B.M.S COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Department of Telecommunication Engineering

(Accredited by NBA for SIX years under Tier – I format)

DIGITAL COMMUNICATION

16TE6DCDCM

REPORT ON

ADAPTIVE DELTA MODULATION WITH DUO BINARY ENCODING

(AN END TO END DIGITAL COMMUNICATION PROJECT)

Submitted by:

Abhash Kumar Jha	1BM17TE002
Advaith Shankar	1BM17TE004
Anukriti Sharma	1BM17TE009
Shubha Prasad	1BM14TE052

B.M.S COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Department of Telecommunication Engineering

(Accredited by NBA for SIX years under Tier – I format)

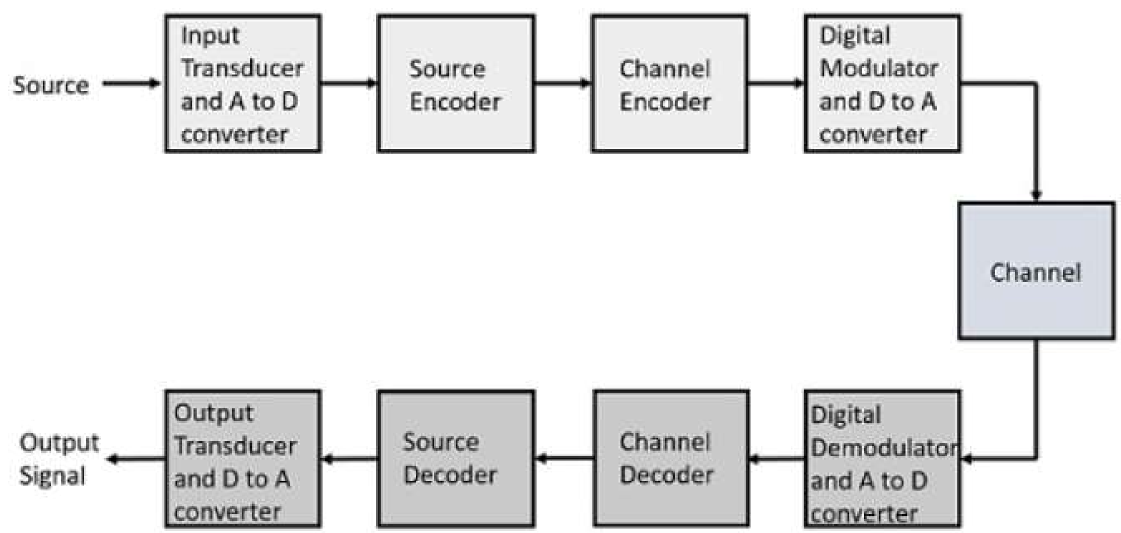
Certificate

This is to certify that the self-study entitled **"ADAPTIVE DELTA MODULATION WITH DUO BINARY ENCODING "** is submitted by ABHASH KUMAR JHA (**USN:1BM17TE-002)** **,** ADVAITH SHANKAR (**USN:1BM17TE-004**), ANUKRITI SHARMA (**USN: 1BM17TE-009)** , SHUBHA PRASAD (**USN: 1BM14TE0-052)** in partial fulfilment for the subject of DIGITAL COMMUNICATION during VI semester of Telecommunication Engineering of academic year 2020-2021

CONTENTS:

1. Introduction to Communication Model

The communication that occurs in our day-to-day life is in the form of signals. These signals, such as sound signals, generally, are analog in nature. When the communication needs to be established over a distance, then the analog signals are sent through wire, using different techniques for effective transmission.

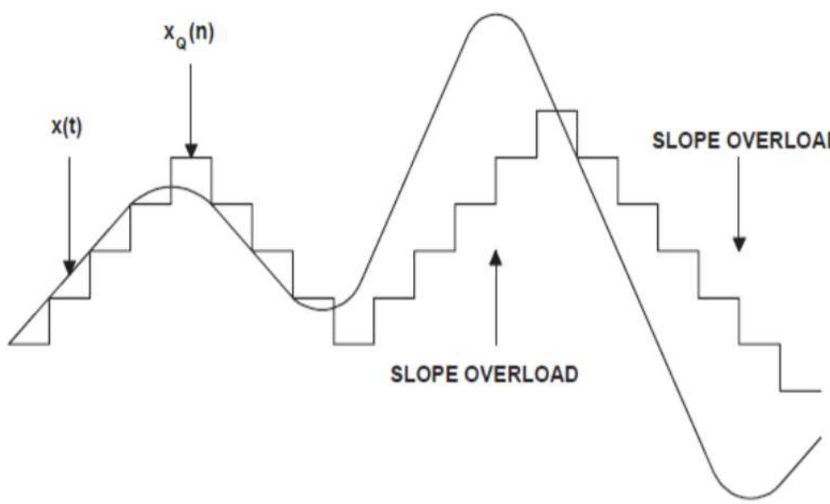


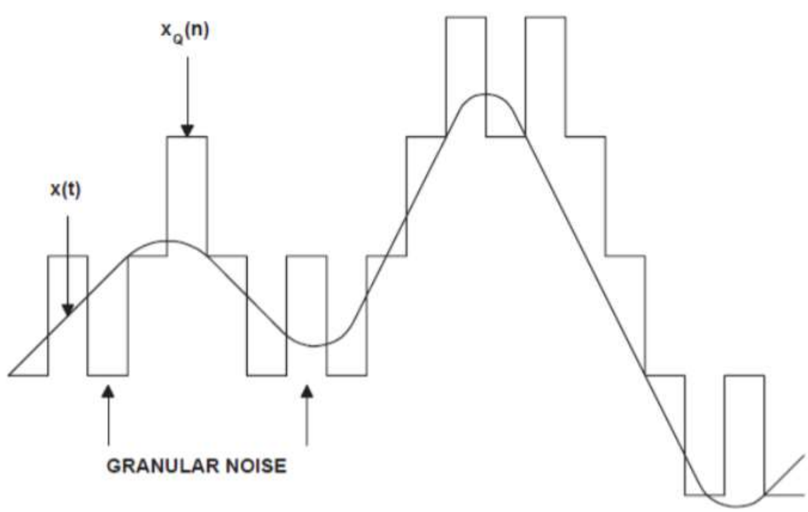
In this project, we are going to implement a Transmitter-Channel-Receiver Model in MATLAB and try to reciprocate the noise into the system for practical view of the system.

We are going to implement Adaptive Delta Modulation Technique and a correlative coding i.e. Duobinary Signaling.

2. Adaptive Delta Modulation

This Modulation is the refined form of delta modulation. This method was introduced to solve the granular noise and slope overload error caused during Delta modulation.





This Modulation method is similar to Delta modulation except that the step size is variable according to the input signal in Adaptive Delta Modulation whereas it is a fixed value in delta modulation

2.1 Working

There are several types of ADM, depending on the type of scheme used for adjusting the step size. In this project, the implementation for step-size is based upon the SONG Algorithm. In practical implementations of the system, the step size $\Delta(nTs)$ *or* $2\partial(nTs)$ is constrained to lie between minimum and maximum values.

The processor detects the pattern to see if the delta modulator is operating in the granular noise region, in which case it produces an alternating ……1010….. Pattern, or in the slope over load region in which case it produces an all-1 or all-0 pattern.

- If the ADM senses a …1010…. pattern, it decreases the step-size, and
- If it senses …..1111…. or ….0000…. pattern, it increases the step-size.

Let m(t) be the input signal and be its staircase approximation.
Let error, at the kth sampling instant *(k = 0, 1, 2, 3 . . .)* be e(k) which can be either positive, negative or zero.
Then,

$$k^{th}\ bit \ = \begin{cases} 1 & e(k) > 0 \\ 0 & e(k) < 0 \\ 0\ or\ 1 & e(k) = 0 \end{cases}$$

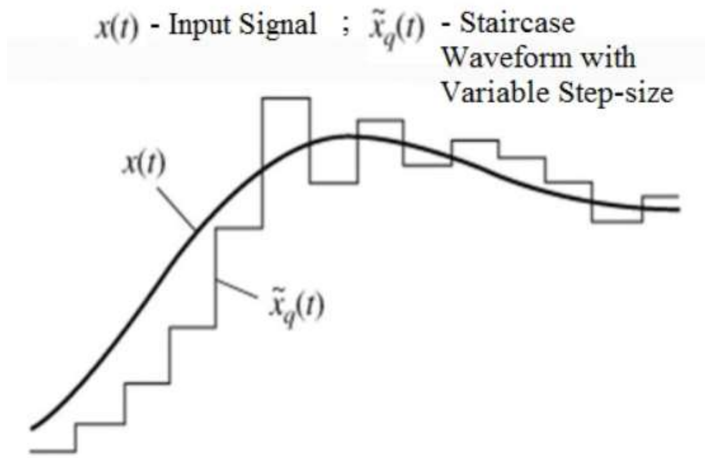
The SONG algorithm used by NASA produces the step-size ∂(k+1) which minimizes the mean-square error between m(t) and m^(t). In the implementation of the SONG system ±5 V was the maximum signal level and the minimum step-size was So = 10 mV.

Here we see that as long as e(k) is of the same sign as e(k-1) the magnitude of the new step-size ∂(k+1) will exceed the magnitude of the old step-size ∂(k) by So, the minimum step-size. However, if e(k) and e(k-1) differ in sign, the magnitude of ∂(k+1) will be less than the magnitude of ∂(k) by the amount So. The algorithm can also be written in terms of the following equation:

$$|\partial(k+1)| = \begin{cases} |\partial(k)| + \partial(0) & if\ e(k) = e(k-1) \\ |\partial(k)| - \partial(0) & if\ e(k) \neq e(k-1) \end{cases}$$

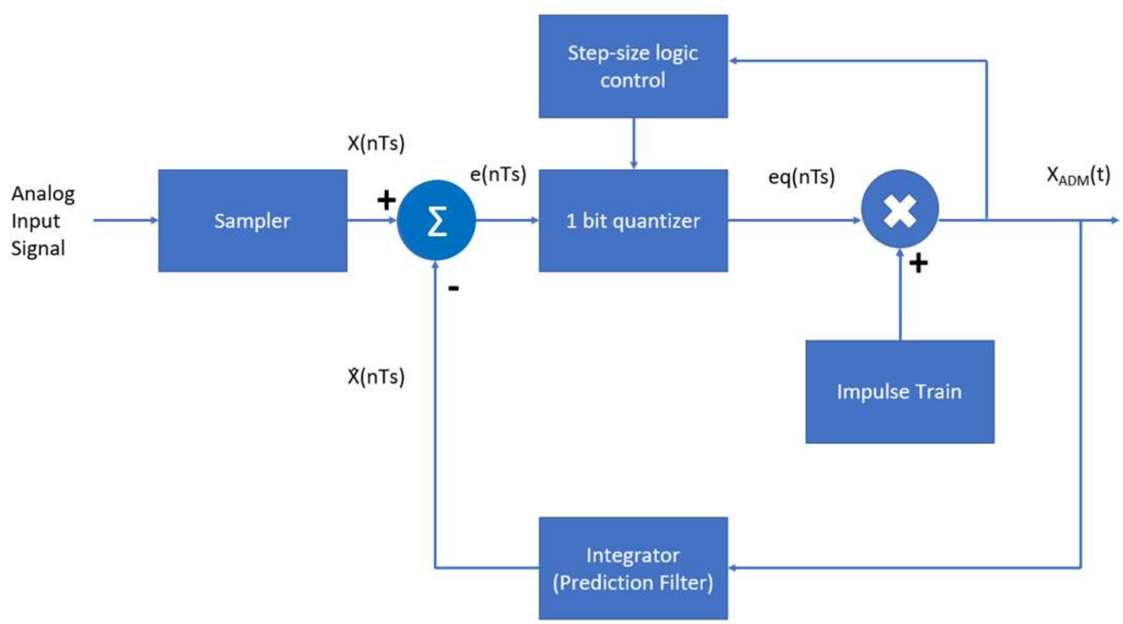
A more general form for the same equation will be,

$$\partial(k) = |\partial(k)|.sgn\big(e(k)\big) + \partial(0).sgn(e(k-1))$$

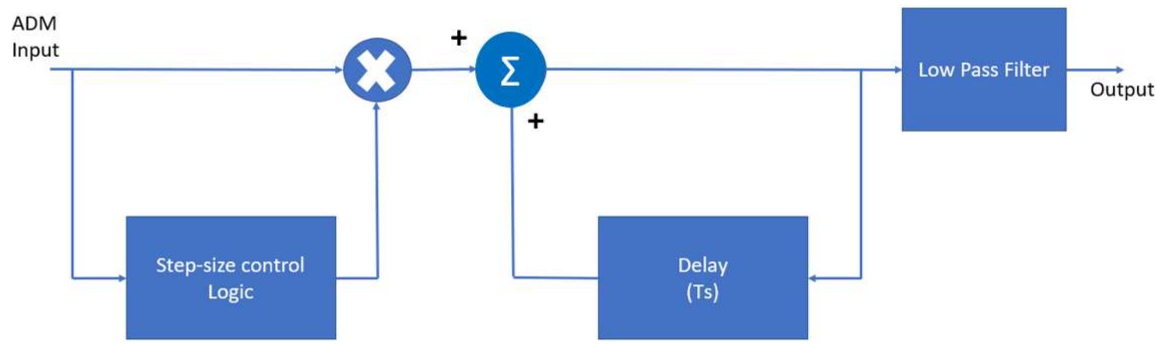


2.2 Block Diagram

TRANSMITTER:



RECIEVER:



2.3 Applications

- This modulation is used for a system which requires improved wireless voice quality as well as speed transfer of bits.
- In television signal transmission this modulation process is used.
- This modulation method is used in voice coding.
- This modulation is also used as a standard by NASA for all communications between mission control and spacecraft.
- Motorola's SECURENET line of digital radio products uses 12kbits/sec Adaptive Delta Modulation.
- To provide voice detection quality audio at deployed areas, military uses 16 to 32 kbit/sec modulation system in TRI-TAC digital telephones.
- US army forces use 16kbit/sec rates to conserve bandwidth over tactical links.
- For improved voice quality US Air Forces uses 32kbits/sec rates.
- In Bluetooth-services to encode voice signals, this modulation is used with 32bits/sec rates.

3. Duobinary Encoder/Decoder

The ISI is treated as an undesirable phenomenon that produces a degradation in system performance, but by adding ISI to the transmitted signal in a controlled manner, it is possible to achieve a bit rate of 2Bo bits per second in a channel of bandwidth Bo Hz. Such a scheme is **correlative coding** or **partial- response signalling** scheme.
One such example is **Duo binary signalling**. Duo means transmission capacity of system is doubled.

3.1 Working

Precoding:

In case of duo binary coding if error occurs in a single bit it reflects as multiple errors because the present decision depends on previous decision also. To make each decision independent we use a precoder at the receiver before performing duo binary operation.

The precoding operation performed on the input binary sequence $\{b_k\}$ converts it into another binary sequence $\{a_k\}$ given by:

$$a_k = b_k \oplus a_{k-1}$$

This $\{a_k\}$ stream is then converted into appropriate NRZ scheme. For simplicity, this project is using binary scheme i.e.

$$a_k = \begin{cases} -1, & \{a_k\} = 0 \\ 1\,, & \{a_k\} = 1 \end{cases}$$

Encoding:

The Encoding is done by adding the current bit with the previous bit i.e. the delayed element of the same bit.

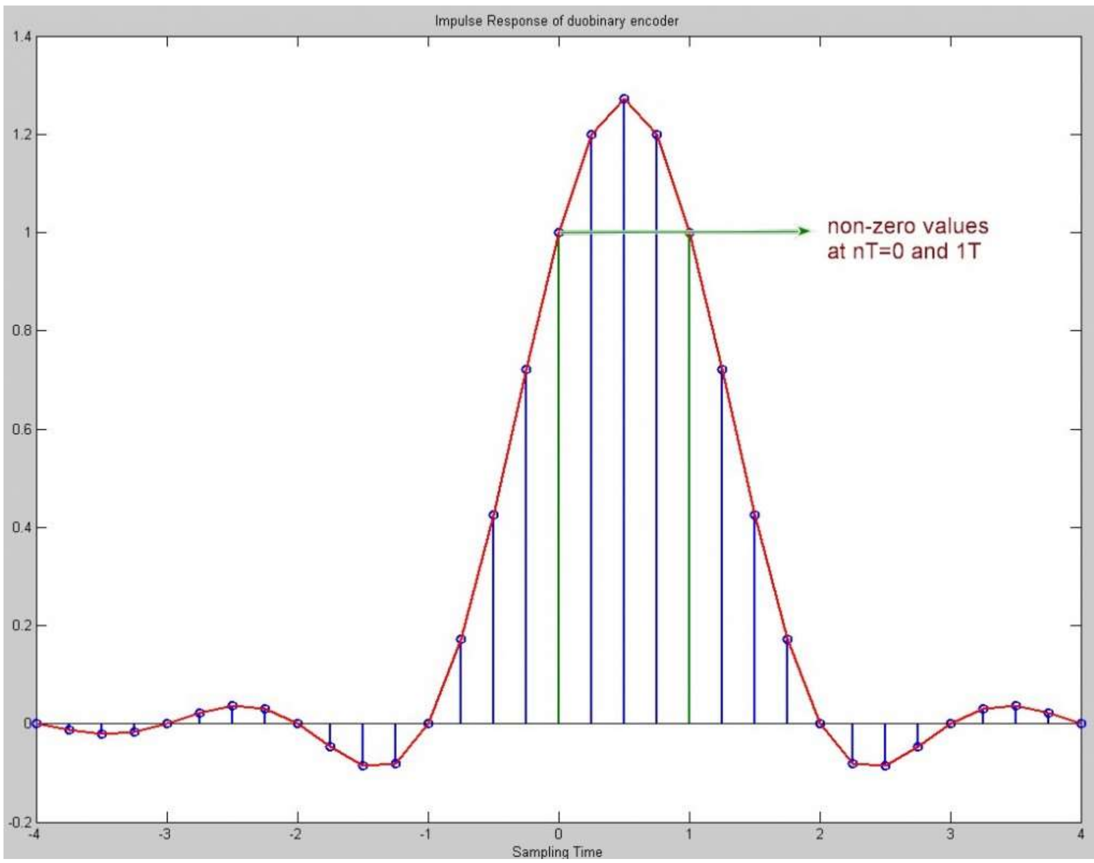
$$c_k = a_k + a_{k-1}$$

If that symbol at precoder is in polar format $\mathbf{c_k}$ takes three levels,

$$C_k = \begin{cases} \pm 2v & \text{if} \quad b_k = \text{symbol } 0 \\ 0v & \text{if} \quad b_k = \text{symbol } 1 \end{cases}$$

From the diagram, impulse response of the duobinary encoder is computed as:

$$h(t) = sinc\left(\frac{t}{T}\right) + sinc\left(\frac{t-T}{T}\right)$$

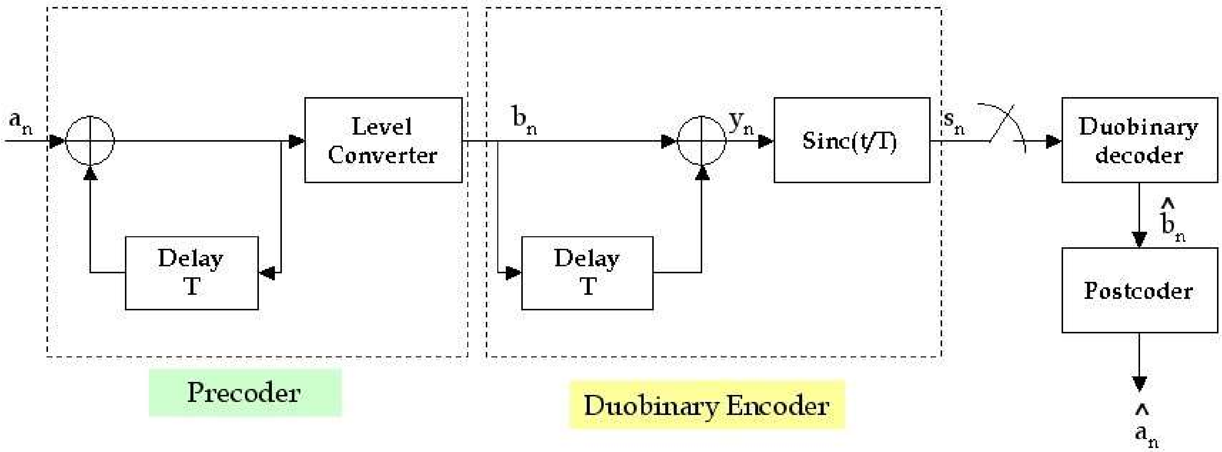


Decoding:

The decision rule for detecting the original input binary sequence $\{b_k\}$ from $\{c_k\}$ is:

$$\widehat{b_k} = \begin{cases} 1, & |c_k| < 1V \\ 0, & |c_k| \geq 1V \end{cases}$$

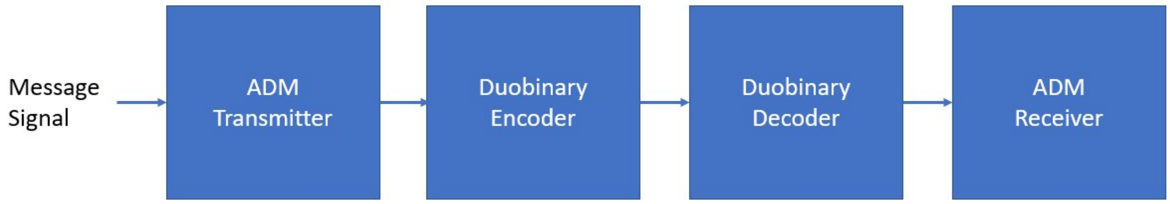
3.2 Block diagram



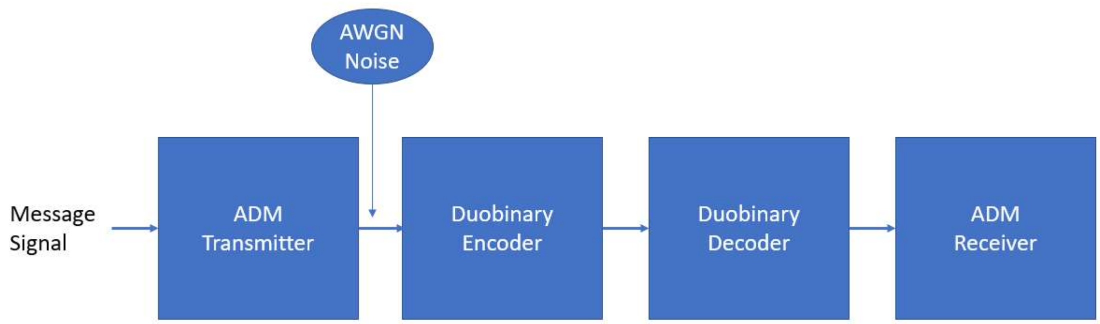
Today the duo-binary techniques are widely applied throughout the world.
While all current applications in digital communications such as data transmission, digital radio, and PCM cable transmission, and other new possibilities are being explored. This technique has been applied to fiber optics and to high density disk recording which have given excellent results

4. Meta Block Diagram

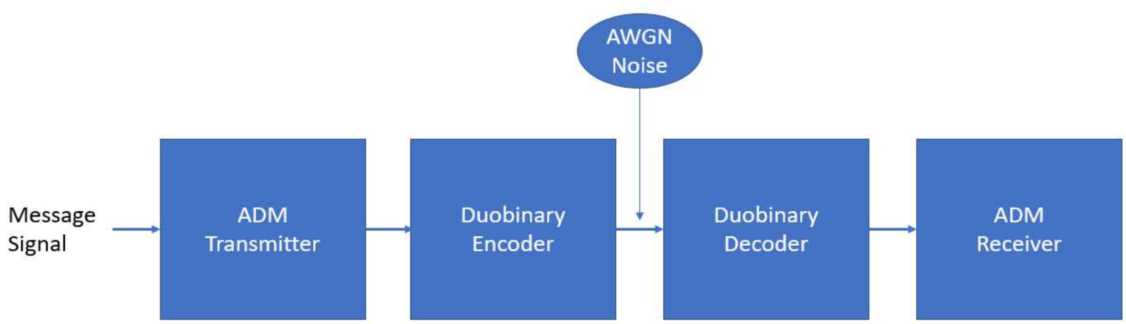
The block diagram for the system without noise will be:



The block diagram for the system with noise at the transmitter's staircase output will be:



The block diagram for the system with noise in between Duobinary Encoder and Decoder is:



5. Code

Structure of the code:

Functions:

`ad_deltamod.m`	ADM Transmitter
`ad_deltademod.m`	ADM Receiver
`Duobinary_Encoder.m`	Duobinary Encoder
`Duobinary_Decoder.m`	Duobinary Decoder

Runner files:

- `Communication_system.m`
- `Communication_system_duobinary_error.m`

ad_deltamod.m

```
function [c,xq,zero_flag_array] = ad_deltamod(x,delta_min)

%This is a Adaptive delta Modulator


%{
x -> The signal input,a vector
delta_min -> min step size
zero_flag_array -> zero indicator to avoid zero error
c -> encoded digital output
%}


    N = length(x);
    delta_array = zeros(1,N); %variable delta
    e = zeros(1,N);           %error
    eq = zeros(1,N);          %quantized error
    xq = zeros(1,N);

    %zero_flag
    zero_flag_array = zeros(1,N);


    for i = 1:N
        if x(i) == 0
            zero_flag_array(i) = 1;
        end
    end


    for i = 1:length(x)
            if(i==1)
                e(i) = x(i);

                if x(i) == 0
                    delta_array(i) = delta_min;
                else
                    delta_array(i) = delta_min*sign(e(i));
                end

                eq(i) = delta_array(i);
                xq(i) = eq(i);
            else
                e(i) = x(i) - xq(i-1);
                delta_array(i) = (abs(delta_array(i-1))*sign(e(i))) +
(delta_array(1)*sign(e(i-1)));
                %delta_array(i) = delta_min*sign(e(i));
                eq(i) = delta_array(i);
                xq(i) = eq(i) + xq(i-1);
            end

    end

    c = ones(1,N);
    for i = 1:N
        if e(i)>0
            %only positive vals
            c(i) = 1;
        else
            %-ve + 0
            c(i) = 0;
```

```
        end
    end

end
```

ad_deltademod.m

```
function [rec_xq,signal] = ad_deltademod(c,zero_flag_array,delta_min)

%This is a adaptive delta demodulator

L = length(c);

signal = zeros(1,L);
rec_delta_array = zeros(1,L);
sign_array = zeros(1,L);
rec_xq = zeros(1,L);

%first we need to generate our error/sign array

for i = 1:L
    if c(i) == 0
        sign_array(i) = -1;
    else
        sign_array(i) = 1;
    end

    if zero_flag_array(i) == 1
        sign_array(i) = 0;
    end

end

for i = 1:L
        if(i==1)
            if(sign_array(i) == 0)
                rec_delta_array(i) = delta_min;
            else
                rec_delta_array(i) = delta_min*sign_array(i);
            end

            rec_xq(i) = rec_delta_array(i);
            signal(i) = rec_xq(i);
        else
            rec_delta_array(i) = (abs(rec_delta_array(i-1))*sign_array(i)) + 
(rec_delta_array(1)*sign_array(i-1));
            rec_xq(i) = rec_delta_array(i) + rec_xq(i-1);
            signal(i) = signal(i-1) + rec_xq(i);
        end
end
end
```

Duobinary_Encoder.m

```
function [c] = Duobinary_Encoder(b)

%This is a precoded duobinary encoder
%b is the input binary sequence : precoded input
% a_volts is basically the NRZ encoder output
%c is the duobinary coded output

%variables
a = zeros(1,length(b));
```

```
a_volts = zeros(1,length(b));
a(1) = xor(1,b(1));


%NRZ Encoder for converting bits to voltage levels
if(a(1) == 1)
    a_volts(1) = 1;
else
    a_volts(1) = -1;
end

for k = 2:length(b)
    a(k) = xor(a(k-1),b(k));
    if(a(k) == 1)
        a_volts(k) = 1;
    else
        a_volts(k) = -1;
    end
end

disp('Precoded output in binary:');
disp(a);
disp('..................................................................');
disp('Precoded output in voltage:');
disp(a_volts);
disp('..................................................................');

%Encoder
c = zeros(1,length(b));
c(1) = 1 + a_volts(1);

for k = 2:length(a)
    c(k) = a_volts(k-1) + a_volts(k);
end
```

Duobinary_Decoder.m

```
function [b_out] = Duobinary_Decoder(c)

%This is a precoded binary decoder
%c is the duobinary coded output
%b_out is the output of suobinary decoder


% decoder decision
b_out = zeros(1,length(c));
for k =1:length(c)
  if(abs(c(k)) > 1)
    b_out(k) =  0;
  else
    b_out(k) = 1;
  end
end
```

Communication_system.m

```
clear all;
close all;
clc;

%% inputs
Am = 5;      %amplitude             %input('Enter the value for amplitude');
fm = 1;      %frequency             %input('Enter the value for frequency');
fs = 30*fm;  %sampling freq         %input('What is the sampling frequency?');

%t = 0:1/fs:10;
dt = 1/fs;                      % seconds per sample
StopTime = 1;                   % seconds
t = (0:dt:StopTime-dt)';

%input signal
x = Am*sin(2*pi*fm*t);

%{
N = size(t,1);
y = fftshift(fft(x));
dF = fs/N;                        % hertz
f = -fs/2:dF:fs/2-dF;             % hertz
%%Plot the spectrum:
figure;
plot(f,abs(y)/N);
xlabel('Frequency (in hertz)');
title('Magnitude Response');
%}

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

%% Without Error
disp('Adaptive Delta Modulation without noise');
disp(' ');


[digital_code,xq,zero_track] = ad_deltamod(x,delta);
%Transmittor
disp('The digital code is:');
disp(digital_code);
disp(' ');

coded = Duobinary_Encoder(digital_code);
%duobinary_encoder
disp(' ');
disp('Encoded digital code is:');
disp(coded);
disp(' ');

decoded_digital_code = Duobinary_Decoder(coded);
%duobinary_decoder
disp('The output of duobinary decoder is');
disp(decoded_digital_code);
disp(' ');

[rec_staircase,my_signal] = ad_deltademod(decoded_digital_code,zero_track,delta);
%Reciever

%y = lowpass(my_signal,2*fm,fs);
%lowpass filter output
b = fir1(100,10*fm/fs);
y = conv2(my_signal,b,'same');

%% With Error
disp('Adaptive Delta Modulation with noise');
```

```
disp(' ');

%The Transmittor part is same so, we will take same variables as above

%Adding noise to the staircase signal -> Channel

x_error = awgn(xq,-10);

digital_code_error = ones(1,length(xq));

digital_code_error(1) = (x_error(1)>0);
for i = 2:length(xq)
    if x(i)-x_error(i-1)>0
        %only positive vals
        digital_code_error(i) = 1;
    else
        %-ve + 0
        digital_code_error(i) = 0;
    end
end

disp('The noisy digital code is:');
disp(digital_code_error);
disp('..................................................................');

coded_error = Duobinary_Encoder(digital_code_error);
%duobinary_encoder
disp('Encoded digital data is:') ;
disp(coded_error);
disp('..................................................................');

decoded_digital_code_error = Duobinary_Decoder(coded_error);
%duobinary_decoder
disp('The output of duobinary decoder is');
disp(decoded_digital_code_error);
disp('..................................................................');

[rec_staircase_error,my_signal_error] =
ad_deltademod(decoded_digital_code_error,zero_track,delta);  %Reciever

%y_error = lowpass(my_signal_error,2*fm,fs);
%low_pass

y_error = conv2(my_signal_error,b,'same');
%% Without Error Plots

%orignal signal,staircase signal, recieved staircase signal,rec_low_pass

figure('Name','Adaptive Delta Modulation without noise','NumberTitle','off');
plot(t,x,'DisplayName','Message signal');
title('Adaptive Delta Modulation');
xlabel('Time (in sec)');
ylabel('Amplitude (in volts)');
hold 'on';
stairs(t,xq,'DisplayName','Staircase signal');
stairs(t,rec_staircase,'DisplayName','Recieved Staircase signal (Overlapping with
staircase signal of reciever)');
plot(t,y,'DisplayName','Recieved signal');
hold 'off';
legend


%Binary signals

bp = .00001; %bit period
br = 1/bp;   %bit rate

digital_code_bit = [];
for n=1:1:length(digital_code)
```

```
    if digital_code(n)==1
       se=ones(1,100);
    else
        se=-1*ones(1,100);
    end
     digital_code_bit=[digital_code_bit,se];
end

coded_bit = [];
for n=1:1:length(coded)
    if coded(n)== 2
       c_se= 2*ones(1,100);
    elseif coded(n) == -2
        c_se= -2*ones(1,100);
    else
        c_se = zeros(1,100);
    end
     coded_bit=[coded_bit c_se];
end

decoded_digital_code_bit = [];
for n=1:1:length(decoded_digital_code)
    if decoded_digital_code(n)==1
       se=ones(1,100);
    else
        se=-1*ones(1,100);
    end
     decoded_digital_code_bit=[decoded_digital_code_bit se];
end


t1= bp/100:bp/100:100*length(x)*(bp/100);
figure('Name','Bit Transmission without error','NumberTitle','off')

subplot(3,1,1);
plot(t1,digital_code_bit,'lineWidth',2.5);
grid on;
axis([ 0 bp*length(coded) -1.5 1.5]);
ylabel('Amplitude(volt)');
xlabel('time(sec)');
title('Digital Signal at Transmitter');

subplot(3,1,2);
plot(t1,coded_bit,'lineWidth',2.5);
grid on;
axis([ 0 bp*length(coded) -2.5 2.5]);
ylabel('Amplitude(volt)');
xlabel('time(sec)');
title('Encoded signal at Duobinary-Encoder');

subplot(3,1,3);
plot(t1,decoded_digital_code_bit,'lineWidth',2.5);
grid on;
axis([ 0 bp*length(coded) -1.5 1.5]);
ylabel('Amplitude(volt)');
xlabel('time(sec)');
title('Digital Signal at Duobinary-Decoder');

%% With Error Plots

%orignal signal,staircase signal, recieved staircase signal,rec_low_pass

figure('Name','Adaptive Delta Modulation with noise','NumberTitle','off');
plot(t,x,'DisplayName','Message signal');
title('Adaptive Delta Modulation with error');
xlabel('Time (in sec)');
ylabel('Amplitude (in volts)');
hold 'on';
stairs(t,xq,'DisplayName','Staircase signal with noise');
```

```
stairs(t,rec_staircase_error,'DisplayName','Recieved Staircase signal with noise');
plot(t,y_error,'DisplayName','Recieved signal with noise');
hold 'off';
legend

%Binary signals

bp = .00001; %bit period
br = 1/bp;   %bit rate


digital_code_error_bit = [];
for n=1:1:length(digital_code_error)
    if digital_code_error(n)==1
       se=ones(1,100);
    else
        se=-1*ones(1,100);
    end
     digital_code_error_bit=[digital_code_error_bit,se];
end

coded_error_bit = [];
for n=1:1:length(coded_error)
    if coded_error(n)== 2
       c_se= 2*ones(1,100);
    elseif coded_error(n) == -2
        c_se= -2*ones(1,100);
    else
        c_se = zeros(1,100);
    end
     coded_error_bit=[coded_error_bit c_se];
end

decoded_digital_code_error_bit = [];
for n=1:1:length(decoded_digital_code_error)
    if decoded_digital_code_error(n)==1
       se=ones(1,100);
    else
        se=-1*ones(1,100);
    end
     decoded_digital_code_error_bit=[decoded_digital_code_error_bit se];
end


t1= bp/100:bp/100:100*length(x)*(bp/100);

figure('Name','Bit Transmission with error','NumberTitle','off')

subplot(3,1,1);
plot(t1,digital_code_error_bit,'lineWidth',2.5);
grid on;
axis([ 0 bp*length(digital_code_error) -1.5 1.5]);
ylabel('Amplitude(volt)');
xlabel('time(sec)');
title('Digital Signal at Transmitter');

subplot(3,1,2);
plot(t1,coded_error_bit,'lineWidth',2.5);
grid on;
axis([ 0 bp*length(coded_error) -2.5 2.5]);
ylabel('Amplitude(volt)');
xlabel('time(sec)');
title('Encoded signal at Duobinary-Encoder');

subplot(3,1,3);
plot(t1,decoded_digital_code_error_bit,'lineWidth',2.5);
grid on;
axis([ 0 bp*length(decoded_digital_code) -1.5 1.5]);
ylabel('Amplitude(volt)');
```

```
xlabel('time(sec)');
title('Decoded signal at Duobinary-Decoder');

%% Parameters Calculations

disp(' ');
disp('Parameters')
disp(' ');

error =   xor(digital_code_error,digital_code);                                    %error
disp('The error due to noise is');
disp(error);

Bit_error_rate = (sum(error(error==1)))/length(error);
disp('Bit Error Rate:');
disp(Bit_error_rate);


%SNR
my_snr_i = snr(xq,x_error-xq);
my_snr_o = snr(y,y_error-y);
disp('Input SNR');
disp(my_snr_i);
disp('Output SNR');
disp(my_snr_o);
```

Communication_system_duobinary_error.m

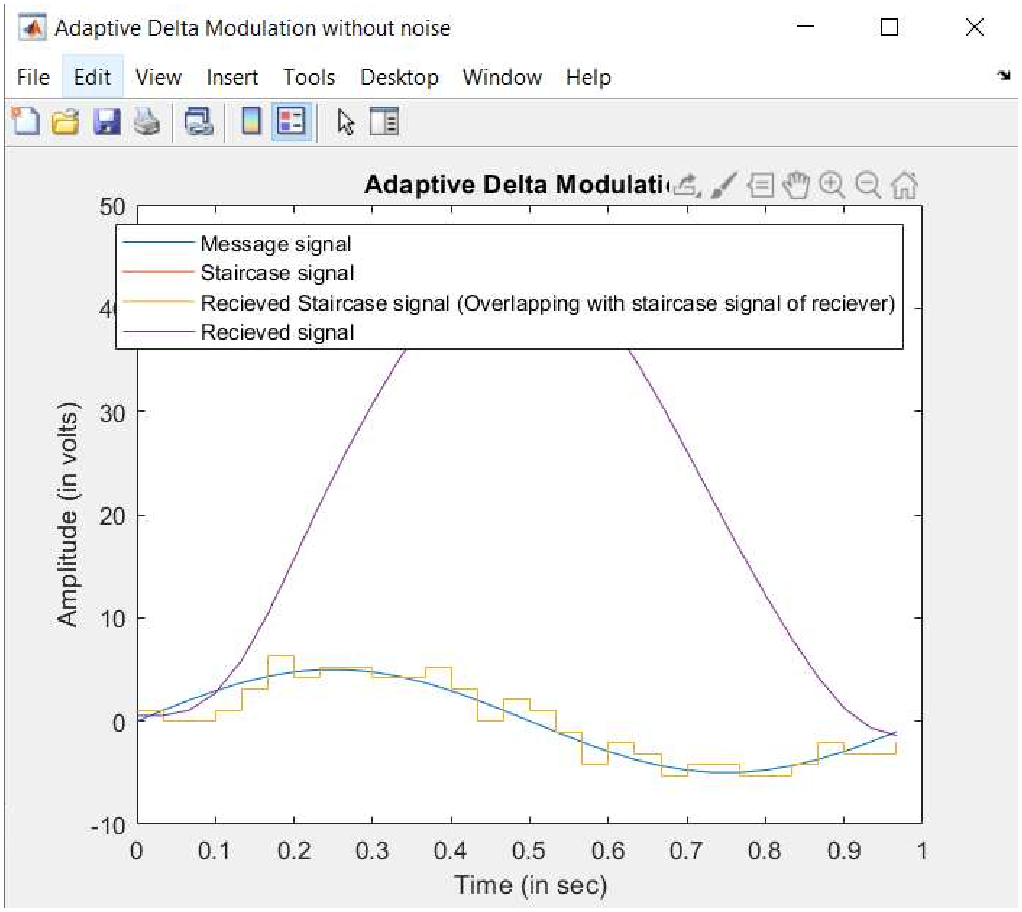
```
Am = 1;      %amplitude            %input('Enter the value for amplitude');
fm = 1;      %frequency            %input('Enter the value for frequency');
fs = 15*fm;  %sampling freq        %input('What is the sampling frequency?');

t = 0:1/fs:1;

%input signal
x = Am*sin(2*pi*fm*t);

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



[digital_code,xq,zero_track] = ad_deltamod(x,delta);
%Transmittor
disp('The digital code is:');
disp(digital_code);
disp(' ');

coded = Duobinary_Encoder(digital_code);
%duobinary_encoder
disp(' ');
disp('Encoded digital code is:');
disp(coded);
disp(' ');

coded = awgn(coded,-20);

decoded_digital_code = Duobinary_Decoder(coded);
%duobinary_decoder
disp('The output of duobinary decoder is');
disp(decoded_digital_code);
disp(' ');

[rec_staircase,my_signal] = ad_deltademod(decoded_digital_code,zero_track,delta);
%Reciever
```

```
y = lowpass(my_signal,1,fs);


error = xor(decoded_digital_code,digital_code);
disp(' ');
disp('error');
disp(error);

figure('Name','Adaptive Delta Modulation(Noise at Duo-binary
Encoder)','NumberTitle','off');
plot(t,x,'DisplayName','Message signal');
title('Adaptive Delta Modulation');
xlabel('Time (in sec)');
ylabel('Amplitude (in volts)');
hold 'on';
stairs(t,xq,'DisplayName','Staircase signal');
stairs(t,rec_staircase,'DisplayName','Recieved Staircase signal at reciever');
plot(t,y,'DisplayName','Recieved signal');
hold 'off';
legend
```

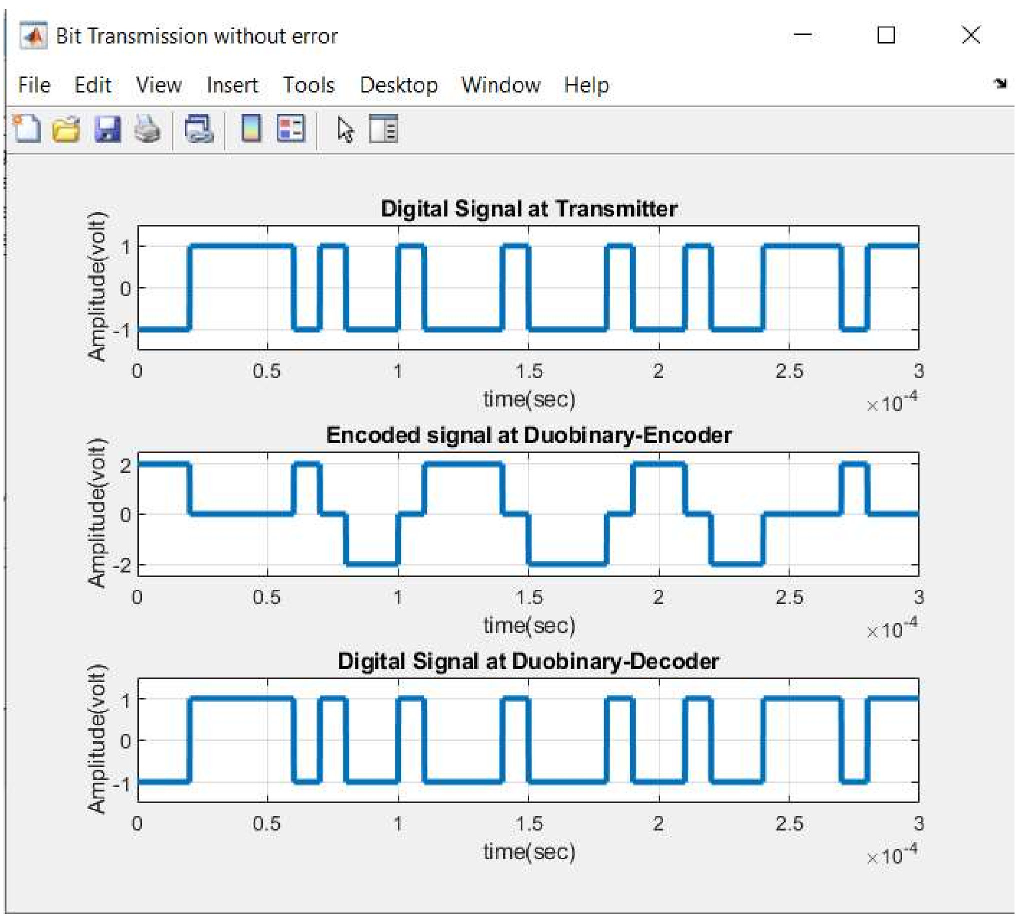
6. Output:

Without Noise Output and Plots:

```
Adaptive Delta Modulation without noise

The digital code is:
  Columns 1 through 22

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

  Columns 23 through 30

     0     0     1     1     1     0     1     1


Precoded output in binary:
  Columns 1 through 22

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

  Columns 23 through 30

     0     0     1     0     1     1     0     1

.................................................................
Precoded output in voltage:
  Columns 1 through 22

     1     1    -1     1    -1     1     1    -1    -1    -1     1     1     1     1    -1    -1    -1    -1     1     1     1    -1

  Columns 23 through 30

    -1    -1     1    -1     1     1    -1     1

.................................................................

Encoded digital code is:
  Columns 1 through 22

     2     2     0     0     0     0     2     0    -2    -2     0     2     2     2     0    -2    -2    -2     0     2     2     0

  Columns 23 through 30

    -2    -2     0     0     0     2     0     0


The output of duobinary decoder is
  Columns 1 through 22

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

  Columns 23 through 30

     0     0     1     1     1     0     1     1
```



Adaptive Delta Modulation without noise
File Edit View Insert Tools Desktop Window Help
Adaptive Delta Modulati
Message signal
Staircase signal
Recieved Staircase signal (Overlapping with staircase signal of reciever)
Recieved signal
Amplitude (in volts)
Time (in sec)

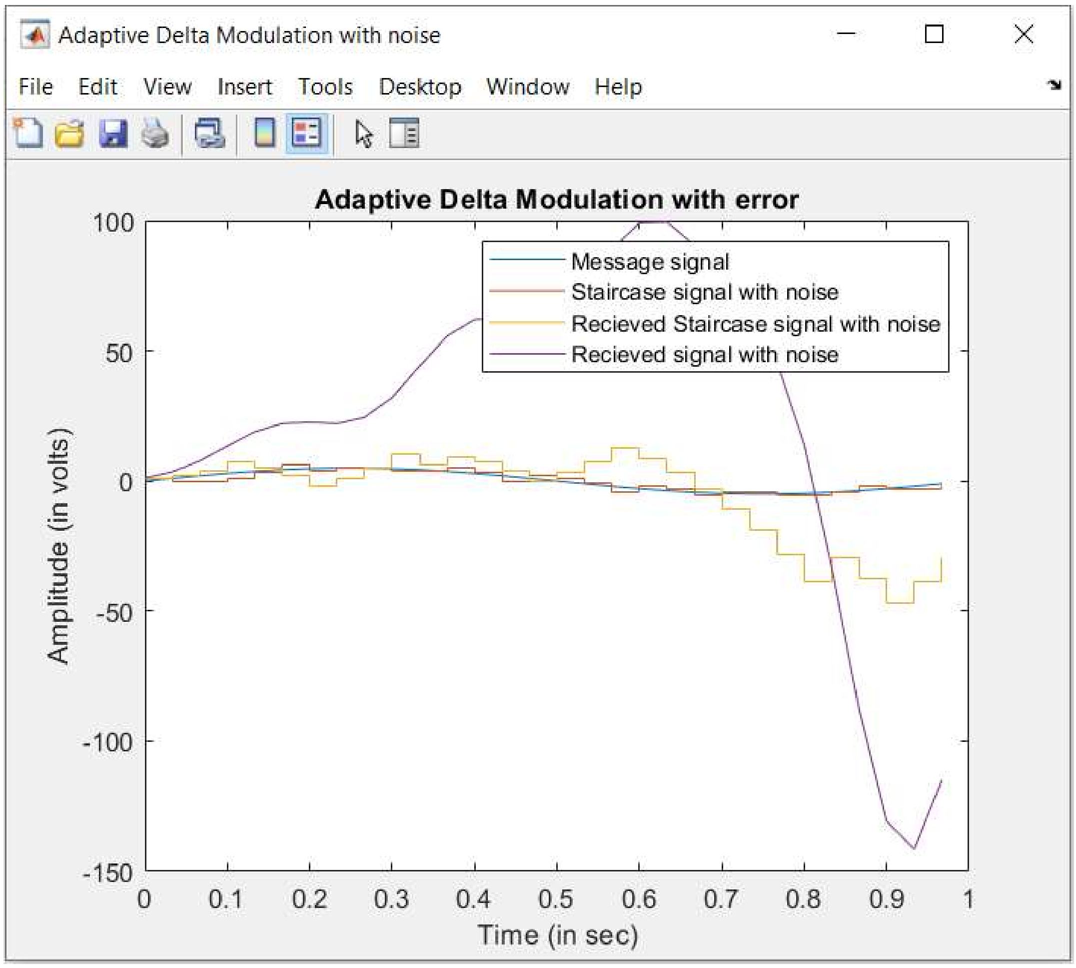


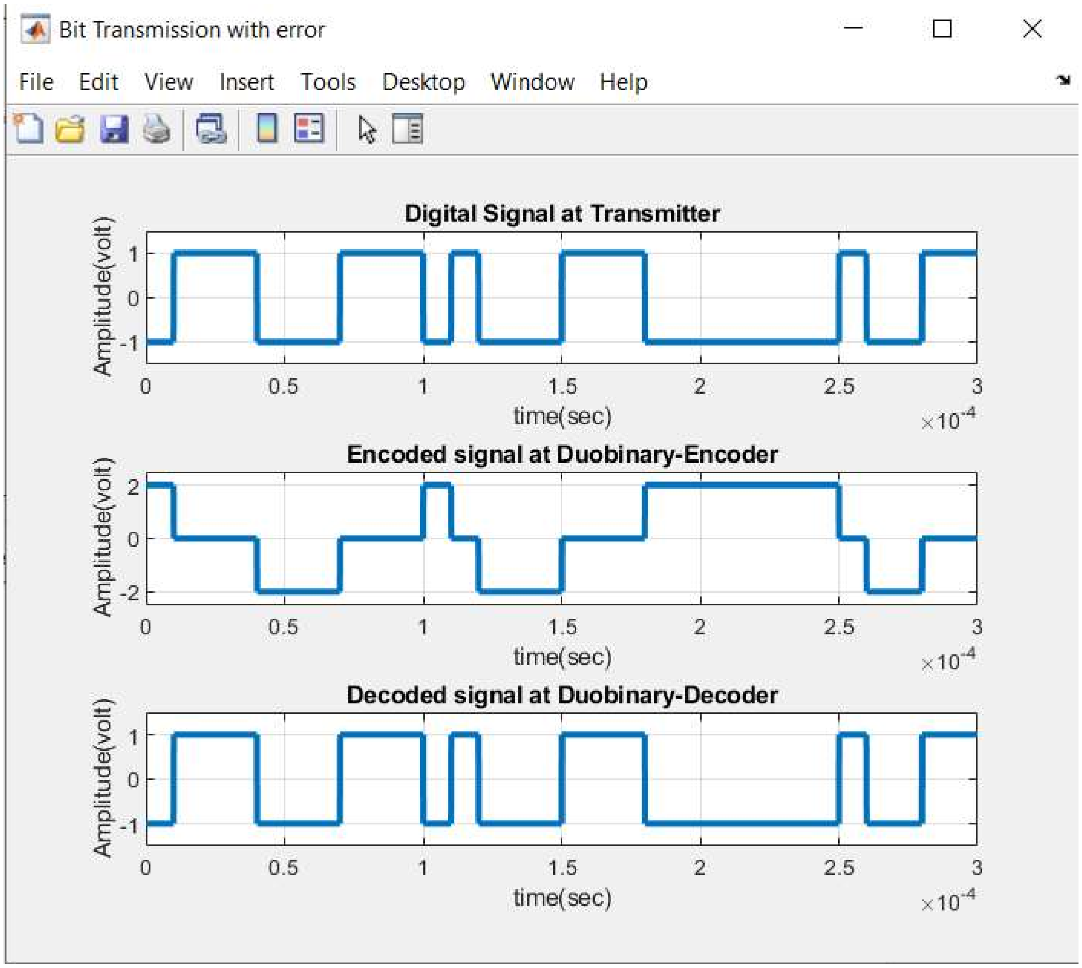
Bit Transmission without error
File Edit View Insert Tools Desktop Window Help
Digital Signal at Transmitter
Encoded signal at Duobinary-Encoder
Digital Signal at Duobinary-Decoder
Amplitude(volt)
time(sec)
×10^-4

With Noise Output and Plots:

```
Adaptive Delta Modulation with noise

The noisy digital code is:
  Columns 1 through 22

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

  Columns 23 through 30

     0     0     0     1     0     0     1     1

...................................................................
Precoded output in binary:
  Columns 1 through 22

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

  Columns 23 through 30

     1     1     1     0     0     0     1     0

...................................................................
Precoded output in voltage:
  Columns 1 through 22

     1    -1     1    -1    -1    -1    -1     1    -1     1     1    -1    -1    -1    -1     1    -1     1     1     1     1     1

  Columns 23 through 30

     1     1     1    -1    -1    -1     1    -1

...................................................................
Encoded digital data is:
  Columns 1 through 22

     2     0     0     0    -2    -2    -2     0     0     0     2     0    -2    -2    -2     0     0     0     2     2     2     2

  Columns 23 through 30

     2     2     2     0    -2    -2     0     0

...................................................................
The output of duobinary decoder is
  Columns 1 through 22

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

  Columns 23 through 30

     0     0     0     1     0     0     1     1

...................................................................

Parameters

The error due to noise is
   0   1   0   0   1   1   0   0   1   1   1   1   0   0   1   1   1   1   1   0   0   1   0   0   1   0   1   0   0   0
```

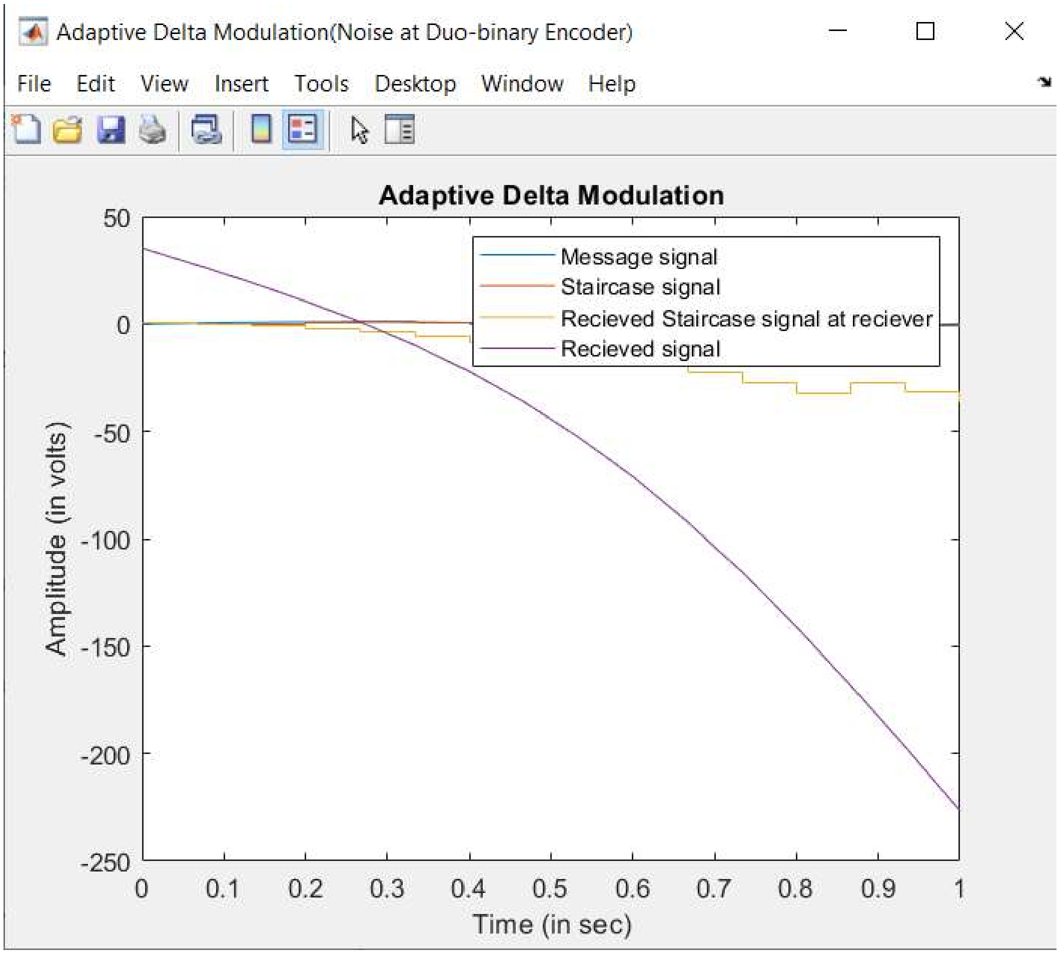
```
Bit Error Rate:
    0.5000

Input SNR
    1.3569

Output SNR
   -5.7311
```





When the error was introduced in between Duobinary Encoder:

```
The digital code is:
     0     0     1     1     1     0     0     1     0     0     0     0     1     1     0     1


Precoded output in binary:
     1     1     0     1     0     0     0     1     1     1     1     1     0     1     1     0

..................................................................
Precoded output in voltage:
     1     1    -1     1    -1    -1    -1     1     1     1     1     1    -1     1     1    -1

..................................................................

Encoded digital code is:
     2     2     0     0     0    -2    -2     0     2     2     2     2     0     0     2     0


The output of duobinary decoder is
     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0



error
   0   0   1   1   1   0   0   1   0   0   0   0   1   0   0   1
```


Adaptive Delta Modulation(Noise at Duo-binary Encoder)
File Edit View Insert Tools Desktop Window Help
Adaptive Delta Modulation
Message signal
Staircase signal
Recieved Staircase signal at reciever
Recieved signal
Amplitude (in volts)
Time (in sec)
50
0
-50
-100
-150
-200
-250
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

7. Conclusion

An end-to-end Digital Communication System in form of Adaptive delta Modulation with duo-binary Signaling was observed with and without noise and the parameters such as error bit stream, bit-error rate and Signal to Noise Ratio(SNR) were calculated.

8. Bibliography

- www.sciencedirectassets.com
- https://research.ijcaonline.org/icedsp/number2/icedsp1016.pdf
- Digtal Communications By Simon Haykins
- https://www.ieee.org/