Contents


```
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');
```

```
Adaptive Delta Modulation without noise

The digital code is:
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30

     1     0     1     1


Precoded output in binary:
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30

     1     1     0     1
```

```
.....................................................................
Precoded output in voltage:
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30

    1     1    -1     1

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

Encoded digital code is:
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30

    0     2     0     0


The output of duobinary decoder is
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30

    1     0     1     1
```

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');
```

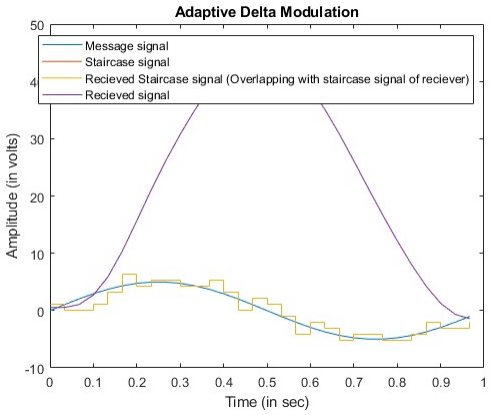
```
Adaptive Delta Modulation with noise

The noisy digital code is:
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30
```

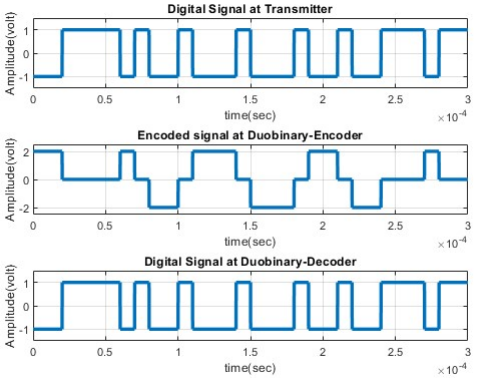
```
     1     0     0     0

.....................................................................
Precoded output in binary:
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30

     0     0     0     0

.....................................................................
Precoded output in voltage:
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30

    -1    -1    -1    -1

.....................................................................
Encoded digital data is:
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30

     0    -2    -2    -2

.....................................................................
The output of duobinary decoder is
  Columns 1 through 13

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

  Columns 14 through 26

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

  Columns 27 through 30

     1     0     0     0

.....................................................................
```

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');
```

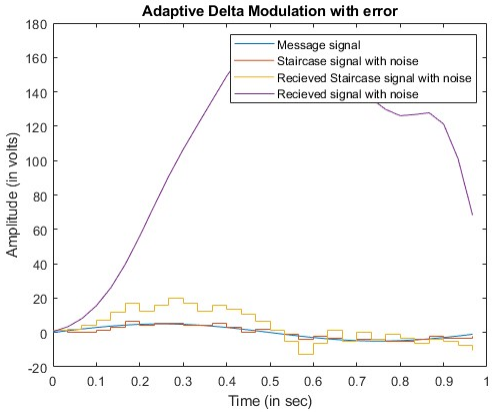


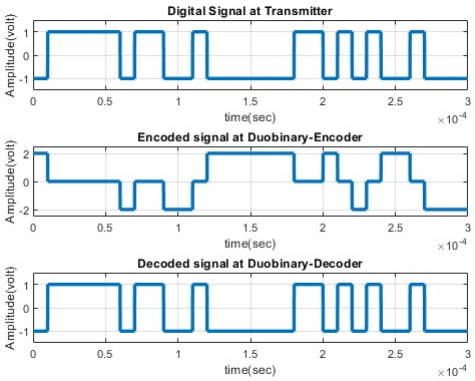


With Error Plots

```
%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);
```

```
Parameters

The error due to noise is
  Columns 1 through 19

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

  Columns 20 through 30

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

```
Bit Error Rate:
    0.3667

Input SNR
    1.3636

Output SNR
  -10.6834
```

Published with MATLAB® R2019b