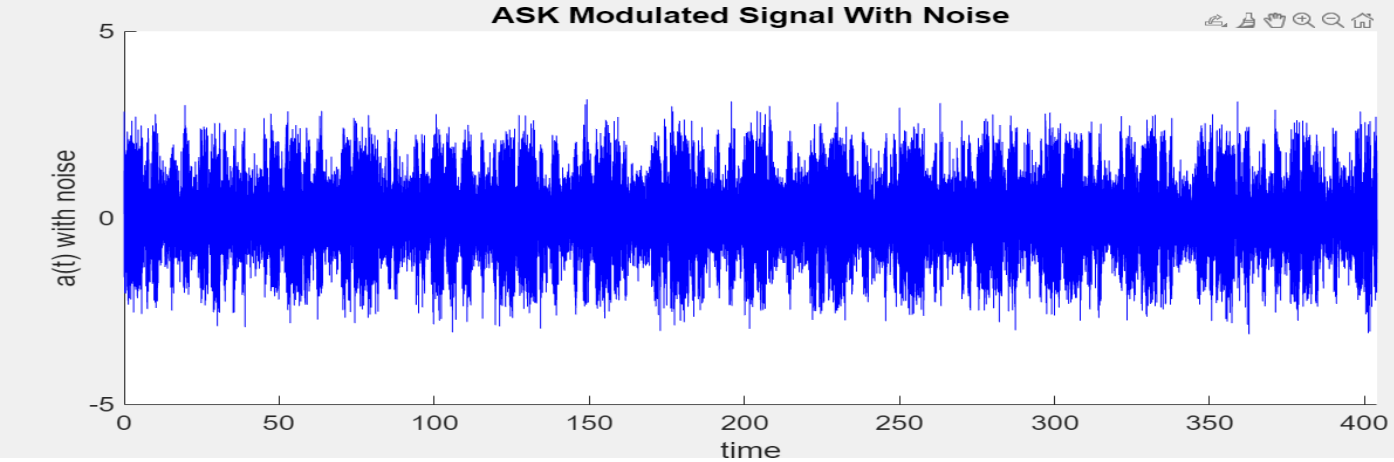
i) Function= 10*sin(10*t-5)

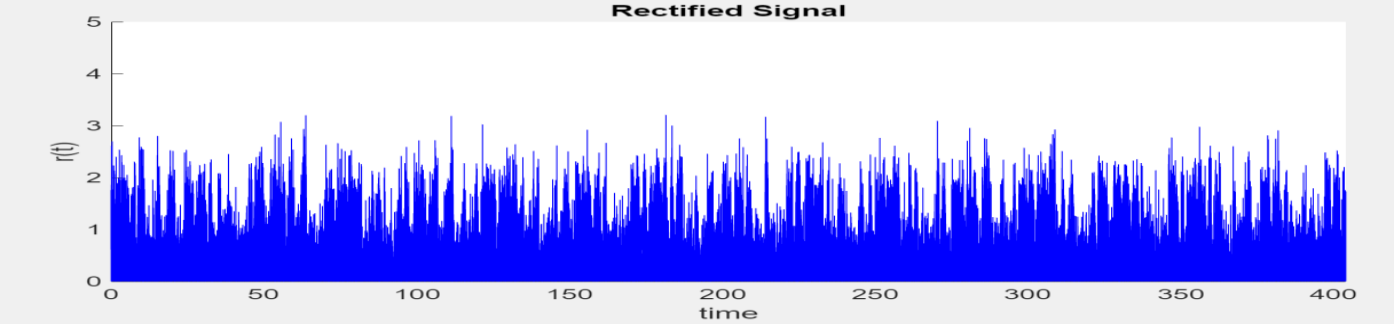


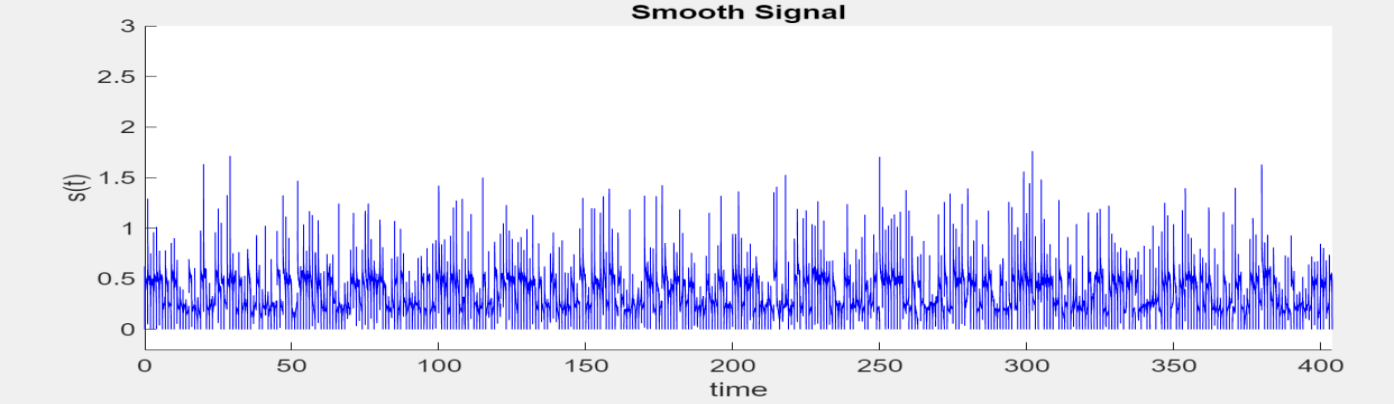

Digitized Signal
displacement
n
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
50
100
150
200
250
300
350
400

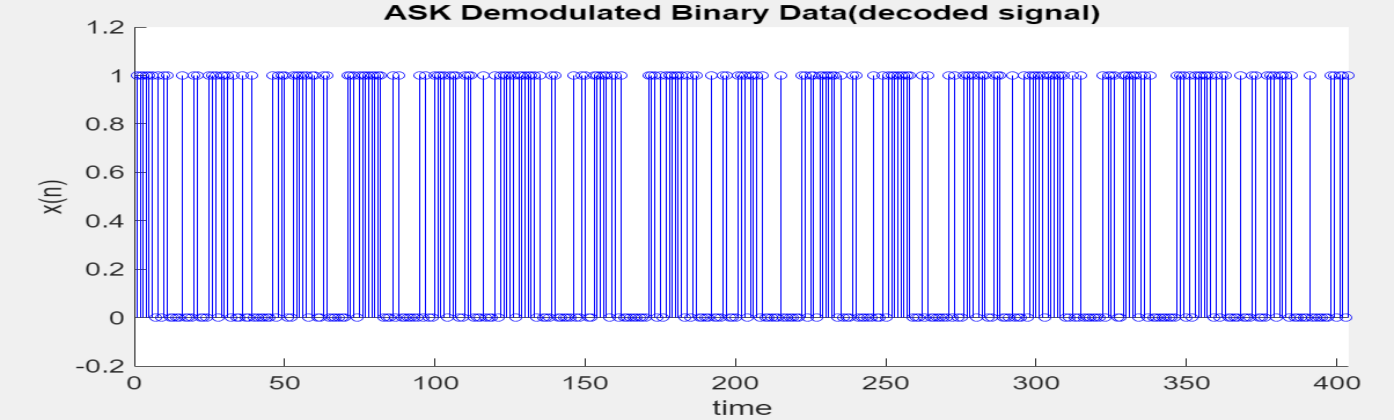

Digital Message Signal of the Binary Bits
m(t)
time
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
50
100
150
200
250
300
350
400

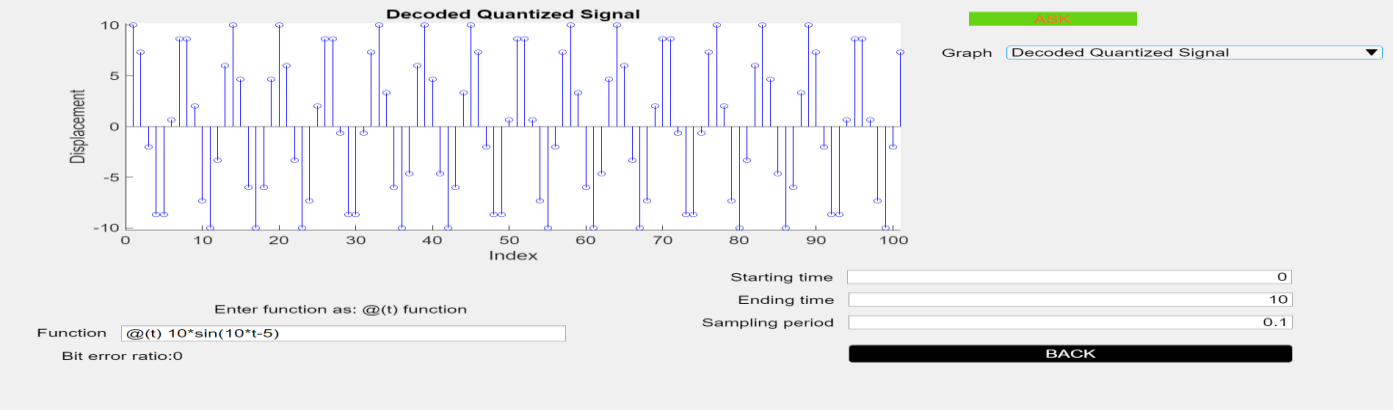

ASK Modulation of the Digital Message Signal
a(t)
time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
50
100
150
200
250
300
350
400

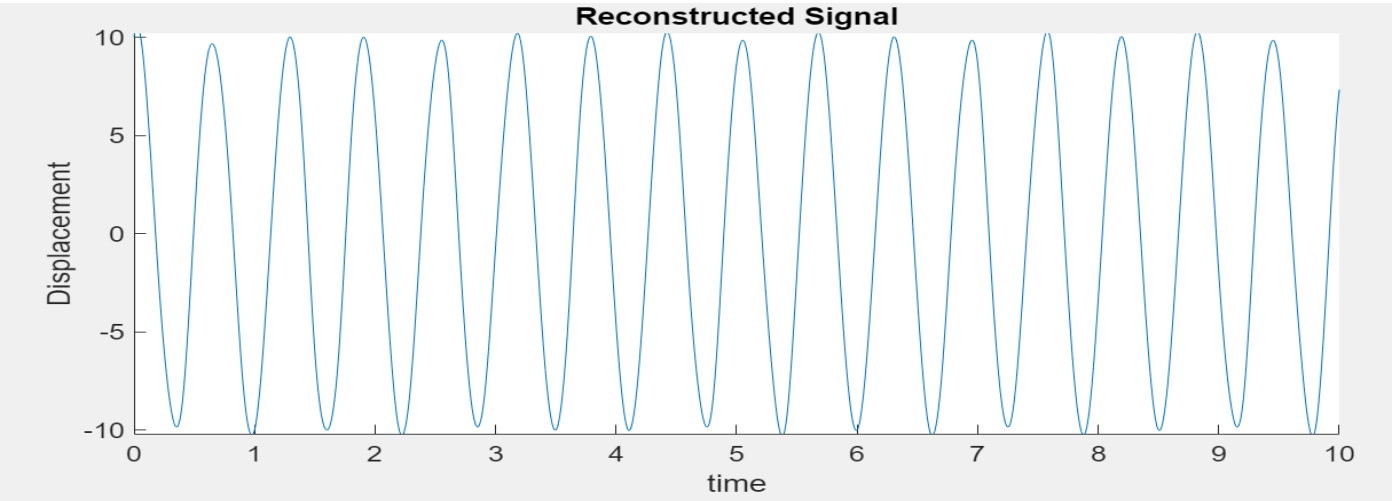

ASK Modulated Signal With Noise
a(t) with noise
time
5
0
-5
0
50
100
150
200
250
300
350
400


Rectified Signal
r(t)
time
5
4
3
2
1
0
0
50
100
150
200
250
300
350
400


Smooth Signal
s(t)
time
3
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250
300
350
400

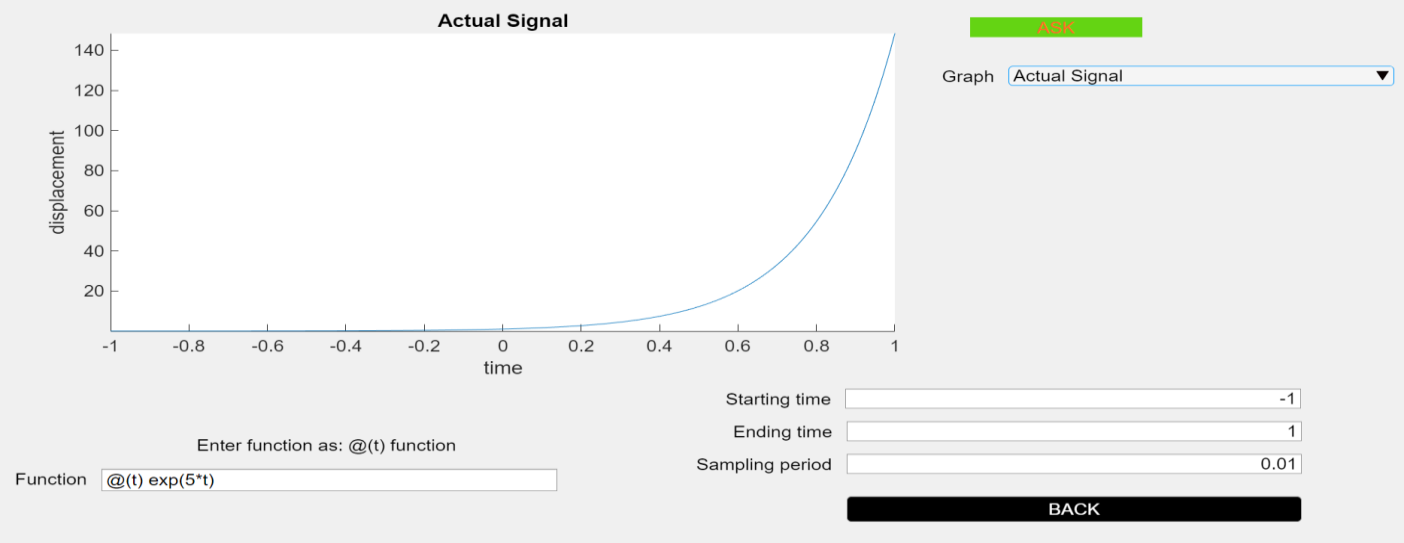

ASK Demodulated Binary Data(decoded signal)
x(n)
time
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
50
100
150
200
250
300
350
400

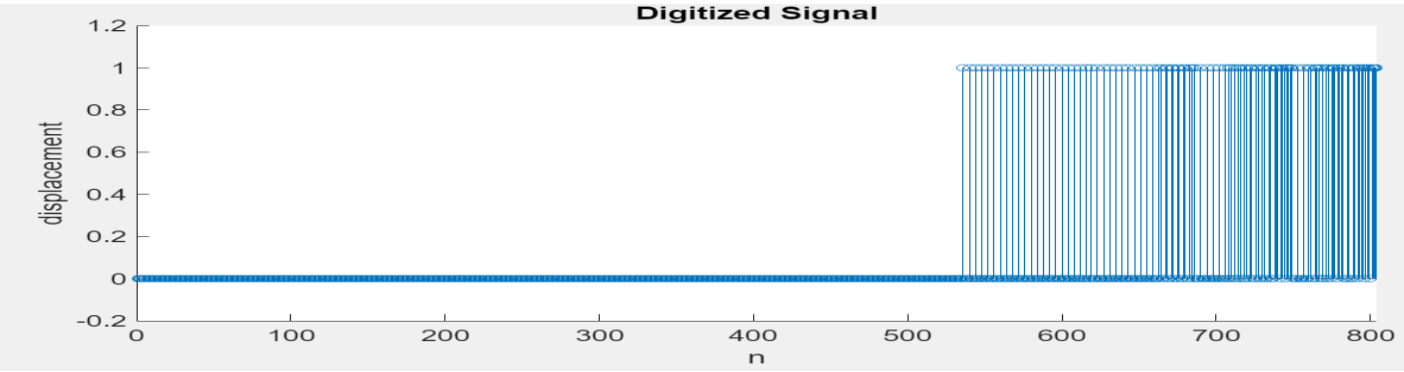


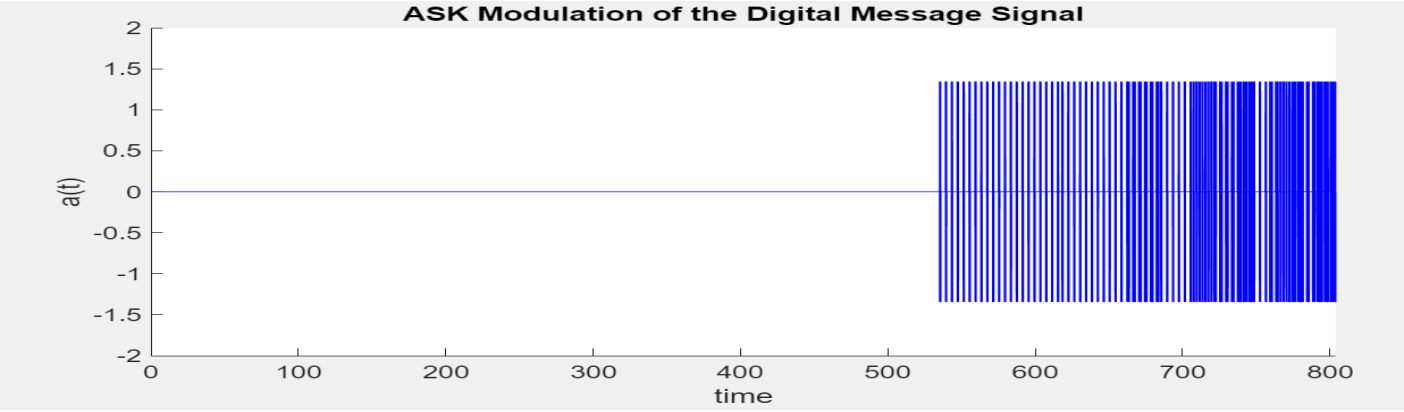


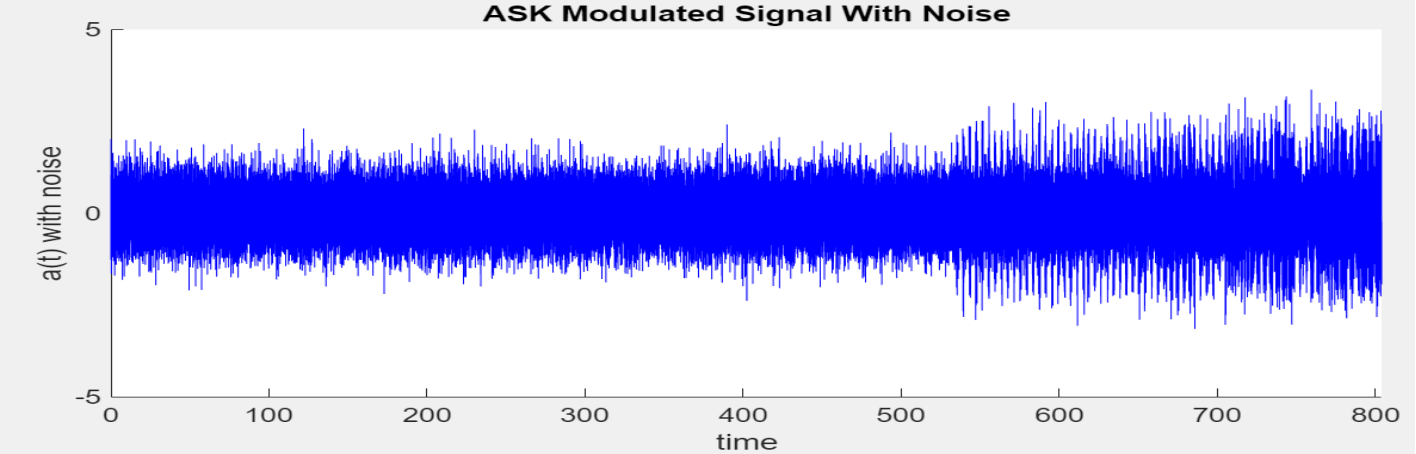
ii) If sampling frquency do not follow nyquist theorem , a warning messege will be show

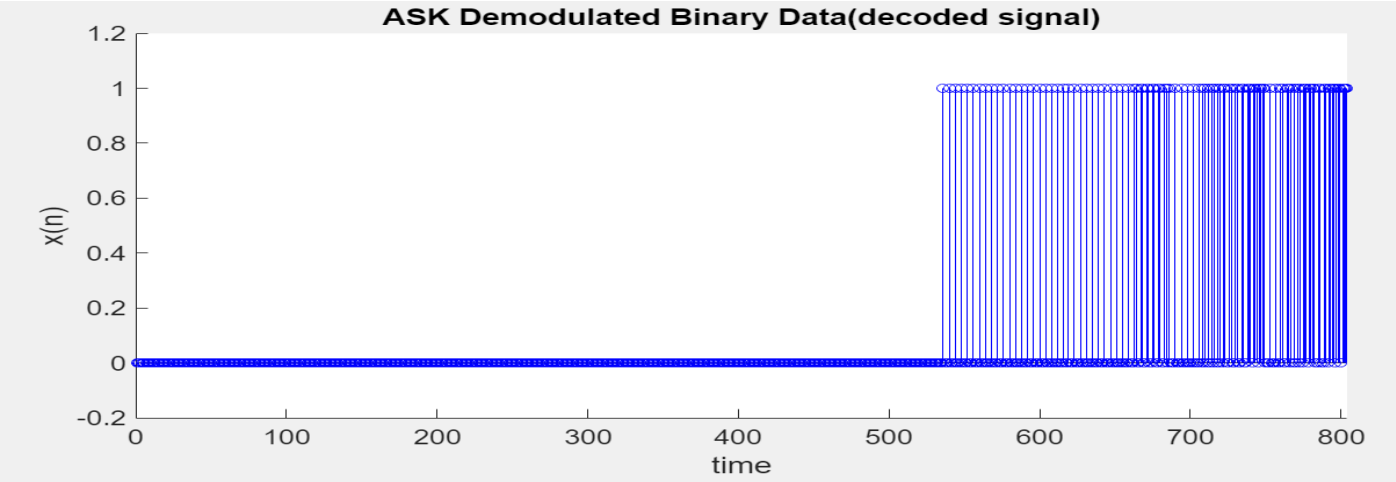
iii) For function exp(5*t)

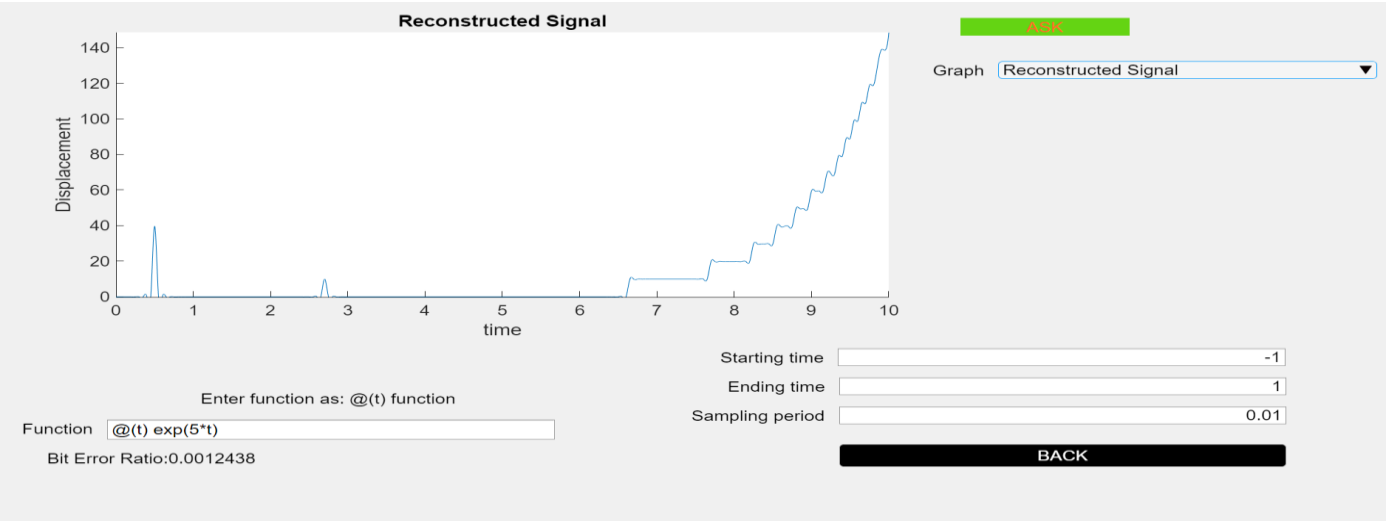

Actual Signal
displacement
time
ASK
Graph
Actual Signal
Starting time
-1
Ending time
1
Sampling period
0.01
Enter function as: @(t) function
Function
@(t) exp(5*t)
BACK


Digitized Signal
displacement
n

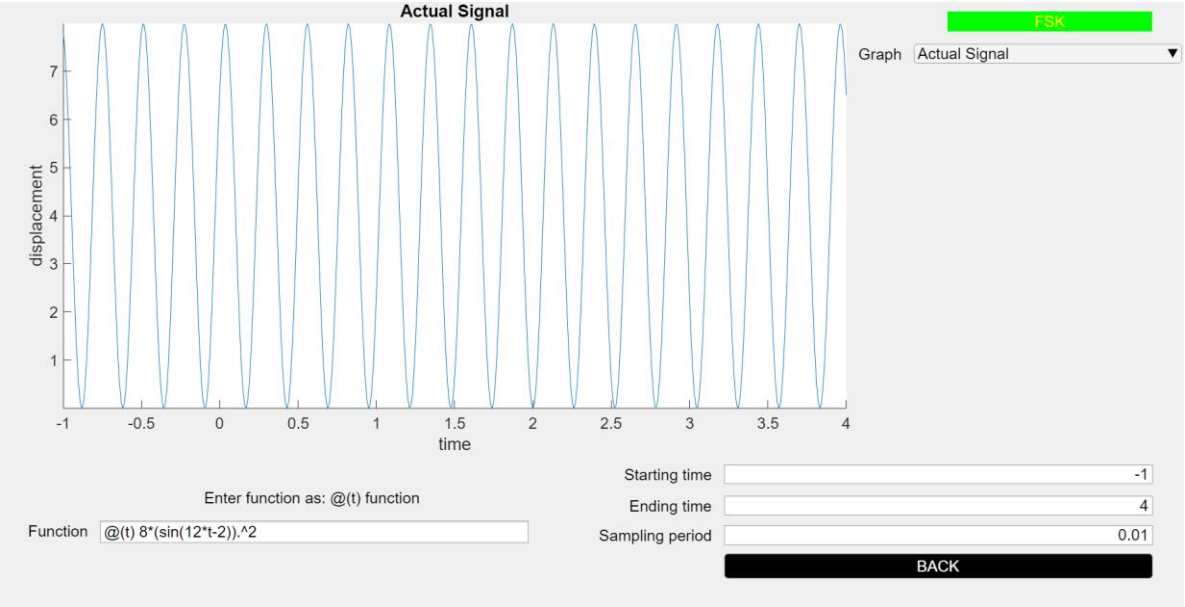

ASK Modulation of the Digital Message Signal
a(t)
time

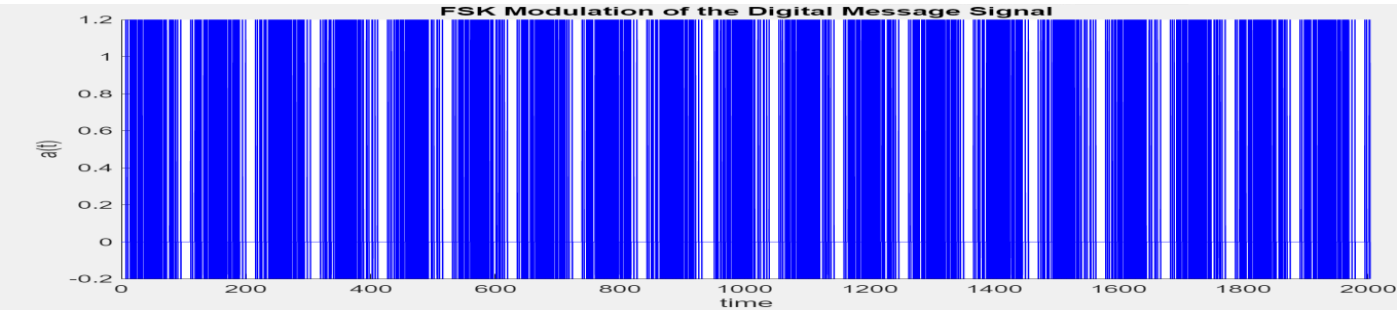


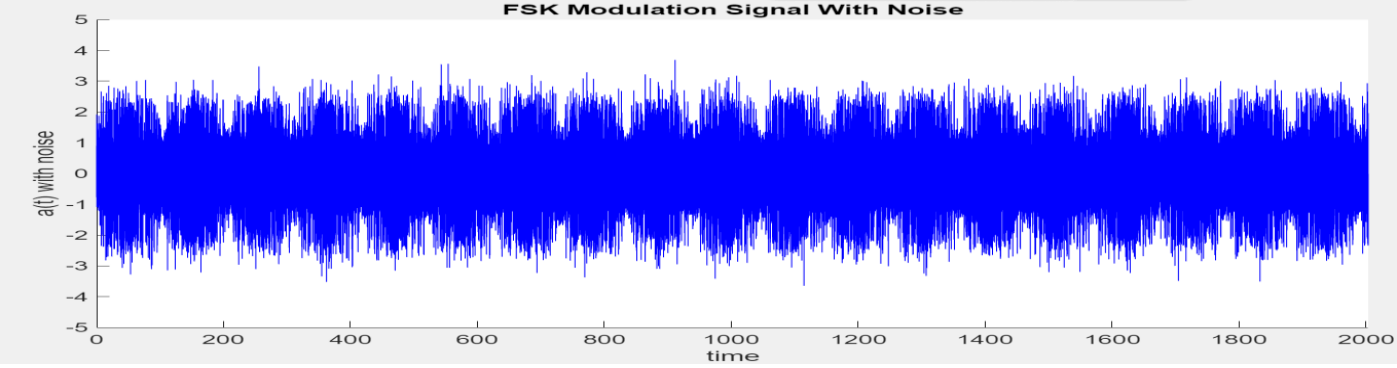


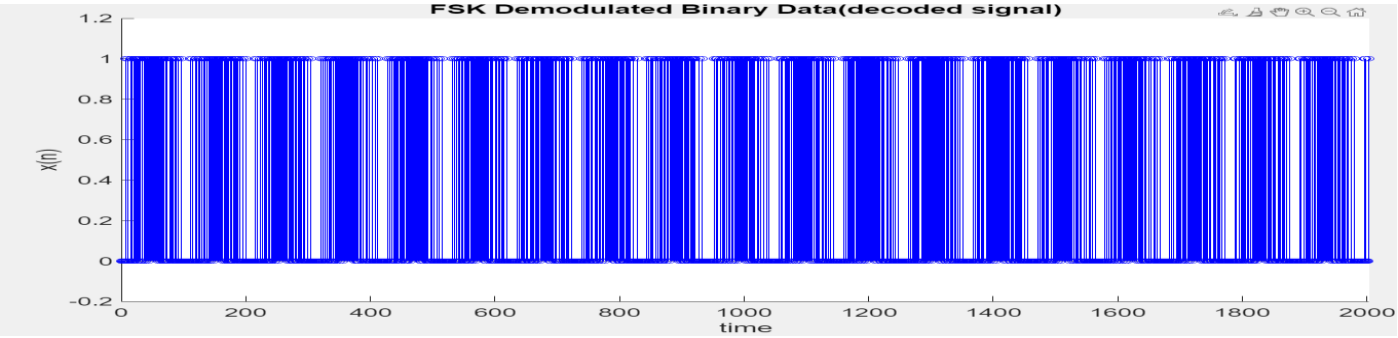


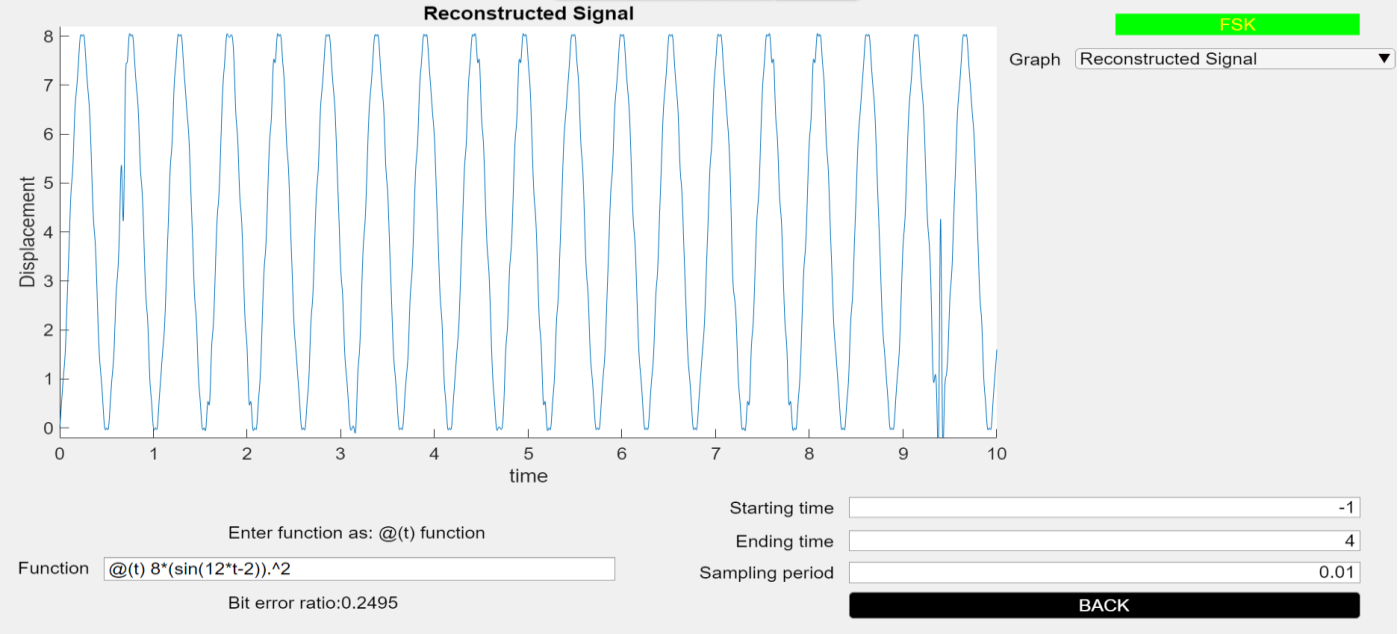
iv) 8*(sin(12*t-2)).^2


Actual Signal
displacement
time
FSK
Graph
Actual Signal
Enter function as: @(t) function
Function
@(t) 8*(sin(12*t-2)).^2
Starting time
-1
Ending time
4
Sampling period
0.01
BACK

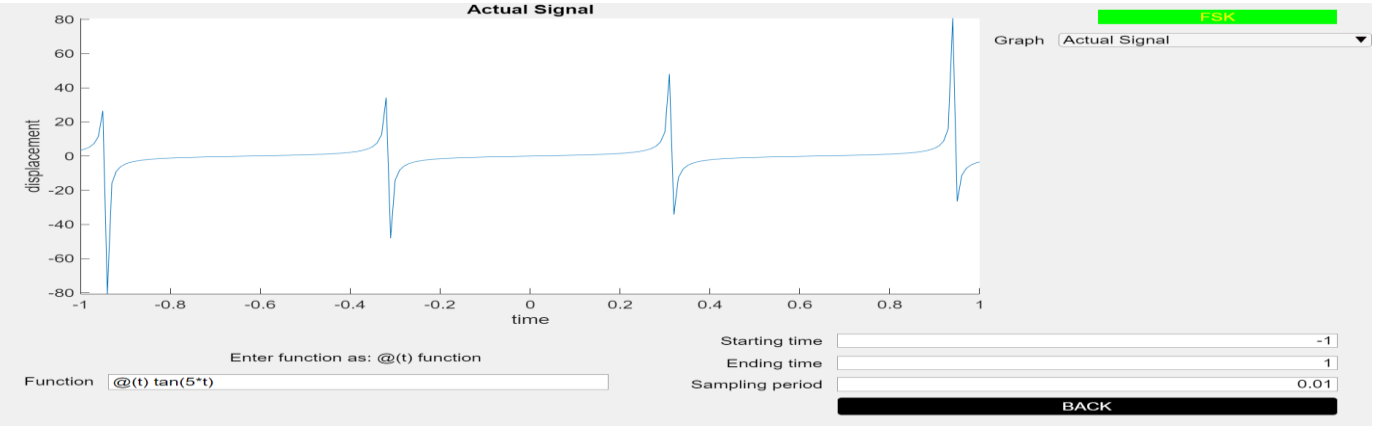

FSK Modulation of the Digital Message Signal
a(t)
time

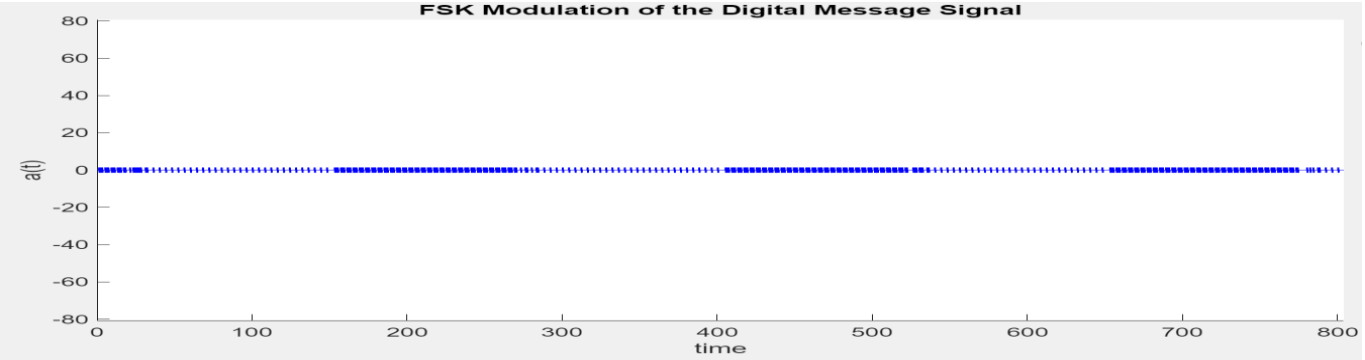

FSK Modulation Signal With Noise
a(t) with noise
time

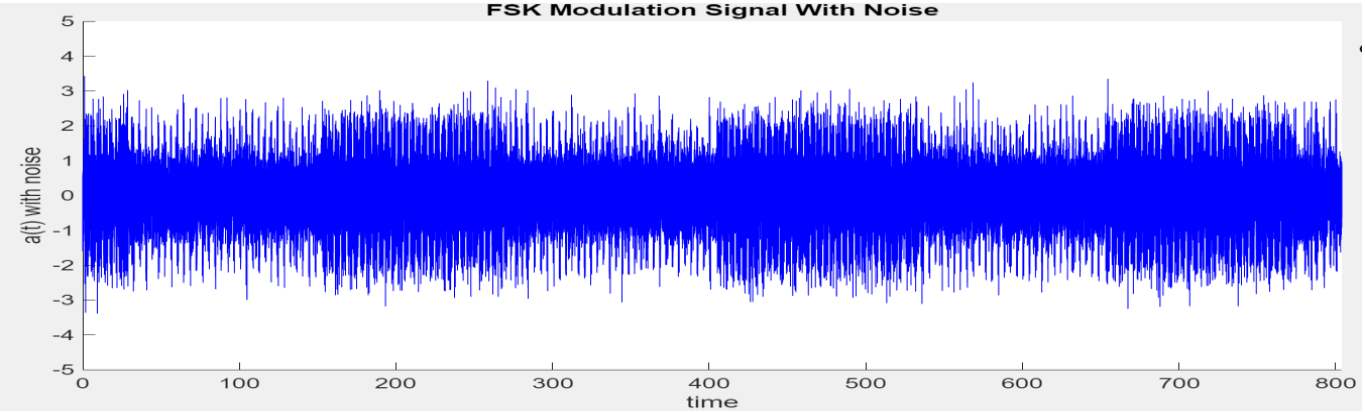

FSK Demodulated Binary Data(decoded signal)
X(n)
time

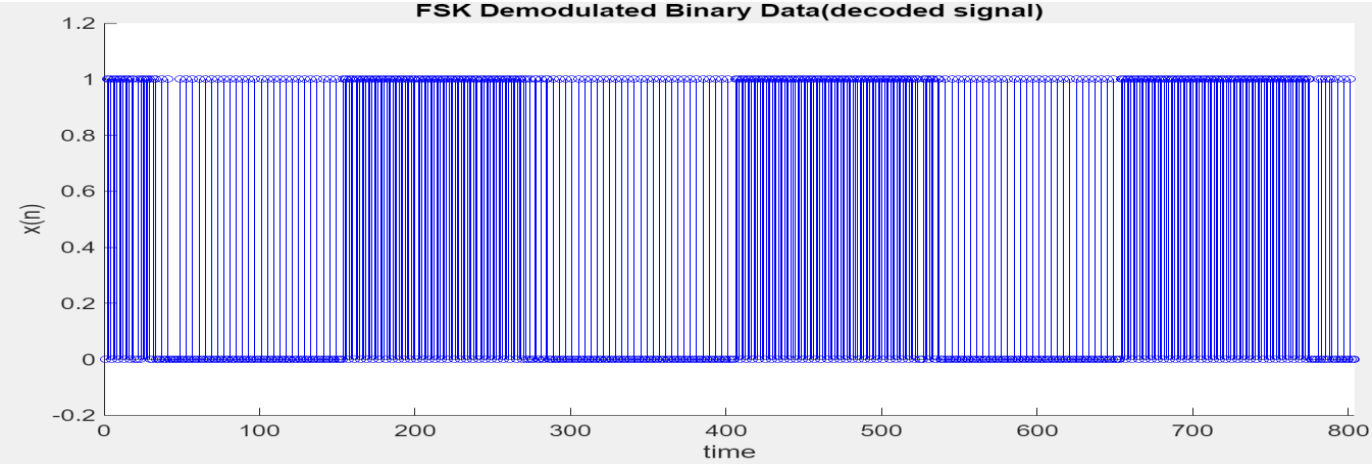


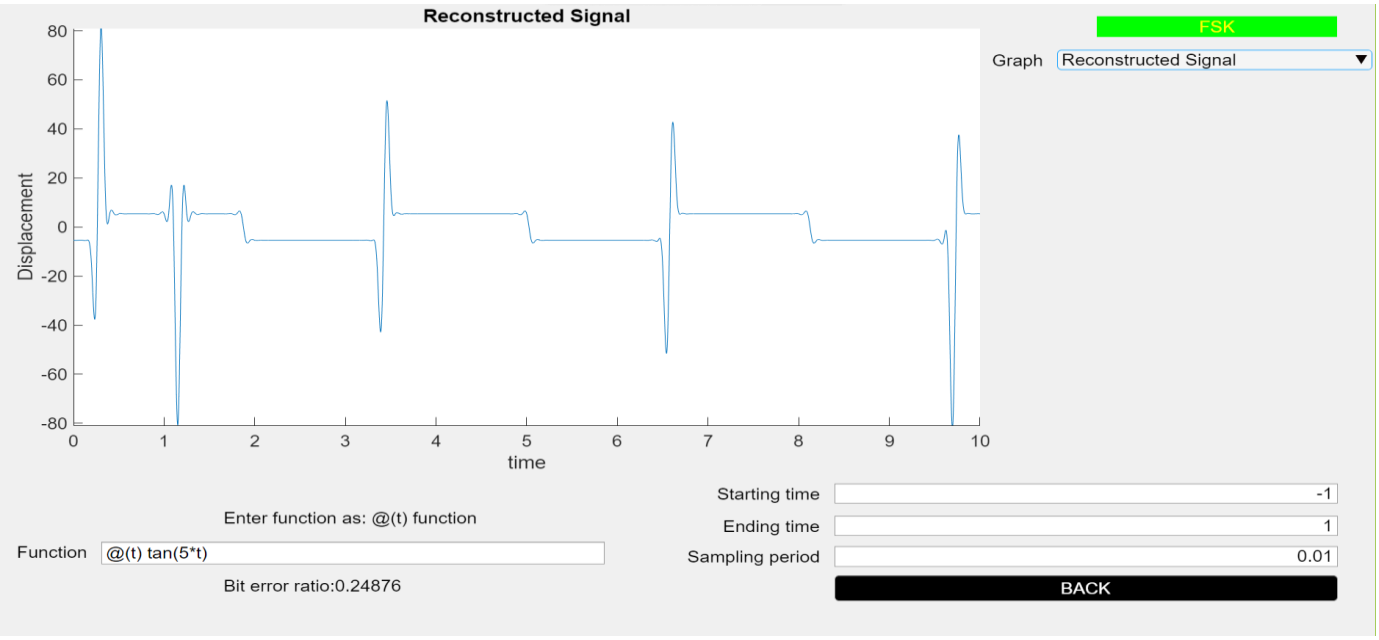
v) function tan(5*t)






FSK Modulation Signal With Noise
a(t) with noise
time


FSK Demodulated Binary Data(decoded signal)
x(n)
time


Reconstructed Signal
Displacement
time
FSK
Graph
Reconstructed Signal
Enter function as: @(t) function
Function
@(t) tan(5*t)
Bit error ratio:0.24876
Starting time
-1
Ending time
1
Sampling period
0.01
BACK

8. Application:

1) Some of the important applications of ASK has been mentioned below:

- Low-frequency RF applications
- Industrial networks devices
- Tire pressuring monitoring systems
- Wireless base station
- transmitting data or information via radio wave.
- electronic analog communication system
- transmit voice by radio.
- Amplitude Modulation(AM) process used in the analog mixer for audio control.
- It also used in Air Band Radios.

2) Some of the important applications of FSK has been mentioned below:

- Transmitting morse code
- Frequency-shift keying (FSK) is commonly used over telephone lines for caller ID (displaying callers' numbers) and remote metering applications.
- Frequency Modulation use cases include FM radio broadcasting, magnetic tape-recording systems, monitoring newborns for seizures via EEG, radar, seismic prospecting, sound synthesis, telemetry, two-way radio systems, and video-transmission systems.
- It is used on voice grade lines for data rates upto 1200 bps

9. Conclusion:

Some of the shortcomings of our project are

- Time of execution of our code is quite high. For this reason, we avoided to work with mat file.
- We used a very basic noise source. So, bit error ratio is low. But in case of high noise, the bit error ratio can be quite high.
- An annoying window pop up while running the simulation.
- We used some built in function is some process.

We are going to overcome these shortcomings by

- As this topic will be covered in our upcoming course, we will get to know about this topic thoroughly. We hope we will develop an efficient algorithm by then which will reduce the time of execution.
- We are going to study about various basic and advanced noise cancellation technique.
- The annoying window may be because of using stem function. We will study this later.
- We will work on building our function rather than using built-in function.

10. Glossary:

1)**awgn** – awgn (in , snr) adds white Gaussian noise to the vector signal. The noise is additive and the received signal is equal to the transmitted signal plus noise.

2)**bi2de-** bi2de (b) converts a binary row vector b to a decimal integer. d = bi2de (b , flg) converts a binary row vector to a decimal integer, where flg determines the position of the most significant digit.

3)**de2bi**-de2bi (d) converts a nonnegative decimal integer d to a binary row vector. If d is a vector, the output b is a matrix in which each row is the binary form of the corresponding element in d.

4)**interp1**-The interp1 command interpolates between data points. yi = interp1(x,Y,xi) returns vector yi containing elements corresponding to the elements of xi and determined by interpolation within vectors x and Y.

5)**length- L** = length(X) returns the length of the largest array dimension in X . For vectors, the length is simply the number of elements

6)**max**- If A is a vector, then max(A) returns the maximum of A

7)**min**- Min is function used in Matlab to find minimum or smallest value from an array

8)**smooth**- Smoothing is a method of reducing the noise within a data set.

9)**stem-** stem(X , Y) plots the data sequence, Y , at values specified by X. If X is a vector and Y is a matrix, then stem plots each column of Y against the set of values specified by X , such that all elements in a row of Y are plotted against the same value.

10)**sum**- A is a vector, sum(A) returns the sum of the element

11) **xcorr**- xcorr(x, y) returns the cross co-relation of two discrete-time sequences. Cross-correlation measures the similarity between a vector x and shifted (lagged) copies of a vector y as a function of the lag. If x and y have different lengths, the function appends zeros to the end of the shorter vector so it has the same length as the other.

Reference:

1.<https://www.rfwireless-world.com/Terminology/ASK-vs-FSK-vs-PSK.html >

2.<https://www.researchgate.net/figure/Digital-modulation-schemes-ASK-FSK-and-PSK_fig3_303471153 >

3.<https://www.mathworks.com/>

4.<https://www.tutorialspoint.com/principles_of_communication/principles_of_communication_digital_modulation_techniques.htm>