

1.PULSE CODE MODULATION AND DEMODULATION

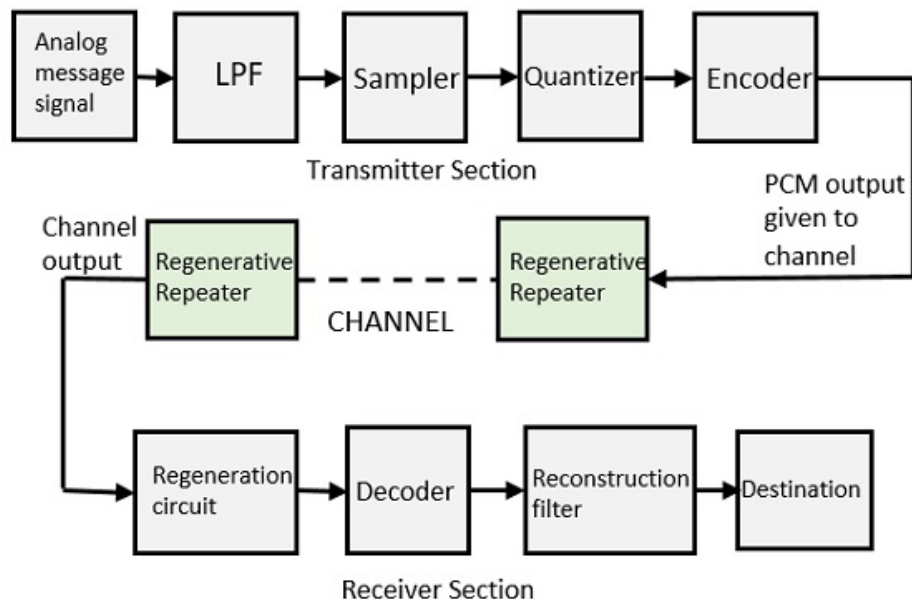
Theory:

Pulse-code modulation (PCM) is a method used to digitally represent sampled analog signals. It is the standard form of digital audio in computers, compact discs, digital telephony and other digital audio applications. In a PCM stream, the amplitude of the analog signal is sampled regularly at uniform intervals, and each sample is quantized to the nearest value within a range of digital steps.

Modulation is the process of varying one or more parameters of a carrier signal in accordance with the instantaneous values of the message signal. In Pulse Code Modulation, the message signal is represented by a sequence of coded pulses. This message signal is achieved by representing the signal in discrete form in both time and amplitude.

The transmitter section of a Pulse Code Modulator circuit consists of Sampling, Quantizing and Encoding, which are performed in the analog-to-digital converter section. The low pass filter prior to sampling prevents aliasing of the message signal. The basic operations in the receiver section are regeneration of impaired signals, decoding, and reconstruction of the quantized pulse train.

Block diagram:



TRANSMITTER END			
Block name	Input Signal	Function of block	Output Signal
Low pass filter	Analog message signal	converts the input signal into band limited signal	band limited signals

sampler	band limited signals	Converts the continuous signal into discrete values	Discrete signals
Quantizer	Discrete signals	Limits the number of different amplitude values by rounding off the amplitude values	Quantized signals
Encoder	Quantized signals	Designates each quantized level by a binary code	Coded signals

RECIEVER END			
Block name	Input Signal	Function of block	Output Signal
Regenerative circuit	Coded signals	Increases the signal strength	Coded signals
Decoder	Coded signals	Decodes the pulse coded waveform to reproduce the predefined quantized values	Quantized signals
Reconstruction filter	Quantized signals	Consist of a low pass filter which smoothens the quantized signals to get the original analog message signal	Analog message signal

Matlab Code:

```

fs = 100;
t = 0:1/fs:1;
x = 1.5+sin(2*pi*3*t)+sin(2*pi*4.5*t)+sin(2*pi*7*t);
subplot(511);
plot(t,x,'b'); grid on; hold on
title("Original Band limited Signal");
%% Sampling
subplot(512);
stem(t,x);hold on;

```

```

title("Sampled Signal");

yq = []; bn = [];
for n=1:length(t)
    if x(n)>=-2 && x(n)<-1
        y = -1.5;
        b = 'ooo';
    elseif x(n)>=-1 && x(n)<0
        y = -0.5;
        b = 'ooi';
    elseif x(n)>=0 && x(n)<1
        y = 0.5;
        b = 'oio';
    elseif x(n)>=1 && x(n)<2
        y = 1.5;
        b = 'oi1';
    elseif x(n)>=2 && x(n)<3
        y = 2.5;
        b = '1oo';
    elseif x(n)>=3 && x(n)<4
        y = 3.5;
        b = '1oi';
    elseif x(n)>=4 && x(n)<5
        y = 4.5;
        b = '1io';
    end
    yq = [yq, y];
    bn = [bn, b];
end
subplot(513)
plot(t,yq,'r'); hold on;
title('Quantized Signal');

%% decoder
y_decod=[];
for n=1:3:length(bn)
    if bn(n)=='o' && bn(n+1)=='o' && bn(n+2)=='o'
        yd=-1.5;
    elseif bn(n)=='o' && bn(n+1)=='o' && bn(n+2)=='i'
        yd=-0.5;
    elseif bn(n)=='o' && bn(n+1)=='i' && bn(n+2)=='o'
        yd=0.5;
    elseif bn(n)=='o' && bn(n+1)=='i' && bn(n+2)=='i'
        yd=1.5;

```

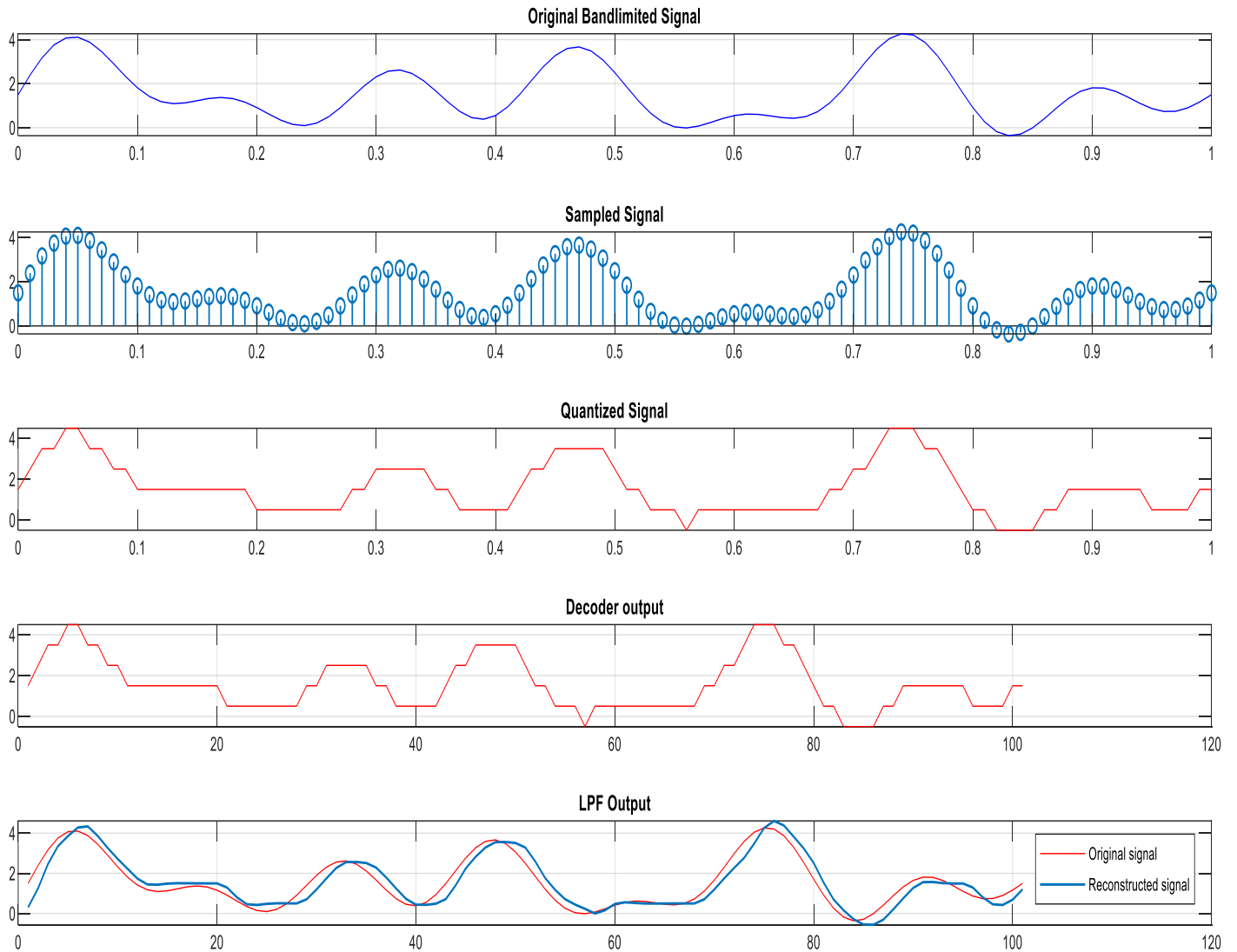
```

elseif bn(n)=='1' && bn(n+1)=='o' && bn(n+2)=='o'
    yd=2.5;
elseif bn(n)=='1' && bn(n+1)=='o' && bn(n+2)=='1'
    yd=3.5;
elseif bn(n)=='1' && bn(n+1)=='1' && bn(n+2)=='o'
    yd=4.5;
end
y_decod = [y_decod, yd];
end
t=1:101;
subplot(514)
plot(y_decod,'r');
hold on; grid on;
title("Decoder output");
%legend('levels=7');

% butterworth lpf
tn=1:101;
subplot(515)
plot(t,x,'r');
hold on; grid on;
legend('level=2^3');
[b,a]=butter(2,0.4,'low');
x_filter = filter(b,a,y_decod);
plot(tn,x_filter);
legend('Original Signal','Reconstructed Signal');
title('LPF Output');

```

Results:



Explanation

Figure 1: Original Signal

It depicts the original band limited signal i.e. the source data which is to be transmitted varying in between 0 to 1.

$$x(t) = 1.5 + \sin(2\pi \cdot 3 \cdot t) + \sin(2\pi \cdot 4.5 \cdot t) + \sin(2\pi \cdot 7 \cdot t)$$

Here (3,4,5,7) are the frequencies of the original signal.

Figure 2: Sampled Signal

It represents the sampled output of the original signal after being sampled with a sampling interval of $1/f_s$ where $f_s=100$.

Figure 3: Quantized Signal

This graph shows the signal $x(t)$ after passing it through a Quantizer and an Encoder. Here the Quantizer used consists of **7 levels** and values are quantized to the mid values. The encoder used is of **3 bits**.

Maximum possible value of $x(t) = 1.5 + 3(\max(\text{sine})) = 4.5$

Minimum possible value of $x(t) = 1.5 - 3(\min(\text{sine})) = -1.5$

So the Quantizer limit is from **-1.5 to 4.5**.

Figure 4: Decoder Output

This graph shows the received signal after passing through the decoder. This graph is almost similar to the Quantizer output provided no noise added.

Figure 5: LPF Output

This is the final received signal obtained by passing the decoder output through a 2nd order low pass butterworth filter having cutoff of 0.2π rad/sample.

In this graph we can see all the different processes the signal goes through to be transmitted with minimum interference. The Pulse Code Modulator circuit digitizes the given analog signal, codes it and samples it, and then transmits it in an analog form. This whole process is repeated in a reverse pattern to obtain the original signal.

2.DIFFERENTIAL PULSE CODE MODULATION AND DEMODULATION

Theory:

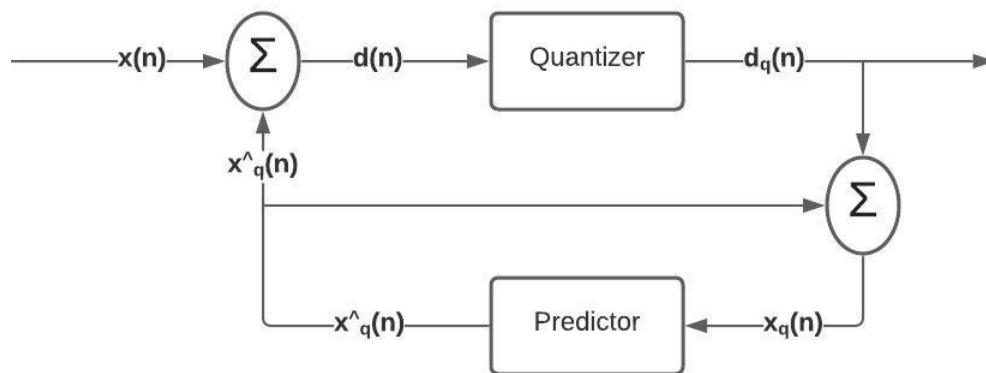
Differential pulse code modulation (DPCM) is a procedure of converting an analog into a digital signal in which an analog signal is sampled and then the difference between the actual sample value and its predicted value (predicted value is based on previous sample or samples) is quantized and then encoded forming a digital value.

For the samples that are highly correlated, when encoded by PCM technique, leave redundant information behind. To process this redundant information and to have a better output, it is a wise decision to take a predicted sampled value, assumed from its previous output and summarize them with the quantized values. Such a process is called as Differential PCM i.e DPCM technique.

Block diagram:

DPCM transmission end:

To encoder



- $x(n)$: sampled signal
- $x_q(n)$: quantized value of $x(n)$
- $\hat{x}_q(n)$: predicted sample of $z_q(n)$
- $d(n)$: difference sample ,i.e. difference between $x(n)$ and $\hat{x}_q(n)$
- $d_q(n)$: quantized value of $d(n)$
- $q(n)$: quantization error.

Working:

The input sampled signal $x(n)$ is passed through a summer where the n^{th} order predicted quantized signal $\hat{x}_q(n)$ is subtracted from it. The output of the summer is $d(n)$ which is called difference sample. This $d(n)$ signal is passed through a Quantizer to get a quantized difference sample ,thus we get $d_q(n)$. One part of this $d_q(n)$ is given as input to the encoder for further transmission. The other part is added with the with $\hat{x}_q(n)$ to get the quantized signal $x_q(n)$. This $x_q(n)$ is given as input to the Prediction filter whose output is used as a feedback .

The equations involved are:

$$d(n) = x(n) - \hat{x}_q(n) \text{ - - - - - (1)}$$

$$\Rightarrow x(n) = d(n) + \hat{x}_q(n) \text{ - - - - - (2)}$$

$$d_q(n) = d(n) + q(n) \text{ - - - - - (3)}$$

We know that, $x_q(n) = x(n) + q(n)$. So substituting equation (2) we get

$$x_q(n) = d(n) + \hat{x}_q(n) + q(n)$$

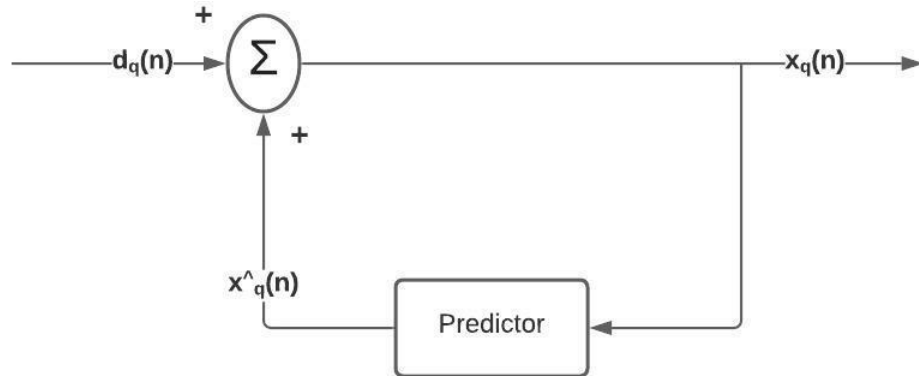
Now using equation (3) we get

$$x_q(n) = \hat{x}_q(n) + d_q(n) \text{ - - - - - (4)}$$

From equation (4) we can conclude that the prediction filter input is:

$$x_q(n) = \hat{x}_q(n) + d_q(n)$$

DPCM receiver end:



First order prediction filter:

Using Taylor series expansion we get,

$$\hat{x}(n) = x(n-1)$$

So the difference sample,

$$d(n) = x(n) - \hat{x}(n)$$

$$\Rightarrow d(n) = x(n) - x(n-1)$$

Second order prediction filter:

Using Taylor series expansion we get,

$$x^{(n)} = 2x(n-1) - x(n-2)$$

So the difference sample,

$$d(n) = x(n) - x^{(n)}$$

$$\Rightarrow \boxed{d(n) = x(n) - [2x(n-1) - x(n-2)]}$$

Matlab Code:

Using 1st order prediction filter:

```
t=0:0.01:1;
f1=10;
f2=2;
x=2+3*sin(2*pi*f1*t)+1.5*cos(2*pi*f2*t);
subplot(511);
hold on;
title('original siganal');
plot(t,x,'k');
hold on;
grid on;

subplot(512);
stem(t,x);hold on;title('sampled signal');

%% 1st Order Prediction filter
yp=[];
for k=1:length(x)
    if k==1
        yp(k)= 0;
    else
        yp(k)=x(k-1);
    end
end
dp = x-yp;
subplot(513)
plot(t,dp);
title('1st order predicted DPCM');
```

```

y_quant=[];
bn=[];
for n=1:length(t)
    if dp(n)>=-2 && dp(n)<-1
        y=-1.5;
        b='ooo';
    elseif dp(n)>=-1 && dp(n)<0
        y=-0.5;
        b='ooi';
    elseif dp(n)>=0 && dp(n)<1
        y=0.5;
        b='oio';
    elseif dp(n)>=1 && dp(n)<2
        y=1.5;
        b='oi';
    elseif dp(n)>=2 && dp(n)<3
        y=2.5;
        b='ioo';
    elseif dp(n)>=3 && dp(n)<4
        y=3.5;
        b='ioi';
    elseif dp(n)>=4 && dp(n)<5
        y=4.5;
        b='iio';
    elseif dp(n)>=5 && dp(n)<6
        y=5.5;
        b='iii';
    end
    y_quant=[y_quant, y];
    bn=[bn, b];
end
subplot(514)
plot(t,y_quant,'r');
hold on;
title('Quantized Signal');
legend('Level=2^3');

%% Decoder
y_decod=[];
for n=1:3:length(bn)
    if bn(n)=='o' && bn(n+1)=='o' && bn(n+2)=='o'
        yd=-1.5;
    elseif bn(n)=='o' && bn(n+1)=='o' && bn(n+2)=='i'

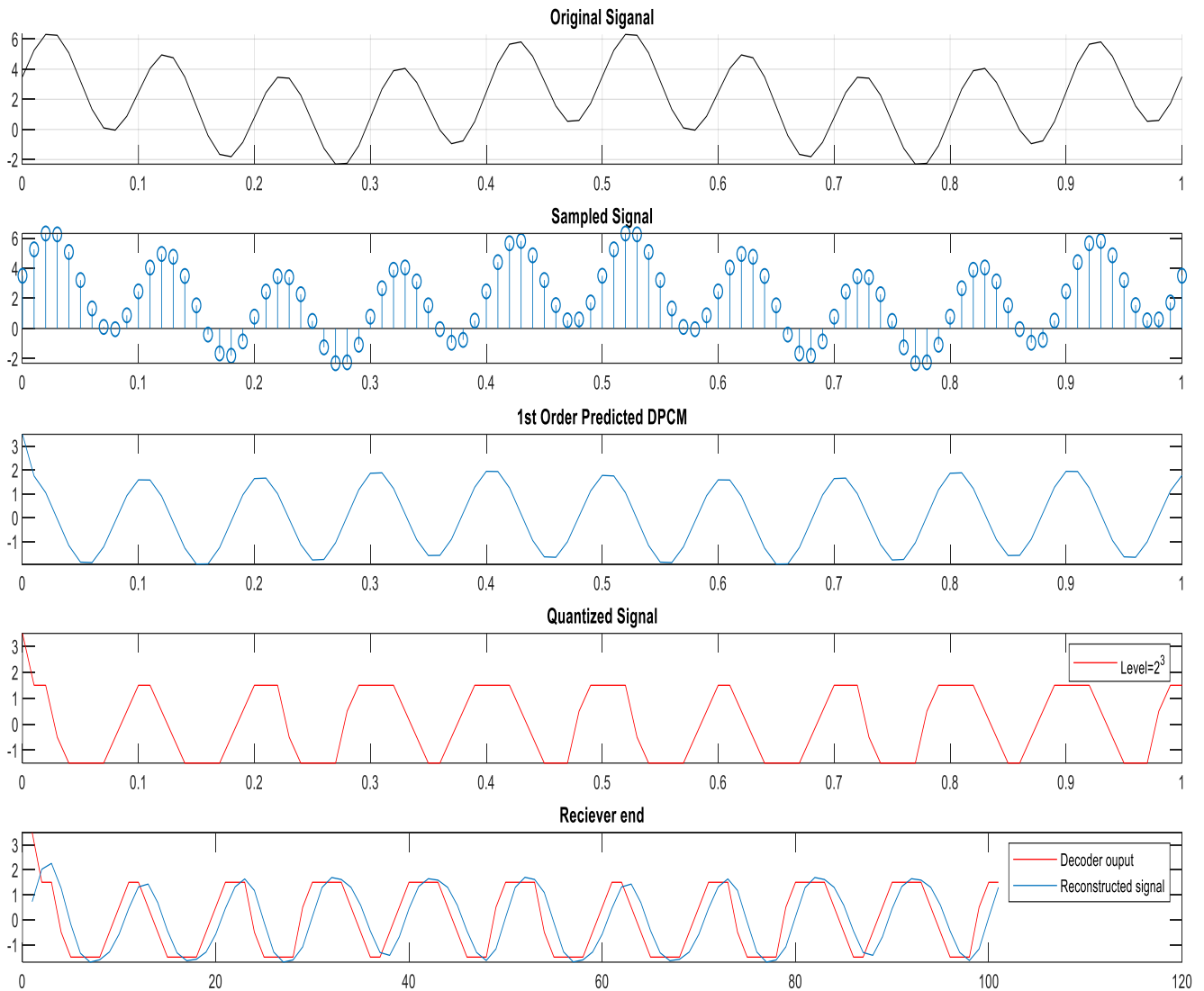
```

```

        yd=-0.5;
    elseif bn(n)=='o' && bn(n+1)=='i' && bn(n+2)=='o'
        yd=0.5;
    elseif bn(n)=='o' && bn(n+1)=='i' && bn(n+2)=='i'
        yd=1.5;
    elseif bn(n)=='i' && bn(n+1)=='o' && bn(n+2)=='o'
        yd=2.5;
    elseif bn(n)=='i' && bn(n+1)=='o' && bn(n+2)=='i'
        yd=3.5;
    elseif bn(n)=='i' && bn(n+1)=='i' && bn(n+2)=='o'
        yd=4.5;
    elseif bn(n)=='i' && bn(n+1)=='i' && bn(n+2)=='i'
        yd=5.5;
    end
    y_decod = [y_decod, yd];
end
t=1:101;
subplot(515)
plot(y_decod,'r');
hold on;
legend('level=2^3');
[b,a]=butter(2,0.4,'low');
x_filter = filter(b,a,y_decod);
plot(t,x_filter);
legend('decoder ouput','reconstructed signal');
title('reciever end');

```

Output:



Matlab Code:

Using 2nd order prediction filter:

```
clc; clear all; close all;
```

```
%% band limited continous time signal
```

```
t=0:0.01:1;
```

```
f1=10;
```

```
f2=2;
```

```
x=2+3*sin(2*pi*f1*t)+1.5*cos(2*pi*f2*t);
```

```
subplot(51);
```

```

hold on;
title('original signal');
plot(t,x,'k');
hold on;
grid on;

%% sampling
subplot(512);
stem(t,x);hold on;title('sampled signal');

%% 2nd order prediction filter
p=[2,-1];
yp=[];
for k=1:length(x)
    if k==1
        yp(k)=0;
    elseif k==2
        yp(k)=p(1)*x(k-1);
    else
        yp(k)=p(1)*x(k-1) + p(2)*x(k-2);
    end
end
dp = x-yp;
subplot(513)
plot(t,dp);
title('predicted DPCM');

%% quantizer and encoder
y_quant=[];
bn=[];
for n=1:length(t)
    if dp(n)>=-2 && dp(n)<-1
        y=-1.5;
        b='ooo';
    elseif dp(n)>=-1 && dp(n)<0
        y=-0.5;
        b='oo1';
    elseif dp(n)>=0 && dp(n)<1
        y=0.5;
        b='o1o';
    elseif dp(n)>=1 && dp(n)<2
        y=1.5;
        b='o11';
    elseif dp(n)>=2 && dp(n)<3

```

```

        y=2.5;
        b='100';
    elseif dp(n)>=3 && dp(n)<4
        y=3.5;
        b='101';
    elseif dp(n)>=4 && dp(n)<5
        y=4.5;
        b='110';
    elseif dp(n)>=5 && dp(n)<6
        y=5.5;
        b='111';
    end
    y_quant=[y_quant, y];
    bn=[bn, b];
end
subplot(514)
plot(t,y_quant,'r');
hold on;
title('quantized signal');
legend('level=2^3');

%% decoder
y_decod=[];
for n=1:length(bn)
    if bn(n)=='o' && bn(n+1)=='o' && bn(n+2)=='o'
        yd=-1.5;
    elseif bn(n)=='o' && bn(n+1)=='o' && bn(n+2)=='1'
        yd=-0.5;
    elseif bn(n)=='o' && bn(n+1)=='1' && bn(n+2)=='o'
        yd=0.5;
    elseif bn(n)=='o' && bn(n+1)=='1' && bn(n+2)=='1'
        yd=1.5;
    elseif bn(n)=='1' && bn(n+1)=='o' && bn(n+2)=='o'
        yd=2.5;
    elseif bn(n)=='1' && bn(n+1)=='o' && bn(n+2)=='1'
        yd=3.5;
    elseif bn(n)=='1' && bn(n+1)=='1' && bn(n+2)=='o'
        yd=4.5;
    elseif bn(n)=='1' && bn(n+1)=='1' && bn(n+2)=='1'
        yd=5.5;
    end
    y_decod = [y_decod, yd];
end

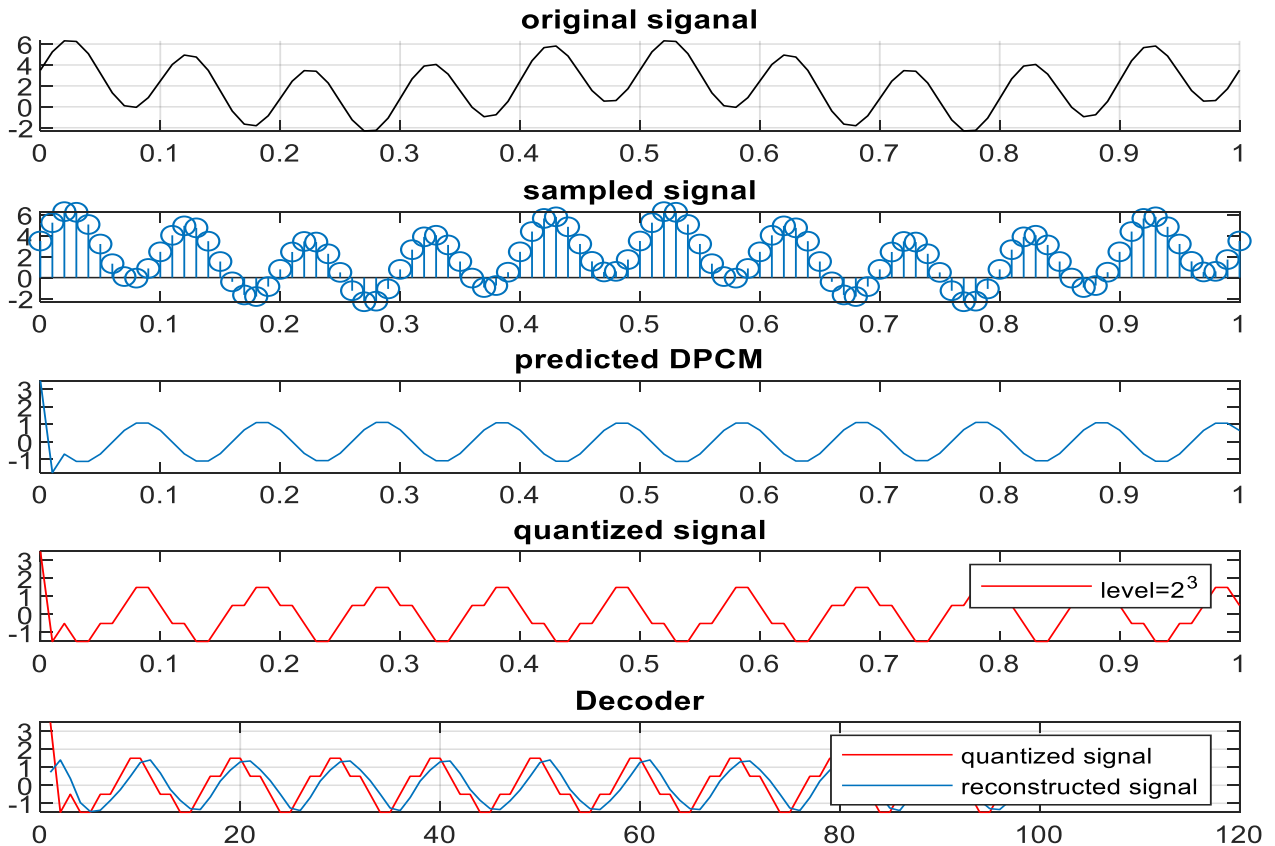
```

```

t=1:101;
subplot(5,1,5)
plot(y_decod,'r');
hold on; grid on;
legend('level=2^3');
[b,a]=butter(2,0.4,'low');
x_filter = filter(b,a,y_decod);
plot(t,x_filter);
legend('quantized signal','reconstructed signal');
title('Decoder');

```

Output:



Matlab Code:

Using 3rd order prediction filter:

```

clc; clear all; close all;

```



```
%% band limited continous time signal
```

```
t=0:0.01:1;  
f1=10;  
f2=2;  
x=2+3*sin(2*pi*f1*t)+1.5*cos(2*pi*f2*t);  
subplot(511);  
hold on;  
title('original siganal');  
plot(t,x,'k');  
hold on;  
grid on;
```

```
%% sampling
```

```
subplot(512);  
stem(t,x);hold on;title('sampled signal');
```

```
%% 3rd order prediction filter
```

```
p=[3,-2,-1];  
yp=[];  
for k=1:length(x)  
    if k==1  
        yp(k)=0;  
    elseif k==2  
        yp(k)=p(1)*x(k-1);  
    elseif k==3  
        yp(k)=p(1)*x(k-1) + p(2)*x(k-2);  
    else  
        yp(k)=p(1)*x(k-1) + p(2)*x(k-2) + p(3)*x(k-3);  
    end  
end  
dp = x-yp;  
subplot(513)  
plot(t,dp);  
title('predicted DPCM');
```

```
%% quantizer and encoder
```

```
y_quant=[];  
bn=[];  
for n=1:length(t)  
    if dp(n)>=-2 && dp(n)<-1  
        y=-1.5;  
        b='ooo';  
    elseif dp(n)>=-1 && dp(n)<0  
        y=-0.5;
```

```

        b='001';
elseif dp(n)>=0 && dp(n)<1
    y=0.5;
    b='010';
elseif dp(n)>=1 && dp(n)<2
    y=1.5;
    b='011';
elseif dp(n)>=2 && dp(n)<3
    y=2.5;
    b='100';
elseif dp(n)>=3 && dp(n)<4
    y=3.5;
    b='101';
elseif dp(n)>=4 && dp(n)<5
    y=4.5;
    b='110';
elseif dp(n)>=5 && dp(n)<6
    y=5.5;
    b='111';
end
y_quant=[y_quant, y];
bn=[bn, b];
end
subplot(514)
plot(t,y_quant,'r');
hold on;
title('quantized signal');
legend('level=2^3');

%% decoder
y_decod=[];
for n=1:3:length(bn)
    if bn(n)=='0' && bn(n+1)=='0' && bn(n+2)=='0'
        yd=-1.5;
    elseif bn(n)=='0' && bn(n+1)=='0' && bn(n+2)=='1'
        yd=-0.5;
    elseif bn(n)=='0' && bn(n+1)=='1' && bn(n+2)=='0'
        yd=0.5;
    elseif bn(n)=='0' && bn(n+1)=='1' && bn(n+2)=='1'
        yd=1.5;
    elseif bn(n)=='1' && bn(n+1)=='0' && bn(n+2)=='0'
        yd=2.5;
    elseif bn(n)=='1' && bn(n+1)=='0' && bn(n+2)=='1'

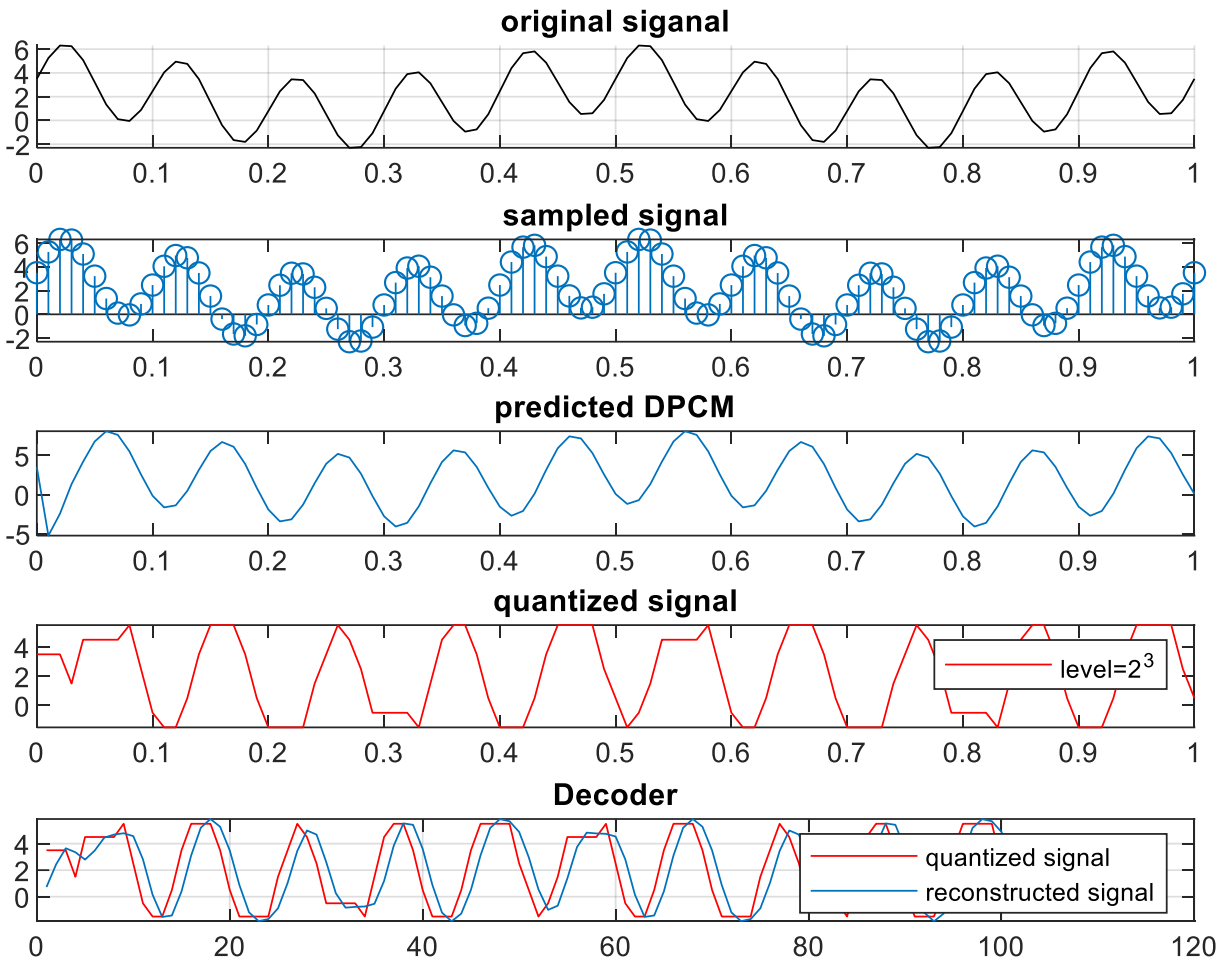
```

```

        yd=3.5;
    elseif bn(n)=='1' && bn(n+1)=='1' && bn(n+2)=='0'
        yd=4.5;
    elseif bn(n)=='1' && bn(n+1)=='1' && bn(n+2)=='1'
        yd=5.5;
    end
    y_decod = [y_decod, yd];
end
t=1:101;
subplot(515)
plot(y_decod,'r');
hold on; grid on;
legend('level=2^3');
[b,a]=butter(2,0.4,'low');
x_filter = filter(b,a,y_decod);
plot(t,x_filter);
legend('quantized signal','reconstructed signal');
title('Decoder');

```

Output:



Result & Explanation:
Figure 1: Original Signal

It depicts the original band limited signal i.e. the source data which is to be transmitted varying in between 0 to 1.

$$x(t) = 2 + 3 \sin(2\pi f_1 t) + 1.5 \cos(2\pi f_2 t)$$

Here (f_1, f_2) are the frequencies of the original signal.

Figure 2: Sampled Signal

It represents the sampled output of the original signal after being sampled with a sampling interval of $1/f_s$ where $f_s=100$.

Figure 3: 1st order DPCM Output

This graph shows the signal $x(t)$ after passing it through a 1st order prediction filter.

Figure 3: Quantized Signal

This graph shows the signal $x(t)$ after passing it through a Quantizer and an Encoder. Here the Quantizer used consists of **8 levels** and values are quantized to the mid values. The encoder used is of **3 bits**.

Maximum possible value of $x(t) = 2 + 3(\max(\text{sine}) + 1.5\max(\text{cosine})) = 5.5$

Minimum possible value of $x(t) = 2 + 3(\min(\text{sine}) + 1.5\min(\text{cosine})) = -1.0$

So the Quantizer limit is from **-1.0 to 5.5**.

Figure 4: Decoder Output

This graph shows the received signal after passing through the decoder. This graph is almost similar to the Quantizer output provided no noise added.

Figure 5: LPF Output

This is the final received signal obtained by passing the decoder output through a 2nd order low pass butterworth filter having cutoff of 0.2π rad/sample.

The input here is a continuous-time signal, so it is sampled first so that a discrete-time signal is the input to the DPCM encoder.

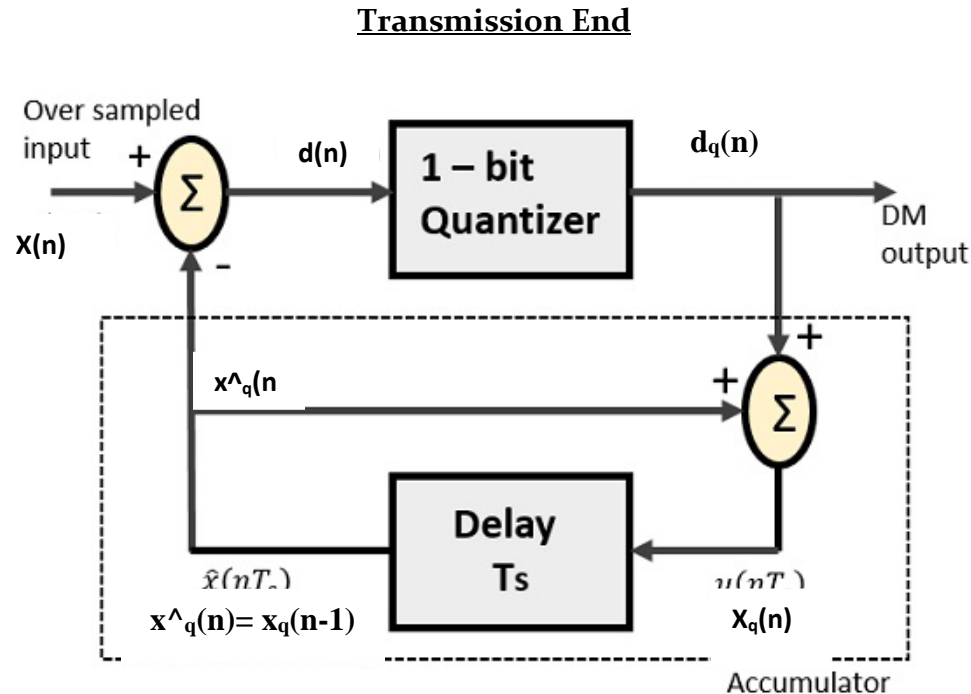
3. DELTA MODULATION AND DEMODULATION.

Theory:

Delta Modulation: In This type of modulation, the sampling rate is much higher and the step size after quantization is of a smaller value Δ . Delta Modulation is a simplified form of DPCM technique, also viewed as **1-bit DPCM scheme**.

The type of modulation, where the sampling rate is much higher and in which the step-size after quantization is of a smaller value Δ , such a modulation is termed as delta modulation. Delta Modulation is a simplified form of DPCM technique, also viewed as 1-bit DPCM scheme. As the sampling interval is reduced, the signal correlation will be higher.

Block diagram:



From the above diagram, we have the notations as –

- $x(n)$: sampled signal
- $x_q(n)$: quantized value of $x(n)$
- $\hat{x}_q(n)$: predicted sample of $z_q(n)$
- $d(n)$: difference sample ,i.e. difference between $x(n)$ and $\hat{x}_q(n)$
- $d_q(n)$: quantized value of $d(n)$
- $q(n)$: quantization error.

Working:

The input sampled signal $x(n)$ is sampled with very high sampling rate, then it is passed through a summer where the 1st order predicted quantized signal $\hat{x}_q(n)$ is subtracted from it. The output of the summer is $d(n)$ which is called difference sample. This $d(n)$ signal is passed through a Quantizer to get a quantized difference sample ,thus we get $d_q(n)$. Now this $d_q(n)$ is given as input is added with the with $\hat{x}_q(n)$ to get the quantized signal $x_q(n)$. This $x_q(n)$ is given as input to the Prediction filter whose output is used as a feedback .

From the above block diagram we can write

$$d(n) = x(n) - x_q(n-1) \text{ - - - - - (1)}$$

$$d_q(n) = d(n) + q(n) \text{ - - - - - (2)}$$

$$x_q(n) = d_q(n) + x_q(n-1) \text{ - - - - - (2)}$$

Accumulator circuit:

Assuming zero initial condition i.e. $x_q(-1) = 0$, from equation (3) we can write:

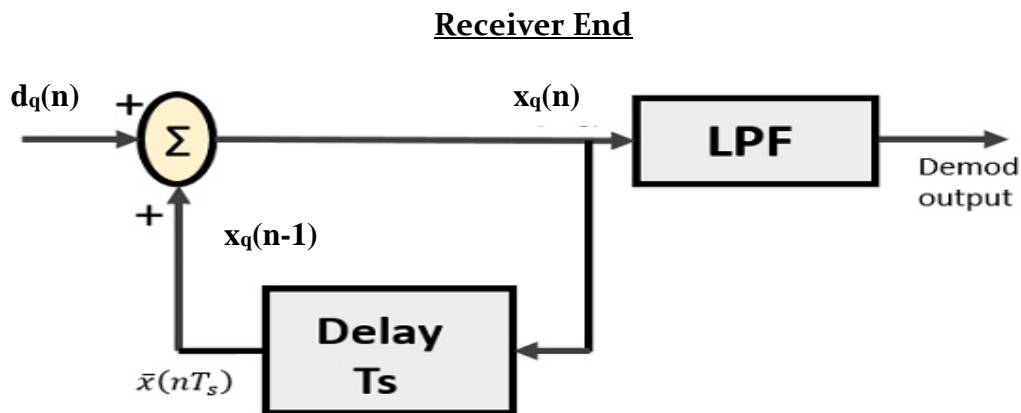
$$x_q(0) = d_q(0) + x_q(-1) = d_q(0) \text{ - - - - - (n=0)}$$

$$x_q(1) = d_q(1) + x_q(0) = d_q(1) + d_q(0) \text{ - - - - - (n=1)}$$

$$x_q(2) = d_q(2) + x_q(1) = d_q(2) + d_q(1) + d_q(0) \text{ - - - - - (n=2)}$$

$$x_q(3) = d_q(3) + x_q(2) = d_q(3) + d_q(2) + d_q(1) + d_q(0) \text{ - - - - - (n=3)}$$

From the above set of equation we can conclude that the feedback circuit for delta modulation is similar to an integrator or accumulator.



Matlab code:

```
fs=400;
t=0:1/fs:1;
f1=10;
f2=2;
x=2+3*sin(2*pi*f1*t)+1.5*cos(2*pi*f2*t);
subplot(51);
hold on;
title('original signal');
plot(t,x,'k');
hold on;
grid on;
```

```

%% Sampling
subplot(512);
stem(t,x);hold on;title('Sampled Signal');grid on;

```

```

%% 1st order prediction filter

```

```

yp=[];
for k=1:length(x)
    if k==1
        yp(k)= 0;
    else
        yp(k)=x(k-1);
    end
end
dp = x-yp;
subplot(513)
plot(t,dp);grid on;
title('1st Order Predicted DPCM');
bn=[];
z_one=[];
for n=1:length(t)
    if dp(n)>=-2 && dp(n)<-1
        b='0';
        z=-1;
    elseif dp(n)>=-1 && dp(n)<0
        b='0';
        z=-1;
    elseif dp(n)>=0 && dp(n)<1
        b='1';
        z=1;
    elseif dp(n)>=1 && dp(n)<2
        b='1';
        z=1;
    elseif dp(n)>=2 && dp(n)<3
        b='1';
        z=1;
    elseif dp(n)>=3 && dp(n)<4
        b='1';
        z=1;
    elseif dp(n)>=4 && dp(n)<5
        b='1';
        z=1;
    elseif dp(n)>=5 && dp(n)<6
        b='1';

```



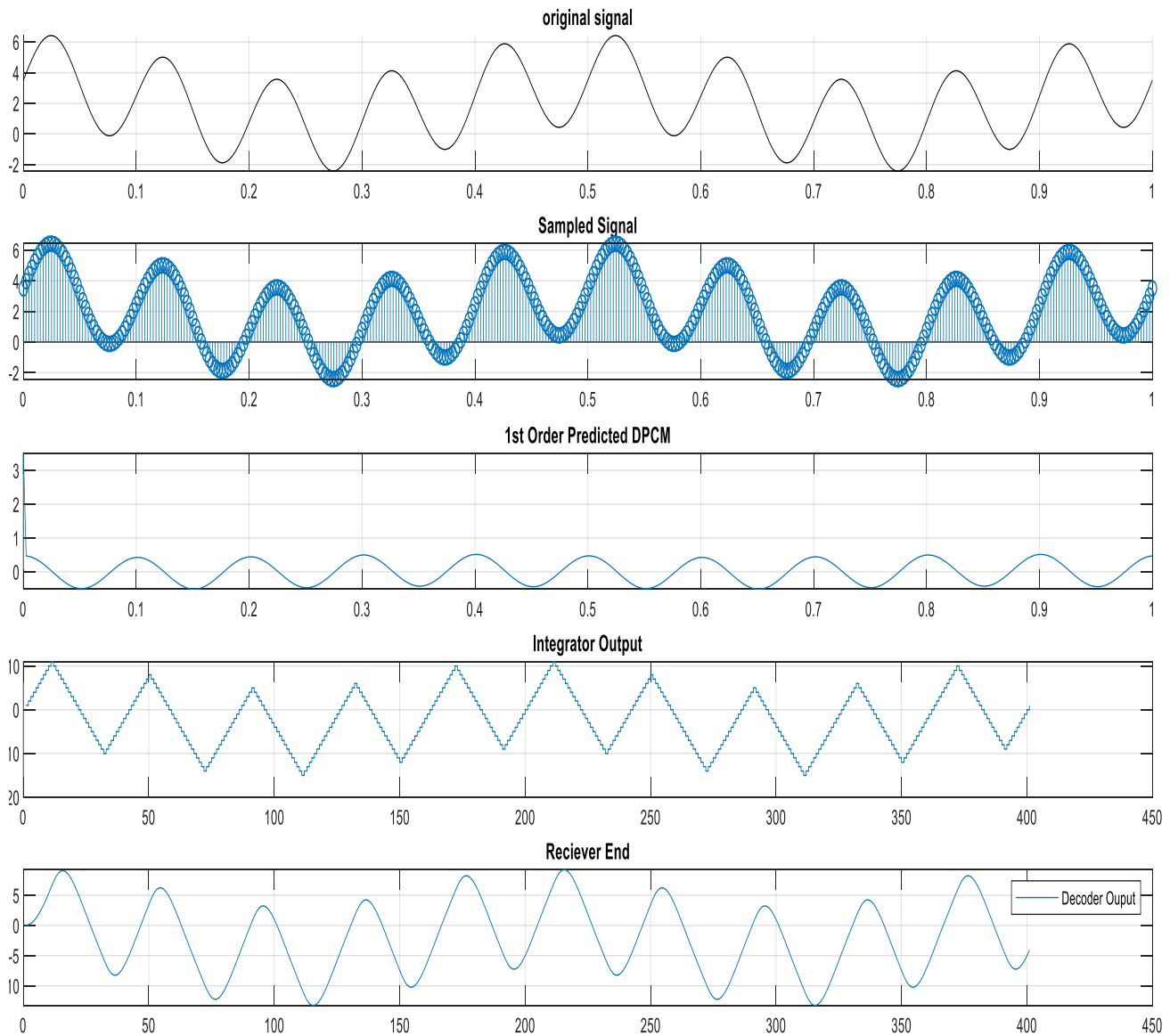
```

        z=1;
    end
    bn=[bn, b];
    z_one=[z_one, z];
end
add = [];
ad = 0;
for n = 1:length(z_one)
    if n==1
        ad = z_one(n);
    else
        ad = add(n-1)+z_one(n);
    end
    add = [add, ad];
end

t=1:length(add);
subplot(514)
stairs(add);grid on;
hold on;
title('Integrator Output');
% legend('level=2^3');
[b,a]=butter(2,0.1,'low');
x_filter = filter(b,a,add);
subplot(515)
plot(t,x_filter);grid on;
legend('Decoder Ouput');
title('Reciever End');

```

Output:



Explanation:

Figure 1: Original Signal

It depicts the original band limited signal i.e. the source data which is to be transmitted varying in between 0 to 1.

$$x(t) = 2 + 3\sin(2\pi f_1 t) + 1.5\cos(2\pi f_2 t)$$

Here (f_1, f_2) are the frequencies of the original signal.

Figure 2: Sampled Signal

It represents the sampled output of the original signal after being sampled with a very high sampling interval of $1/f_s$ where $f_s=400$.

Figure 3: 1st order DPCM Output

This graph shows the signal $x(t)$ after passing it through a 1st order prediction filter. Here we can see the amplitude of the original signal decreases significantly.

Figure 4: Integrator Output

This graph shows the received signal after passing through the integrator or accumulator. The graph is like a stair case which increase by one when 'i' is encountered and decreases by one with every 'o' encountered.

Figure 5: Receiver End

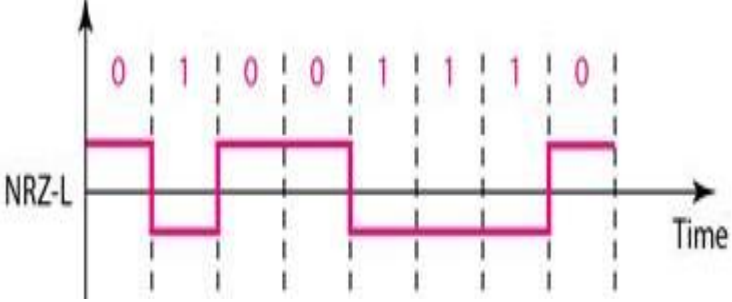
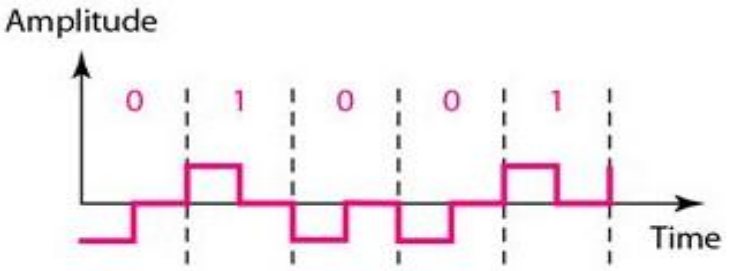
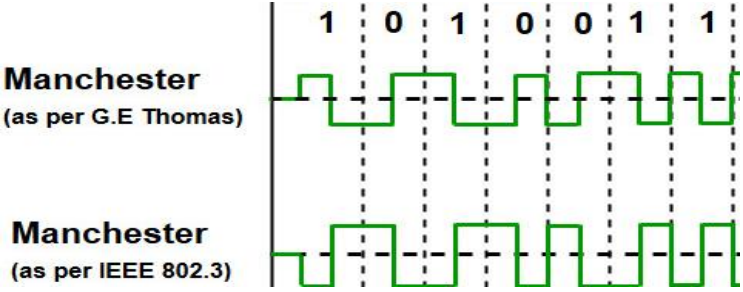
This is the final received signal obtained by passing the decoder output through a 2nd order low pass butterworth filter having normalized cutoff of 0.1π rad/sample.

4. REPRESENT THE BINARY DATA USING THE POLAR AND MANCHESTER LINE CODING TECHNIQUES, AND IMPLEMENT THE MATCHED FILTER RECEIVER FOR OBTAINING THE BINARY DATA.

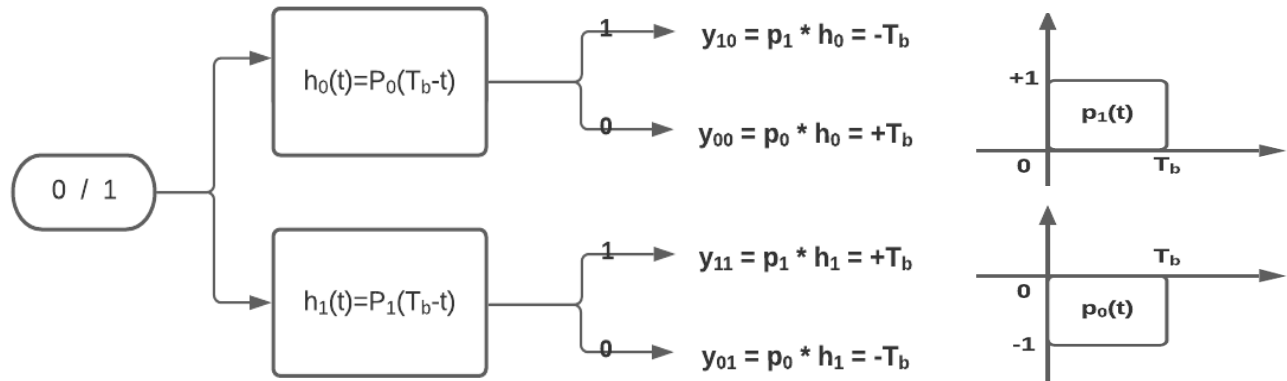
Theory:

A **line code** is the code used for data transmission of a digital signal over a transmission line. This process of coding is chosen so as to avoid overlap and distortion of signal such as inter-symbol interference.

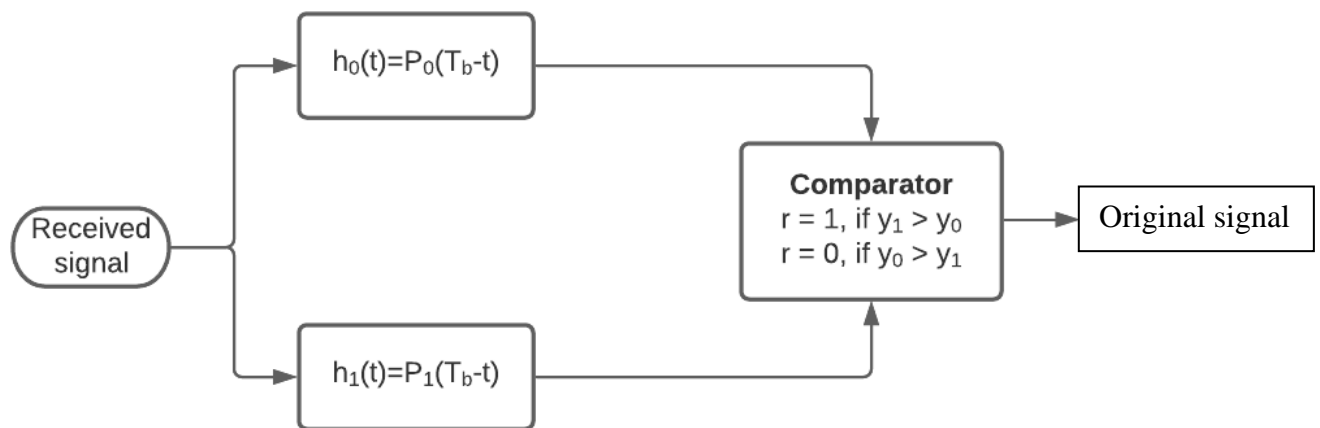
Line coding	Waveform
Polar Non-Return-Zero (NRZ)	

<p>In this type of Polar signaling, a High in data is represented by a positive pulse, while a Low in data is represented by a negative pulse. The following figure depicts this well</p>	 <p>NRZ-L</p>
<p>Polar Return-to-Zero (RZ)</p> <p>In this type of Polar signaling, a High in data, though represented by a Mark pulse, its duration T_o is less than the symbol bit duration. Half of the bit duration remains high but it immediately returns to zero and shows the absence of pulse during the remaining half of the bit duration.</p>	 <p>Amplitude</p> <p>Time</p>
<p>Manchester Line Coding</p> <p>In Manchester encoding, the duration of the bit is divided into two halves. The voltage remains at one level during the first half and moves to the other level in the second half. The transition at the middle of the bit provides synchronization.</p>	 <p>Manchester (as per G.E Thomas)</p> <p>Manchester (as per IEEE 802.3)</p>

Block diagram:



Matched filter:



Matlab Code:

Using 1st order prediction filter:

```
clc; clear;
bn = '1011101011';
ln = length(bn);
Tb = 1;
Ts = 0.001;
t = 0:Ts:Tb-Ts;
s = input('Enter case:');
switch s
    case 1 %Polar NRZ
        po = -1*ones(1,length(t));
        p1 = ones(1,length(t));
    case 2 %Polar RZ
        po = [-1*ones(1,length(t)/2), zeros(1,length(t)/2)];
        p1 = [ones(1,length(t)/2), zeros(1,length(t)/2)];
    case 3 %Manchester
```

```

        po = [-1*ones(1,length(t)/2), 1*ones(1,length(t)/2)];
        p1 = [1*ones(1,length(t)/2), -1*ones(1,length(t)/2)];
    end

    arr=[];
    xp = [];
    for n = 1:ln
        if bn(n)=='o'
            xn = po;
            a = 0;
        else
            xn = p1;
            a = 1;
        end
        xp = [xp, xn];
        arr = [arr, a];
    end

    switch s
        case 1 %Polar NRZ
            subplot(211)
            stairs(xp); hold on; axis([0 length(xp) -2 2]); grid on;
            title('Polar NRZ');
        case 2 %Polar RZ
            subplot(211)
            stairs(xp); hold on; axis([0 length(xp) -2 2]); grid on;
            title('Polar RZ');
        case 3 %Manchester
            subplot(211)
            stairs(xp); hold on; axis([0 length(xp) -2 2]); grid on;
            title('Manchester');
    end

    y = awgn(xp, 20);
    subplot(212)
    stairs(y); hold on; axis([0 length(xp) -2 2]); grid on;
    title('received signal under AWGN of 20 dB');

    ho = fliplr(po);
    h1 = fliplr(p1);
    rn = [];
    n = Tb/Ts;
    for k = Tb:ln*Tb
        for p = (k-1)*n:k*n

```

```

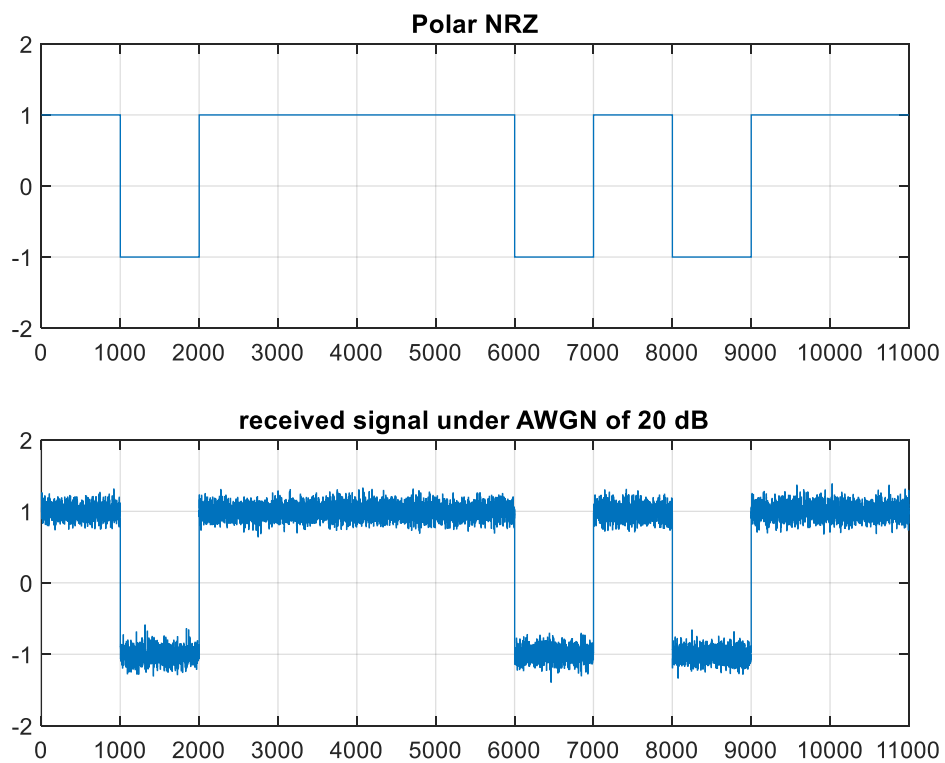
        yo = conv(y,ho);
        y1 = conv(y,h1);
    end
    if yo(k*1000) > y1(k*1000)
        rn = [rn, 0];
    else
        rn = [rn, 1];
    end
end
err = xor(arr, rn)

```

Results:

Explanation

Figure 1: Polar NRZ



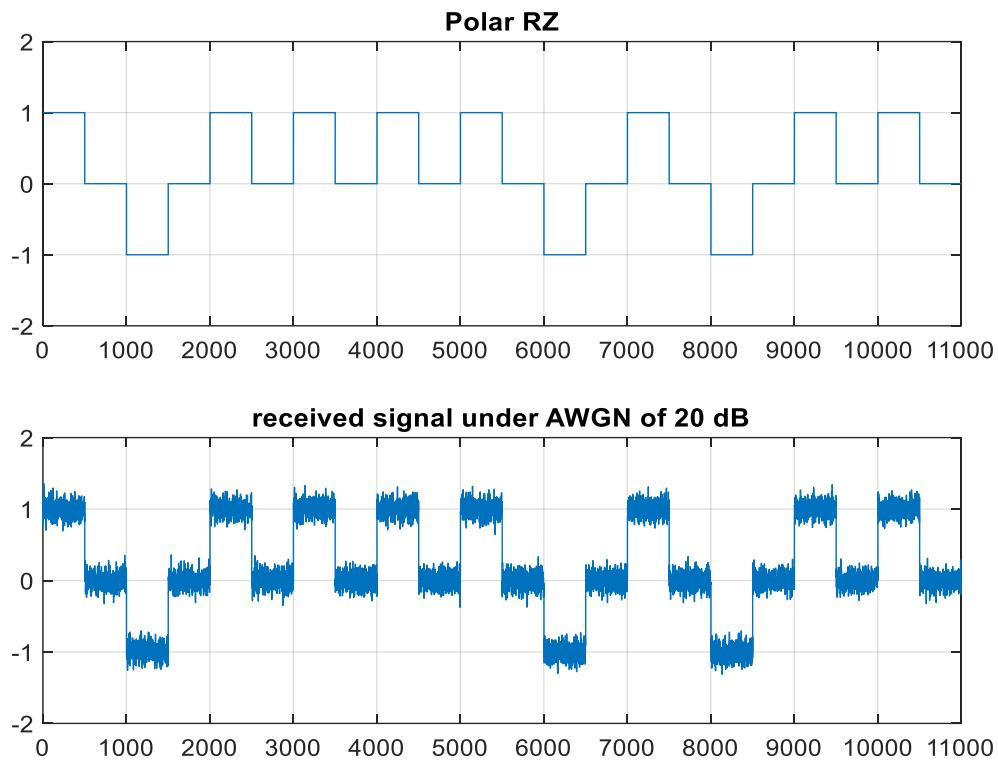
It depicts the original signal i.e. the binary data after being line coded using Polar NRZ line coding technique.

bn = 1 0 1 1 1 1 0 1 0 1 1

Figure 1: Received signal under AWGM of 20db

It represents the Polar NRZ line coded data after a noise of 20db is added to it by using awgm inbuilt function.

Figure 2: Polar RZ



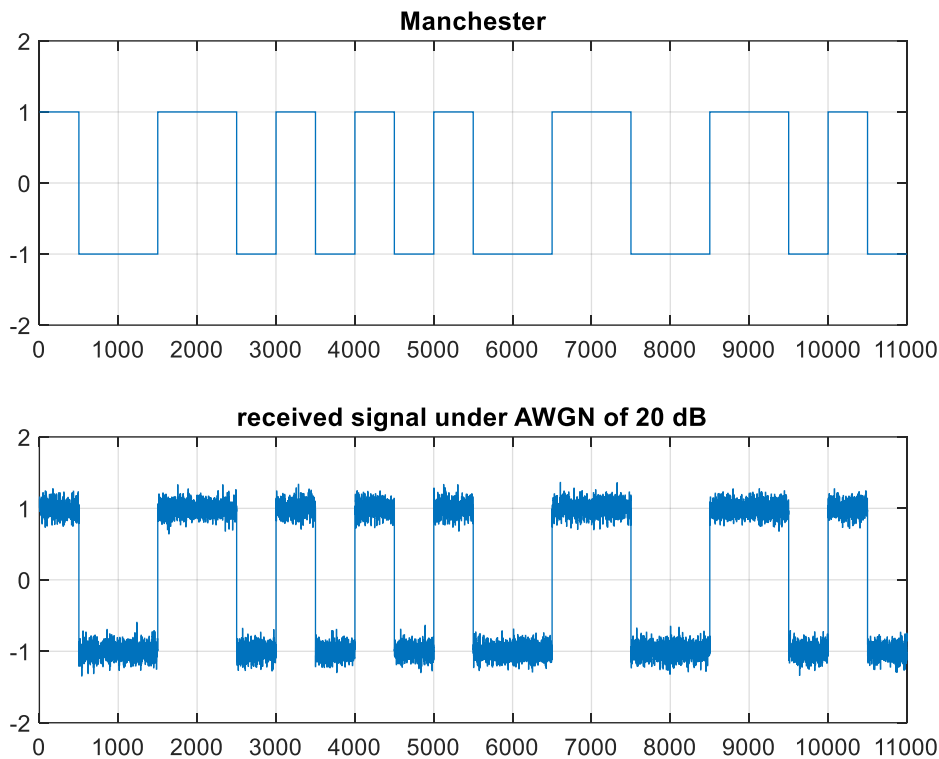
It depicts the original signal i.e. the binary data after being line coded using Polar RZ line coding technique.

$bn = 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1$
--

Figure 2: Received signal under AWGM of 20db

It represents the Polar NRZ line coded data after a noise of 20db is added to it by using awgm inbuilt function.

Figure 3: Manchester Line Coding



It depicts the original signal i.e. the binary data after being line coded using Manchester line coding technique.

bn = 1 0 1 1 1 1 0 1 0 1 1

Figure 3: Received signal under AWGM of 20db

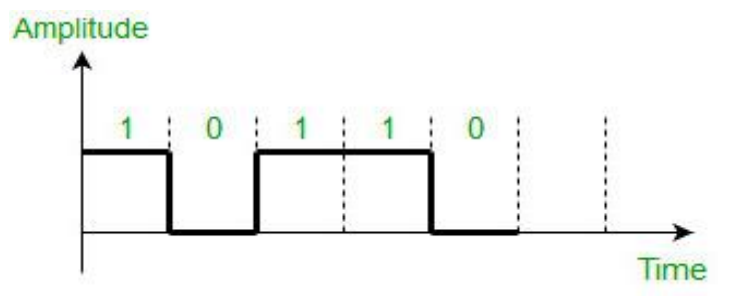
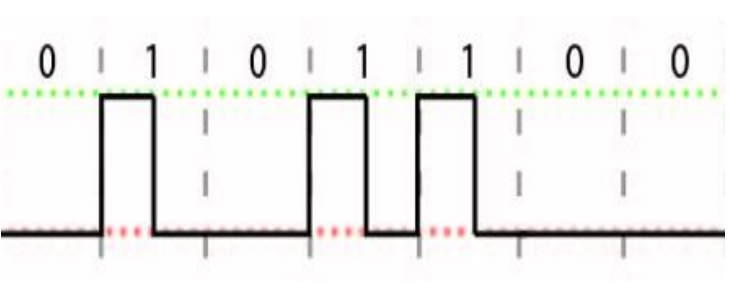
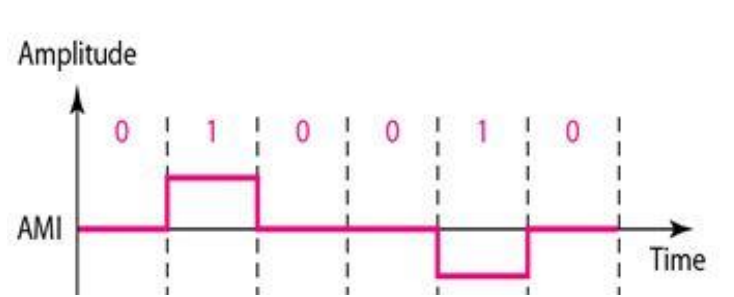
It represents the Manchester line coded data after a noise of 20db is added to it by using awgm inbuilt function.

In each of the lined coding techniques the output of the comparator was verified by performing **XOR** operation with the original binary data. The result was all zeroes which confirms that there is no deviation from the original data.

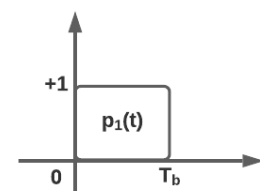
5. REPRESENT THE BINARY DATA USING THE UNIPOLAR AND BIPOLAR (AMI) LINE CODING TECHNIQUES, AND IMPLEMENT THE MATCHED FILTER RECEIVER FOR OBTAINING THE BINARY DATA.

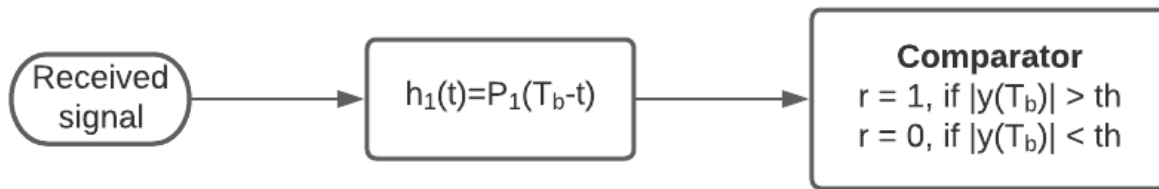
Theory:

A **line code** is the code used for data transmission of a digital signal over a transmission line. This process of coding is chosen so as to avoid overlap and distortion of signal such as inter-symbol interference.

Line coding	Waveform
<p>Unipolar Non-Return-Zero (NRZ)</p> <p>In this type of unipolar signaling, a High in data is represented by a positive pulse called as Mark, which has a duration T_0 equal to the symbol bit duration. A Low in data input has no pulse.</p>	
<p>Unipolar Return-to-Zero (RZ)</p> <p>In this type of unipolar signaling, a High in data, though represented by a Mark pulse, its duration T_0 is less than the symbol bit duration. Half of the bit duration remains high but it immediately returns to zero and shows the absence of pulse during the remaining half of the bit duration.</p>	
<p>Alternate Mark Inversion (AMI)</p> <p>For a 1, the voltage level gets a transition from + to - or from - to +, having alternate 1s to be of equal polarity. A 0 will have a zero voltage level.</p>	

Block diagram of Matched filter:





Matlab Code:

```

clc; clear;
bn = '1011101011';
ln = length(bn);
Tb = 1;
Ts = 0.001;
t = 0:Ts:Tb-Ts;
s = input('Enter case:');
switch s
    case 1 %Unipolar NRZ
        po = zeros(1,length(t));
        p1 = ones(1,length(t));
    case 2 %Unipolar RZ
        po = zeros(1,length(t));
        p1 = [ones(1,length(t)/2), zeros(1,length(t)/2)];
    case 3 %Bipolar(AMI)
        po = zeros(1,length(t));
        p1 = [ones(1,length(t)/2), zeros(1,length(t)/2)];
end

c = 0;
arr=[];
xp = [];
for n = 1:ln
    if bn(n)=='0'
        xn = po;
        a = 0;
    else
        if s == 3
            a = 1;
            if c==0
                xn = p1;

```

```

        c=c+1;
    else
        xn = -p1;
        c=c-1;
    end
else
    a = 1;
    xn = p1;
end
end
xp = [xp, xn];
arr = [arr, a];
end

switch s
case 1 %Unipolar NRZ
    subplot(211)
    stairs (xp); hold on; axis([0 length(xp) -2 2]); grid on;
    title('Unipolar NRZ');
case 2 %Unipolar RZ
    subplot(211)
    stairs (xp); hold on; axis([0 length(xp) -2 2]); grid on;
    title('Unipolar RZ');
case 3 %Bipolar(AMI)
    subplot(211)
    stairs (xp); hold on; axis([0 length(xp) -2 2]); grid on;
    title('Bipolar(AMI)');
end
y = awgn(xp, 20);
subplot(212)
stairs (y); hold on; axis([0 length(xp) -2 2]); grid on;
title('received signal under AWGN of 20 dB');

th = 10;
h = fliplr(p1);
rn = [];
n = Tb/Ts;
for k = Tb:ln*Tb
    for p = (k-1)*n:k*n
        y1 = conv(y,h);
    end
    if abs(y1(k*n)) > th
        rn = [rn, 1];
    else

```

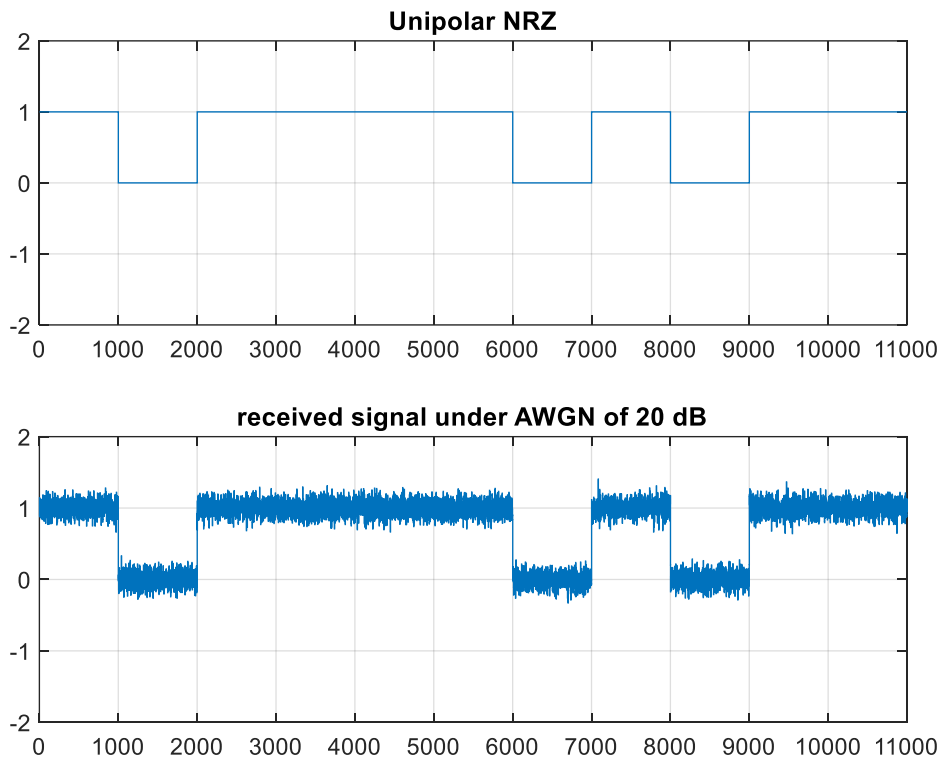
```

        rn = [rn, 0];
    end
end
err = xor(arr, rn)

```

Results & Explanation:

Figure 1: Unipolar NRZ



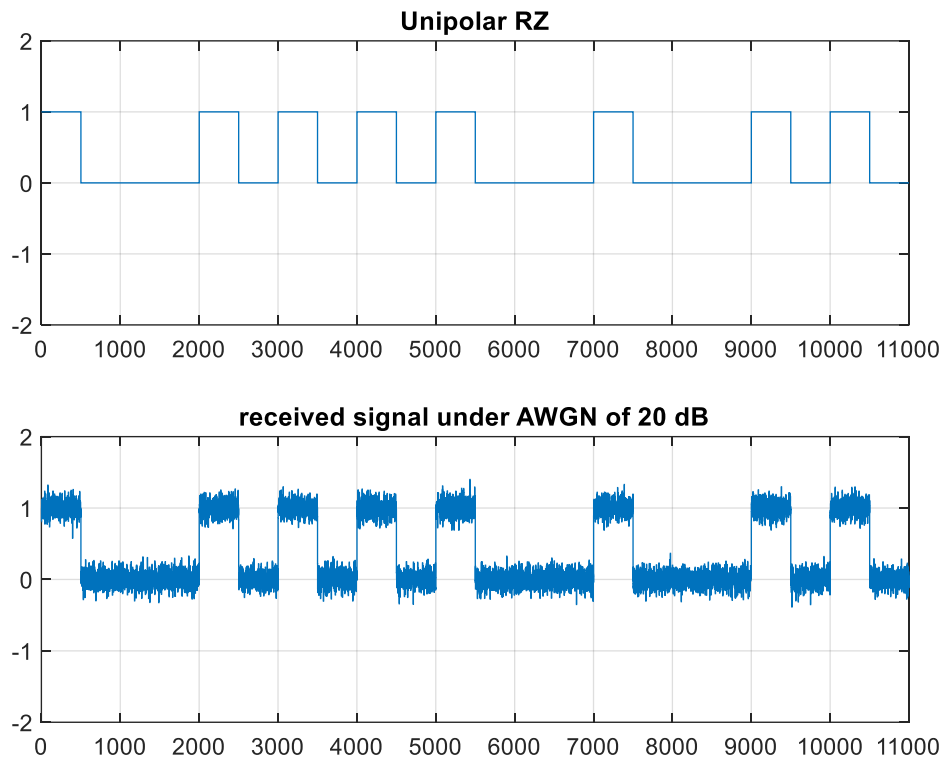
It depicts the original signal i.e. the binary data after being line coded using unipolar NRZ line coding technique.

bn = 1 0 1 1 1 1 0 1 0 1 1

Figure 1: Received signal under AWGM of 20dB

It represents the unipolar NRZ line coded data after a noise of 20dB is added to it by using awgm inbuilt function.

Figure 2: Unipolar RZ



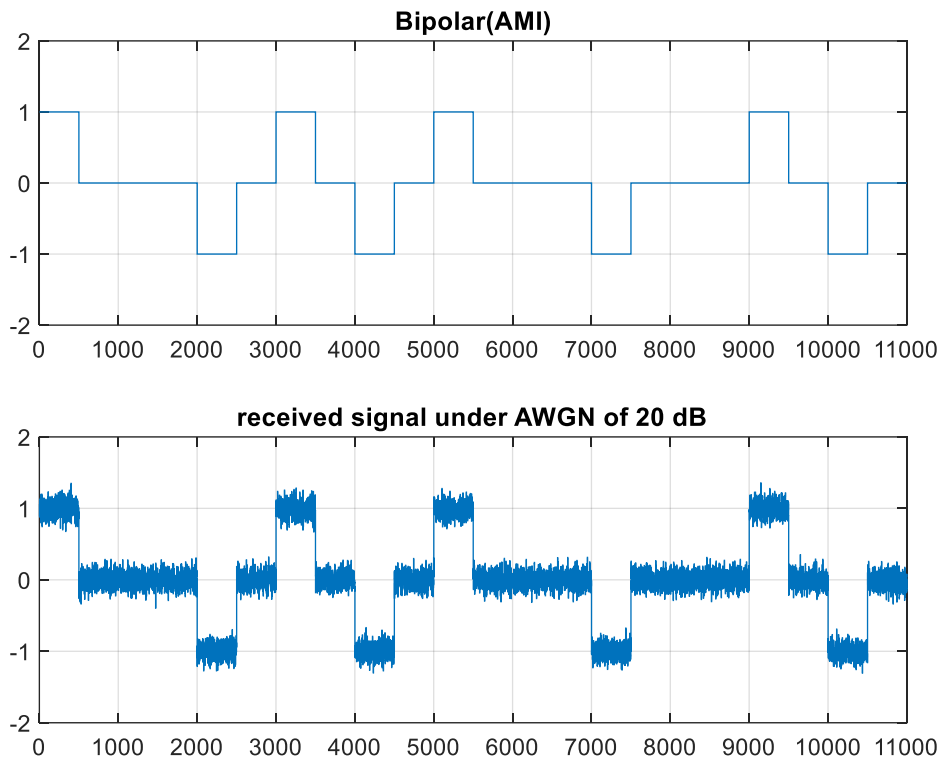
It depicts the original signal i.e. the binary data after being line coded using Unipolar RZ line coding technique.

bn = 1 0 1 1 1 1 0 1 0 1 1

Figure 2: Received signal under AWGM of 20dB

It represents the unipolar NRZ line coded data after a noise of 20dB is added to it by using awgm inbuilt function.

Figure 3: Bipolar (AMI)



It depicts the original signal i.e. the binary data after being line coded using Bipolar line coding technique.

bn = 1 0 1 1 1 1 0 1 0 1 1

Figure 3: Received signal under AWGM of 20dB

It represents the Bipolar line coded data after a noise of 20dB is added to it by using awgm inbuilt function.

In each of the lined coding techniques the output of the comparator was verified by performing **XOR** operation with the original binary data. The result was all zeroes which confirms that there is no deviation from the original data.

6. AMPLITUDE SHIFT KEYING MODULATOR AND DEMODULATOR

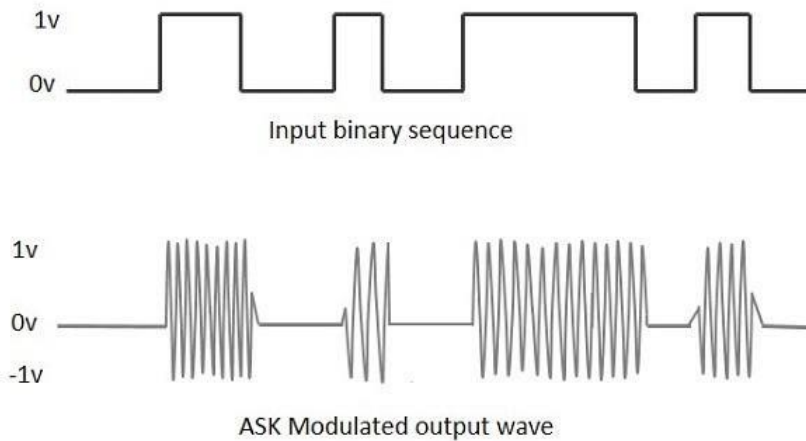
Theory:

Amplitude Shift Keying ASK is a type of Amplitude Modulation which represents the binary data in the form of variations in the amplitude

of a signal.

Any modulated signal has a high frequency carrier. The binary signal when ASK modulated, gives a zero value for Low input while it gives the carrier output for High input.

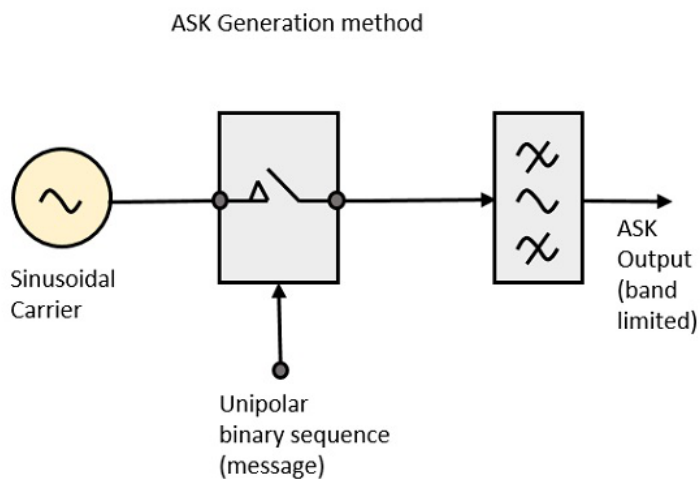
The following figure represents ASK modulated waveform along with its input.



ASK Modulator:

The ASK modulator block diagram comprises of the carrier signal generator, the binary sequence from the message signal and the band-

limited filter. Following is the block diagram of the ASK Modulator.



The carrier generator, sends a continuous high-frequency carrier. The binary sequence from the message signal makes the unipolar input to be either High or Low. The high signal closes the switch, allowing a carrier wave. Hence, the output will be the carrier signal at high input. When there is low input, the switch opens, allowing no voltage to appear. Hence, the output will be low. The band-limiting filter, shapes the pulse depending upon the amplitude and phase characteristics of the band-limiting filter or the pulse-shaping filter.

ASK Demodulator:

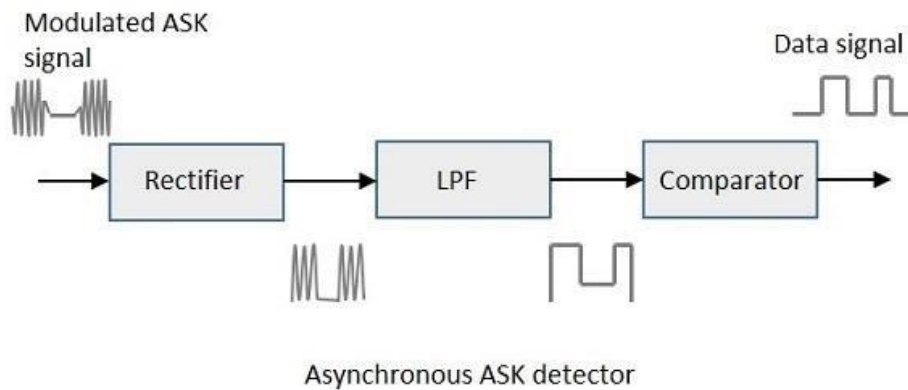
There are two types of ASK Demodulation techniques. They are:

- a. Asynchronous ASK Demodulation/detection
- b. Synchronous ASK Demodulation/detection

The clock frequency at the transmitter when matches with the clock frequency at the receiver, it is known as a Synchronous method, as the frequency gets synchronized. Otherwise, it is known as Asynchronous.

Asynchronous ASK Demodulator:

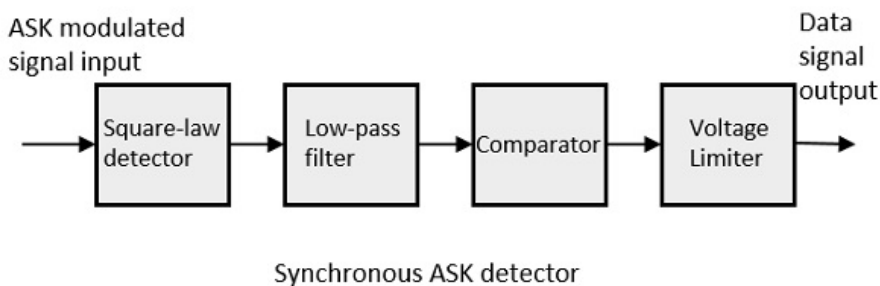
The Asynchronous ASK detector consists of a half-wave rectifier, a low pass filter, and a comparator. Following is the block diagram for the same.



The modulated ASK signal is given to the half-wave rectifier, which delivers a positive half output. The low pass filter suppresses the higher frequencies and gives an envelope detected output from which the comparator delivers a digital output.

Synchronous ASK Demodulator:

Synchronous ASK detector consists of a Square law detector, low pass filter, a comparator, and a voltage limiter. Following is the block diagram for the same.



The ASK modulated input signal is given to the Square law detector. A square law detector is one whose output voltage is proportional to the square of the amplitude modulated input voltage. The low pass filter minimizes the higher frequencies. The comparator and the voltage limiter help to get a clean digital output.

MATLAB code:

```

%% Digital input
clc;
clear all;
bn = '1100101';
len=length(bn);
tb=1;
ts=0.001;
t=0:ts:tb-ts;
fc=6;
ct=cos(2*pi*fc*t);
ct1 = cos(2*pi*fc*(0:ts:len*tb-ts));
subplot(511);title('sample signal'); hold on;
plot(0:ts:len*tb-ts,ct1,'b'); grid on; axis([0, len, -2, 2]);

```

```

%% ASK Modulation
xp=[];
for n=1:len
    if bn(n)=='0'
        xn=0.*ct;
    else
        xn=1.*ct;
    end
    xp=[xp, xn];
end
subplot(512);title('ASK Modulation'); hold on;
plot(0:ts:len*tb-ts,xp,'b'); grid on; axis([0, len, -2, 2]);

```

```

%% Gaussian Noise addition
xr=awgn(xp,20);
subplot(513);title('Noise added signal'); hold on;
plot(0:ts:len*tb-ts,xr); grid on; axis([0, len, -2, 2]);

```

```

%% ASK Demodulation
ask_thld=0.25;
dem=[];
res=[];
for i=1:len
    n=tb/ts;
    j=(i-1)*n;
    yr=xr(j+1:j+n).*ct;
    dem =[dem, yr];
    [b,a]=butter(2,0.5,'low');
    ask_filter = filter(b,a,dem);

```

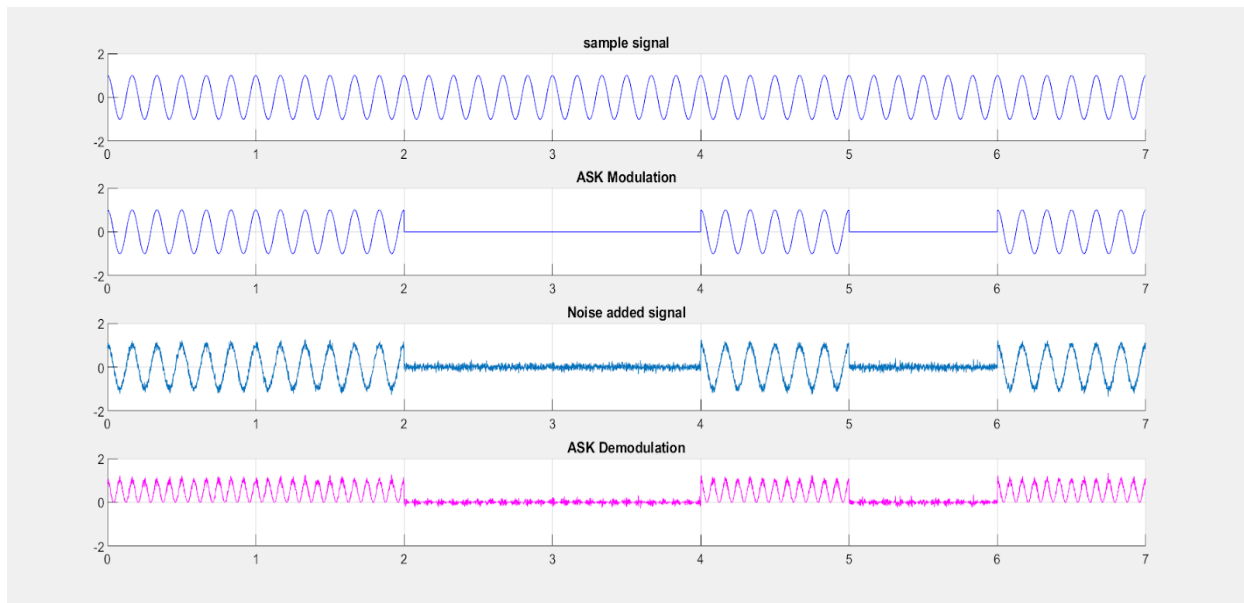
```

y_out=mean(ask_filter);
if y_out(1)>=ask_thld
    y(i)='1';
else
    y(i)='0';
end
res=[res; i, y_out];
end

subplot(514);title('ASK Demodulation'); hold on;
plot(o:ts:len*tb-ts,dem,'m'); grid on; axis([o, len, -2, 2]);

y
res
err=xor(bn,y)

```



Result:

```
Command Window

y =

    '1111111'

res =

    1.0000    0.4970
    2.0000    0.4993
    3.0000    0.3343
    4.0000    0.2514
    5.0000    0.3012
    6.0000    0.2511
    7.0000    0.2866

err =

    1x7 logical array

    0     0     0     0     0     0     0
```

In this experiment we performed Amplitude shift keying (ASK) modulation and demodulation, ASK represents the binary data in the form of variations in the amplitude of a signal.

Subplot 1 in result diagram shows the sample signal.

Subplot 2 in result diagram shows the ASK modulated signal which is formed by merging the UNRZ (Unipolar non return to zero) output with carrier signal.

Subplot 3 in result diagram shows the ASK signal after Additive White Gaussian Noise (AWGN) of 20dB is added to it.

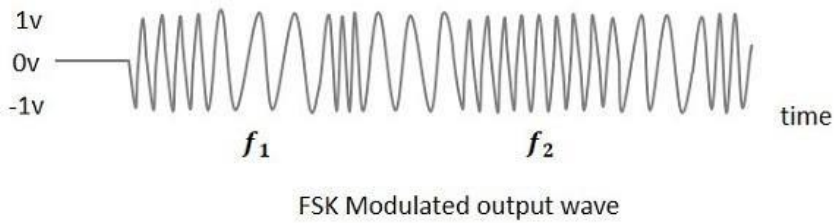
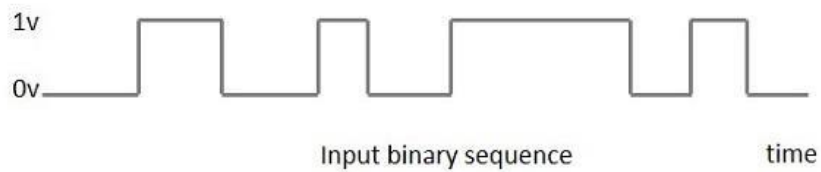
Subplot 4 in result diagram shows the ASK demodulated signal which is formed after ASK signal with AWGN noise is merged to carrier signal.

Command window shows that after comparing the demodulated values with the threshold, we get the reconstructed output which is equal to the binary input taken initially.

7. FREQUENCY SHIFT KEYING MODULATOR AND DEMODULATOR

Theory: Frequency Shift Keying (FSK) is the digital modulation technique in which the frequency of the carrier signal varies according to the digital signal changes. FSK is a scheme of frequency modulation.

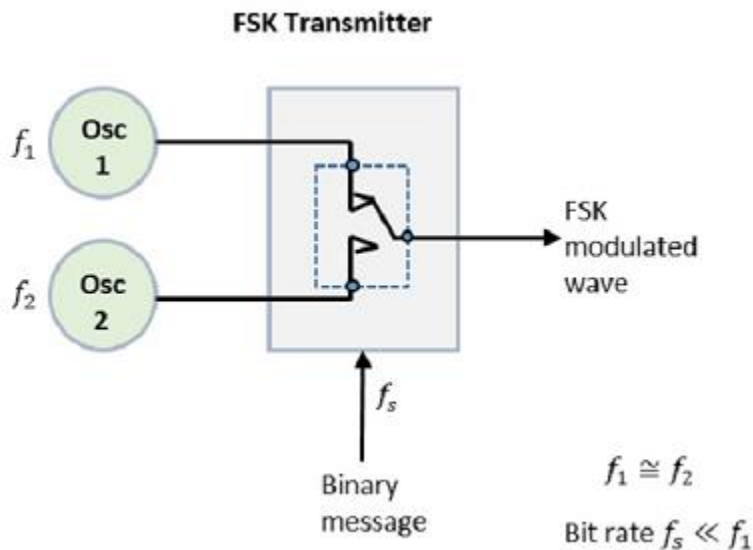
The output of a FSK modulated wave is high in frequency for a binary High input and is low in frequency for a binary Low input. The binary 1s and 0s are called Mark and Space frequencies.



To find the process of obtaining this FSK modulated wave, let us know about the working of a FSK modulator.

FSK Modulator:

The FSK modulator block diagram comprises of two oscillators with a clock and the input binary sequence. Following is its block diagram.



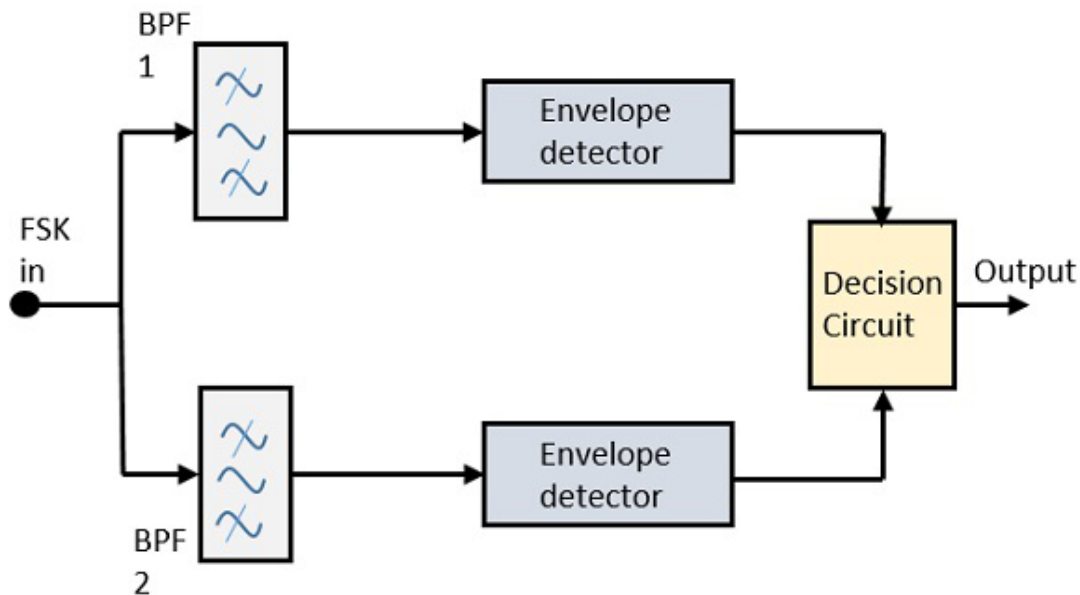
The two oscillators, producing a higher and a lower frequency signals, are connected to a switch along with an internal clock. To avoid the abrupt phase discontinuities of the output waveform during the transmission of the message, a clock is applied to both the oscillators, internally. The binary input sequence is applied to the transmitter so as to choose the frequencies according to the binary input.

FSK Demodulator:

There are different methods for demodulating a FSK wave. The main methods of FSK detection are asynchronous detector and synchronous detector. The synchronous detector is a coherent one, while asynchronous detector is a non-coherent one.

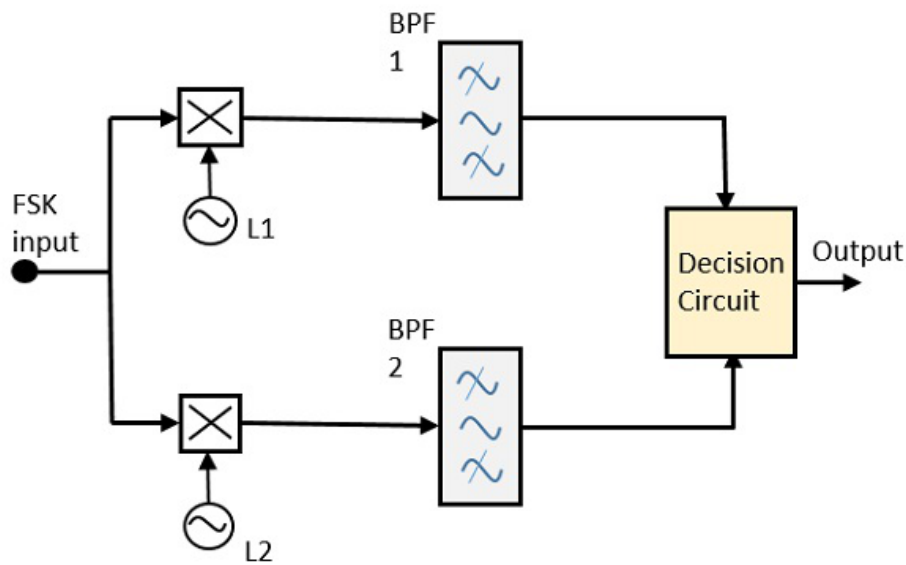
Asynchronous FSK Detector:

The block diagram of Asynchronous FSK detector consists of two band pass filters, two envelope detectors, and a decision circuit. Following is the diagrammatic representation.



The FSK signal is passed through the two Band Pass Filters BPFs, tuned to Space and Mark frequencies. The output from these two BPFs look like ASK signal, which is given to the envelope detector. The signal in each envelope detector is modulated asynchronously. The decision circuit chooses which output is more likely and selects it from any one of the envelope detectors. It also re-shapes the waveform to a rectangular one.

Synchronous FSK Detector: The block diagram of Synchronous FSK detector consists of two mixers with local oscillator circuits, two band pass filters and a decision circuit. Following is the diagrammatic representation.



The FSK signal input is given to the two mixers with local oscillator circuits. These two are connected to two band pass filters. These combinations act as demodulators and the decision circuit chooses which output is more likely and selects it from any one of the detectors. The two signals have a minimum frequency separation.

MATLAB code:

```
%% Digital input
```

```
clc;
```

```
clear all;
```

```
bn = '1100101';
```

```
len=length(bn);
```

```
tb=1;
```

```
ts=0.001;
```

```
t=0:ts:tb-ts;
```

```
fc1=5;
```

```
fc2=8;
```

```
ct1=sin(2*pi*fc1*t);
```

```
ct2=sin(2*pi*fc2*t);
```

```
%% FSK Modulation
```

```
xp=[];
```

```
for n=1:len
```

```
    if bn(n)=='o'
```

```
        xn=1.*ct1;
```

```
    else
```



```

        xn=1.*ct2;
    end
    xp=[xp, xn];
end
subplot(411);title('FSK Modulation '); hold on;
plot(o:ts:len*tb-ts,xp,'b'); grid on; axis([o, len, -2, 2]);

%% Gaussian Noise Addition
xr=awgn(xp,17);
subplot(412);title('Noise added signal'); hold on;
plot(o:ts:len*tb-ts,xr); grid on;

%% FSK Demodulation
fsk_thld=0.25;
dem1=[];
dem2=[];
res=[];
for i=1:len
    n=tb/ts;
    j=(i-1)*n;
    yr1=xr(j+1:j+n).*ct1;
    yr2=xr(j+1:j+n).*ct2;
    dem1=[dem1, yr1];
    dem2=[dem2, yr2];

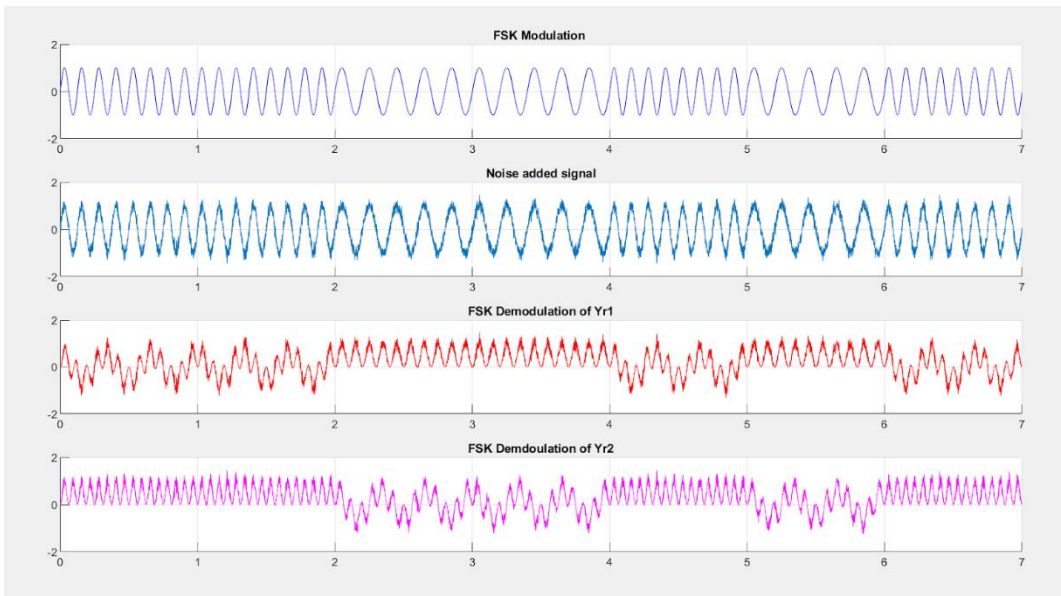
    y_out=[mean(yr1),mean(yr2)];
    if y_out(1)>=fsk_thld
        y(i)='0';
    else
        y(i)='1';
    end
    res=[res; i, y_out];
end

subplot(413);title('FSK Demodulation of Yr1'); hold on;
plot(o:ts:len*tb-ts,dem1,'r'); grid on; axis([o, len, -2, 2]);
subplot(414);title('FSK Demdoulation of Yr2'); hold on;
plot(o:ts:len*tb-ts,dem2,'m'); grid on; axis([o, len, -2, 2]);

y
res
err=xor(bn,y)

```

Result:



Command Window

```
y =
    '1100101'

res =

    1.0000    -0.0025     0.4923
    2.0000     0.0026     0.5008
    3.0000     0.5043    -0.0005
    4.0000     0.5047     0.0050
    5.0000    -0.0007     0.5007
    6.0000     0.4967     0.0031
    7.0000     0.0013     0.4989

err =

    1x7 logical array

    0     0     0     0     0     0     0
```

In this experiment we performed Frequency shift keying (FSK) modulation and demodulation. In FSK, the frequency of the carrier signal varies according to the digital signal changes.

Subplot 1 shows the FSK modulated signal which is formed by merging a positive pulse of amplitude +1 with the carrier signal.

Subplot 2 shows the FSK signal after Additive White Gaussian Noise (AWGN) of 17dB is added to it.

Subplots 3 and 4 shows the FSK demodulated signals which is formed after FSK signal with AWGN noise is merged with both of the carrier signals.

Command window shows that after comparing the demodulated values with the threshold, we get the reconstructed output which is equal to the binary input taken initially.

8. PHASE SHIFT KEYING MODULATOR AND DEMODULATOR

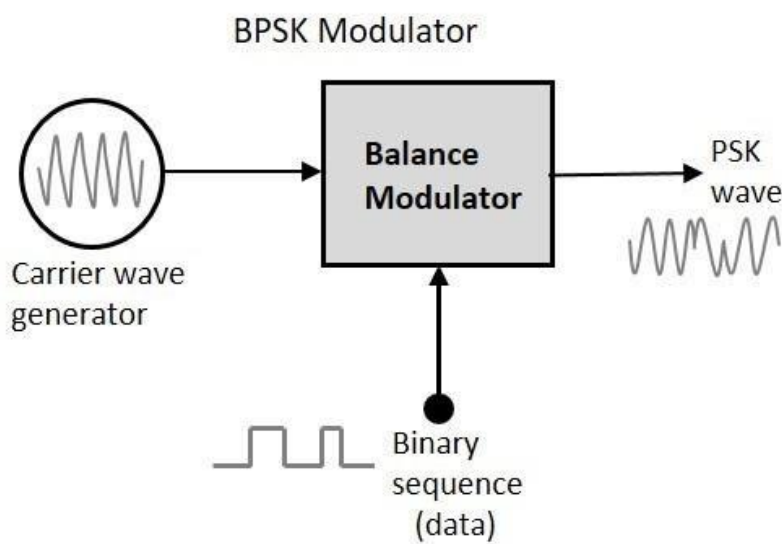
Theory: Phase Shift Keying PSK is the digital modulation technique in which the phase of the carrier signal is changed by varying the sine and cosine inputs at a particular time. PSK technique is widely used for wireless LANs, bio-metric, contactless operations, along with RFID and Bluetooth communications.

Binary Phase Shift Keying (BPSK):

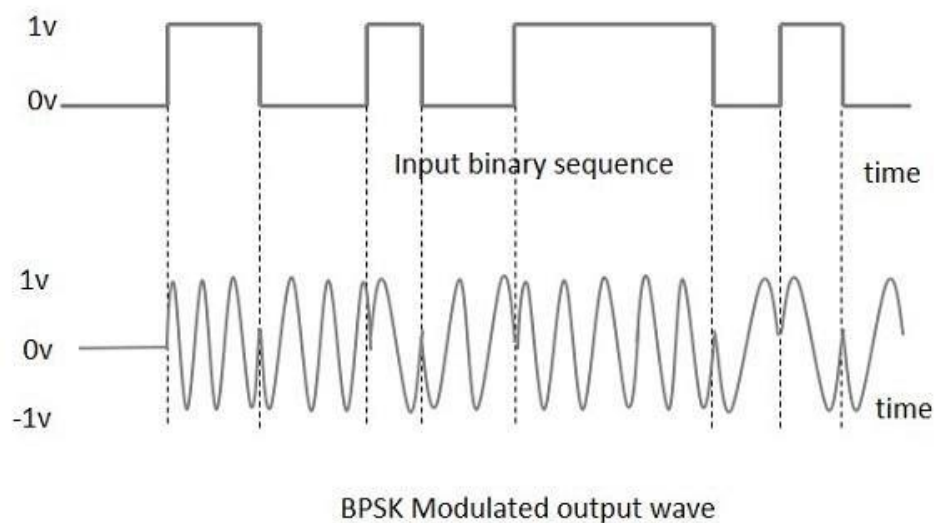
This is also called as 2-phase PSK or Phase Reversal Keying. In this technique, the sine wave carrier takes two phase reversals such as 0° and 180° .

BPSK is basically a Double Side Band Suppressed Carrier DSBSC modulation scheme, for message being the digital information.

The block diagram of Binary Phase Shift Keying consists of the balance modulator which has the carrier sine wave as one input and the binary sequence as the other input.



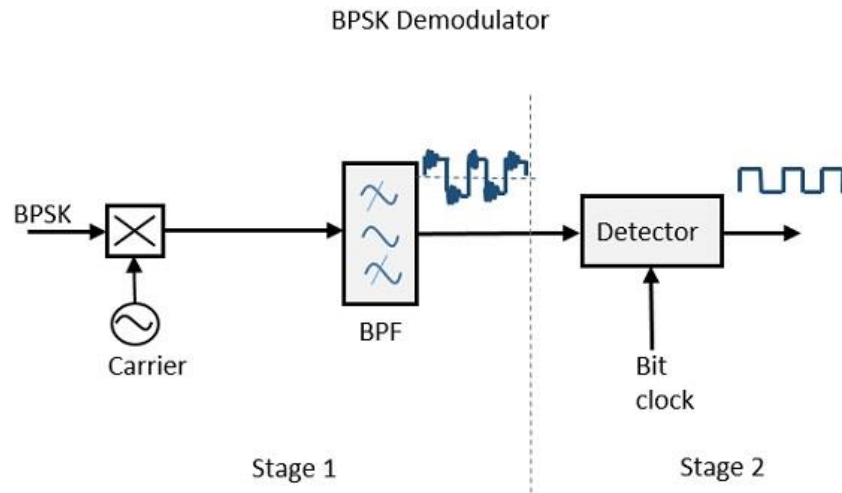
The modulation of BPSK is done using a balance modulator, which multiplies the two signals applied at the input. For a zero binary input, the phase will be 0° and for a high input, the phase reversal is of 180° . Following is the diagram of BPSK modulated output wave:



The output sine wave of the modulator will be the direct input carrier or the inverted 180° phase shifted input carrier, which is a function of the data signal.

BPSK Demodulator:

The block diagram of BPSK demodulator consists of a mixer with local oscillator circuit, a bandpass filter, a two-input detector circuit.



By recovering the band-limited message signal, with the help of the mixer circuit and the band pass filter, the first stage of demodulation gets completed. The base band signal which is band limited is obtained and this signal is used to regenerate the binary message bit stream.

In the next stage of demodulation, the bit clock rate is needed at the detector circuit to produce the original binary message signal. If the bit rate is a sub-multiple of the carrier frequency, then the bit clock regeneration is simplified. To make the circuit easily understandable, a decision-making circuit may also be inserted at the 2nd stage of detection.

MATLAB code:

```
%% Digital input
clc;
clear all;
bn = '1100101';
len=length(bn);
tb=1;
ts=0.001;
t=0:ts:tb-ts;
fc=5;
ct=cos(2*pi*fc*t);
```

```
%% PSK Modulation
xp=[];
for n=1:len
```

```

    if bn(n)=='o'
        xn=-1.*ct;
    else
        xn=1.*ct;
    end
    xp=[xp, xn];
end
subplot(311);title('PSK Modulation '); hold on;
plot(0:ts:len*tb-ts,xp,'b'); grid on; axis([0, len, -2, 2]);

%% Gaussian Noise addition
xr=awgn(xp,15);
subplot(312);title('Noise added signal'); hold on;
plot(0:ts:len*tb-ts,xr); grid on; axis([0, len, -2, 2]);
% xr = xp;

%% PSK Demodulation
psk_thld=0.25;
dem=[];
res=[];
for i=1:len
    n=tb/ts;
    j=(i-1)*n;
    yr=xr(j+1:j+n).*ct;
    dem =[dem, yr];
    [b,a]=butter(2,0.5,'low');
    psk_filter = filter(b,a,dem);

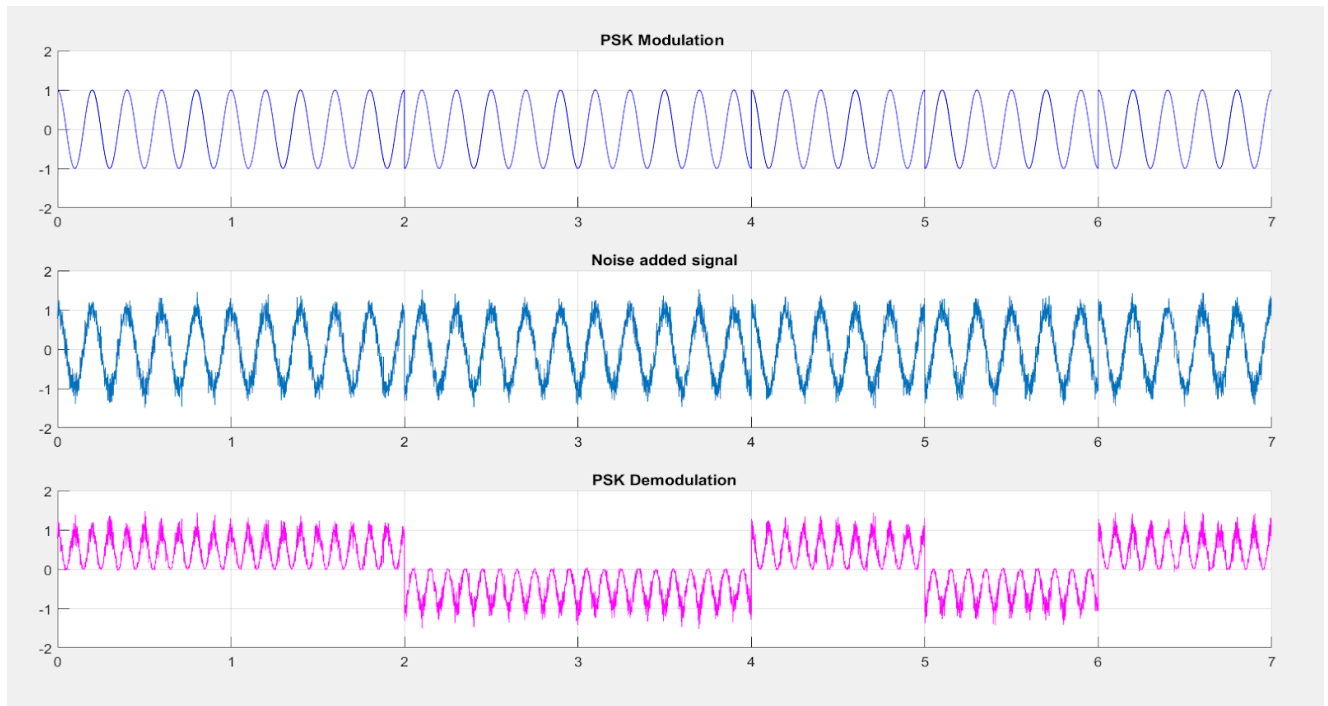
    y_out=mean(psk_filter);
    if y_out(1)>=psk_thld
        y(i)='1';
    else
        y(i)='0';
    end
    res=[res; i, y_out];
end

subplot(313);title('PSK Demodulation'); hold on;
plot(0:ts:len*tb-ts,dem,'m'); grid on; axis([0, len, -2, 2]);

y
res
err=xor(bn,y)

```

Result:



```
Command Window

y =

    '1100000'

res =

    1.0000    0.4996
    2.0000    0.4981
    3.0000    0.1673
    4.0000    0.0013
    5.0000    0.1010
    6.0000    0.0006
    7.0000    0.0722

err =

    1x7 logical array

    0     0     0     0     0     0     0
```

In this experiment we performed the Phase Shift Keying (PSK) modulation and demodulation. In PSK, the phase of the carrier signal is changed by varying the sine and cosine inputs at a particular time.

Subplot 1 shows the PSK modulated signal which is formed by merging the polar line coded output with carrier signal.

Subplot 2 shows the PSK signal after Additive White Gaussian Noise (AWGN) of 15dB is added to it.

Subplot 3 shows the PSK demodulated signal which is formed after PSK signal with AWGN is merged with carrier signal.

Command window shows that after comparing the demodulated values with the threshold, we get the reconstructed output which is equal to the binary input taken initially.

Submitted by:-

Sandeep ray

B218063

Etc, 5th sem.