**BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY**


**Department of Electrical and Electronic Engineering**

**Course No. :** EEE 438 **Group :** G1

**Course Title:** Wireless Communication Laboratory

**Experiment No.** **:** 01

**Name of the Experiment** **:** Basic Digital Modulation Techniques

**Name:** Md Awsafur Rahman **ID:** 1706066

**Level:** 4 **Term:** 2

**Date of Performance:** 30/11/2022

**Date of Submission:** 06/12/2022

**Lab Task 1:**

**(a)**

**BASK:**

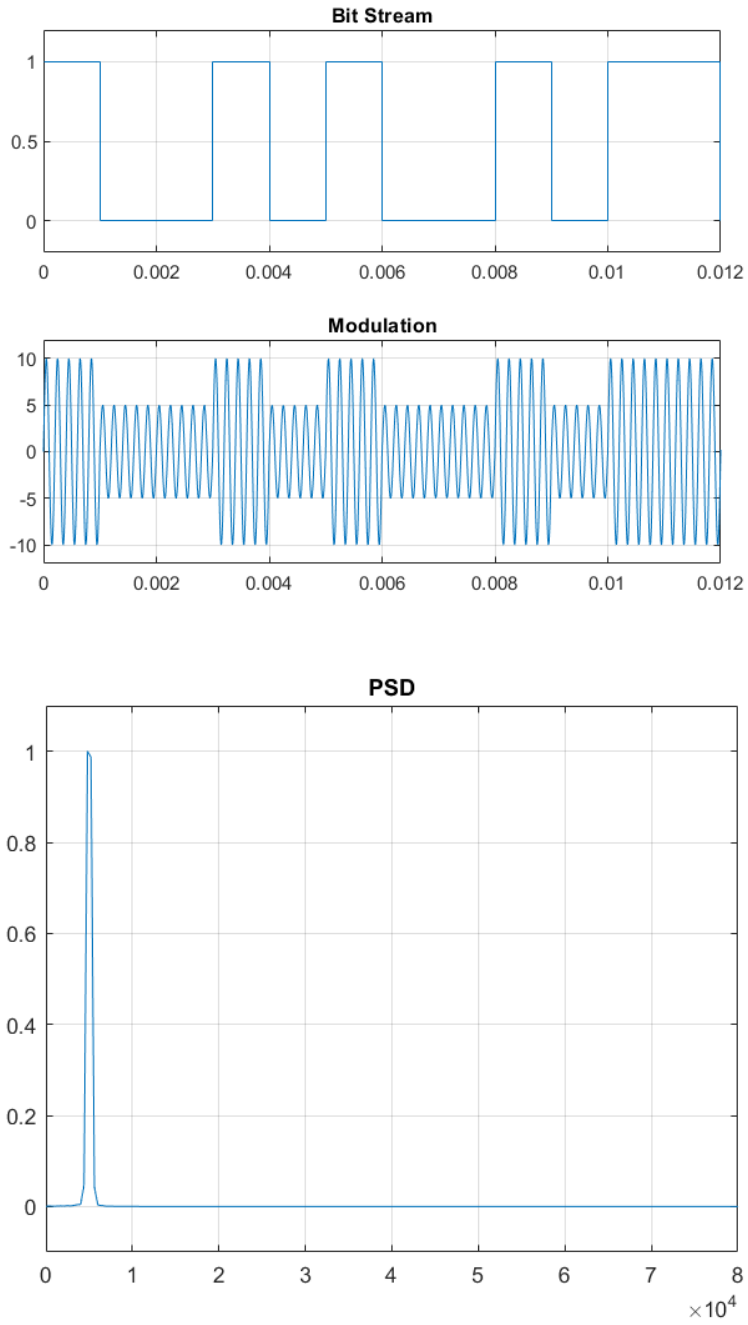



Fig: BASK modulation and power spectrum

## Code snippet for BASK modulation:

```
clc;
clear all;
close all;
N=5000; % number of points
A0=5; % symbol for 0
A1=10; % symbol for 1
fc=5e3; % carrier frequence
Rb = 1000; % bit rate (num_bits in 1s)
Tb = 1/Rb; % bit duration (time of 1 bit)
Fs = 40*fc; % sampling frequence
Ts = 1/Fs; % sampling time
k = Fs*Tb; % number of samples in 1 bit


bits = randi([0,1], 1, N);
bits = repelem(bits, k); % replicate bit to generate actual signal
t= 0:Ts:N*Tb-Ts; %linspace(0, 0.5, N*k);

figure;

% bitstream
subplot(211);
stairs(t,bits);
xlim([0, 0.02]);
ylim([-0.2, 1.2]);
title("Bit Stream")
grid on;

% modulated
cbits = 1.-bits;
y = bits.*(A1*sin(2*pi*fc*t)) + cbits.*(A0*sin(2*pi*fc*t));
subplot(212);
plot(t, y);
xlim([0, 0.02]);
ylim([-12, 12]);
title("Modulation")
grid on;

figure;
[pxx f] = pwelch(y, 500, 200, 500, Fs);
pxx = pxx/max(pxx);
plot(f, pxx);
xlim([0, 80000]);
ylim([-0.1, 1.1]);
title("PSD")
grid on;
```

BASK modulated signal has two amplitude levels for two symbols. The power spectrum only exists at the carrier frequency (fc).

**BFSK:**

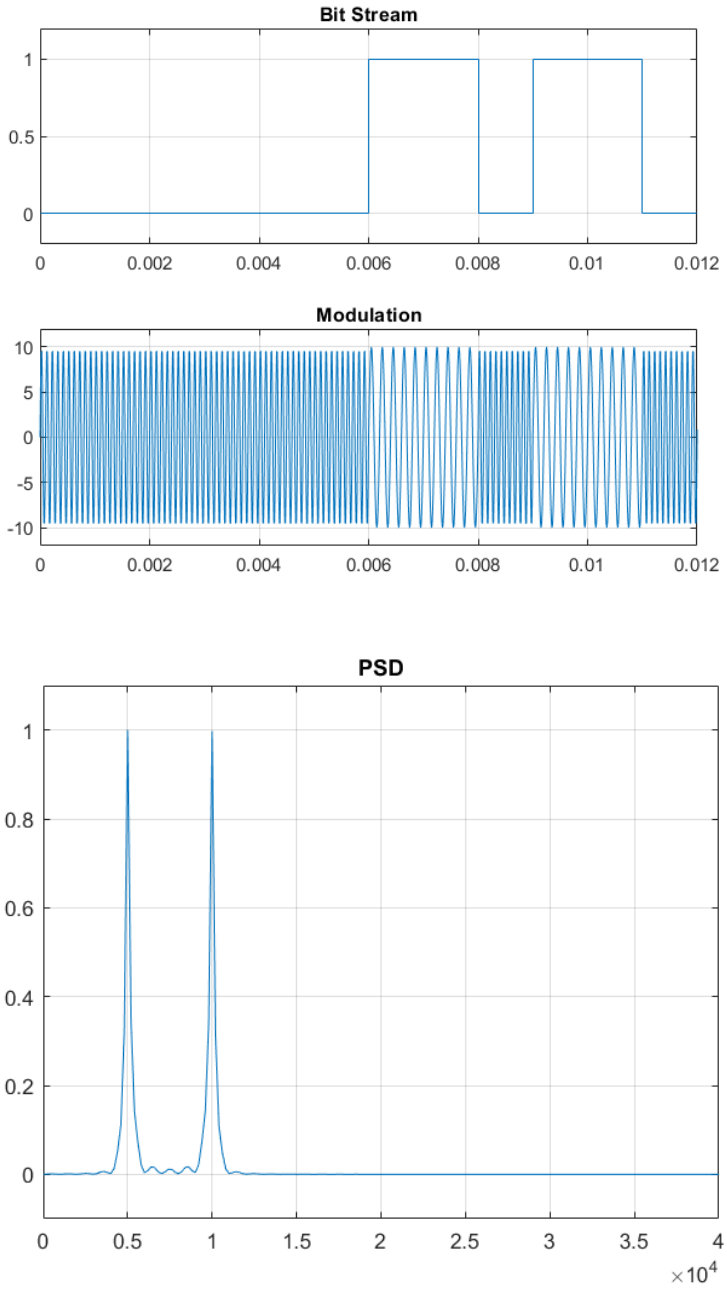



**Fig:** Modulated signal and power spectral density for BFSK modulation

## Code snippet for BFSK modulation:

```
clc;
clear all;
close all;
N=5000;
A0=10;
A1=10;
fc1=5e3;
fc2=2*fc1;
Rb = 1000;
Tb = 1/Rb;
Fs = 20*fc1;
Ts = 1/Fs;
k = Fs*Tb;


bits = randi([0,1], 1, N);
bits = repelem(bits, k);
t= 0:Ts:N*Tb-Ts; %linspace(0, 0.5, N*k);


figure;
subplot(211);
stairs(t,bits)
xlim([0, 0.012]);
ylim([-0.2, 1.2]);
title("Bit Stream")
grid on;


cbits = 1.-bits;
y = bits.*(A1*sin(2*pi*fc1*t)) + cbits.*(A0*sin(2*pi*fc2*t));
subplot(212);
plot(t, y);
xlim([0, 0.012]);
ylim([-12, 12]);
title("Modulation")
grid on;

figure;
[pxx f] = pwelch(y, 500, 200, 500, Fs);
pxx = pxx/max(pxx);
plot(f, pxx);
xlim([0, 40000]);
ylim([-0.1, 1.1]);
title("PSD")
grid on;
```

**BPSK:**

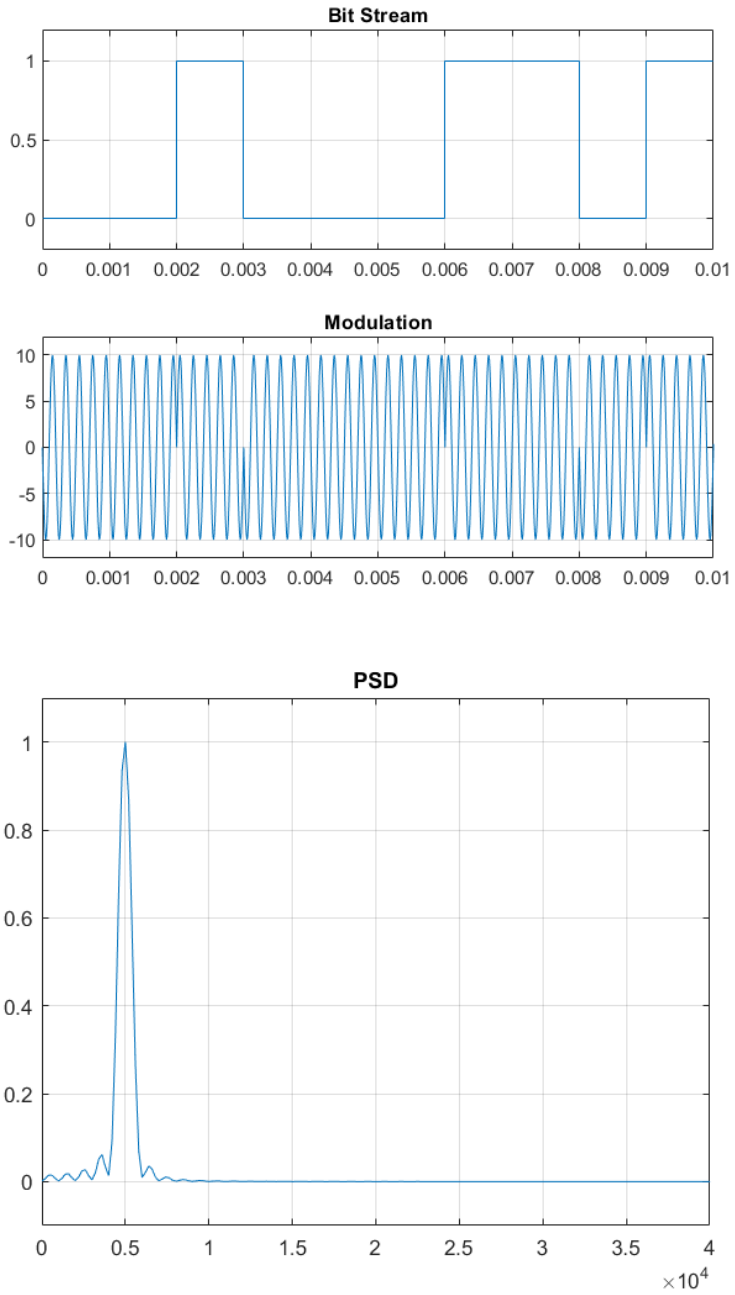



Fig: BPSK modulation and power spectral density

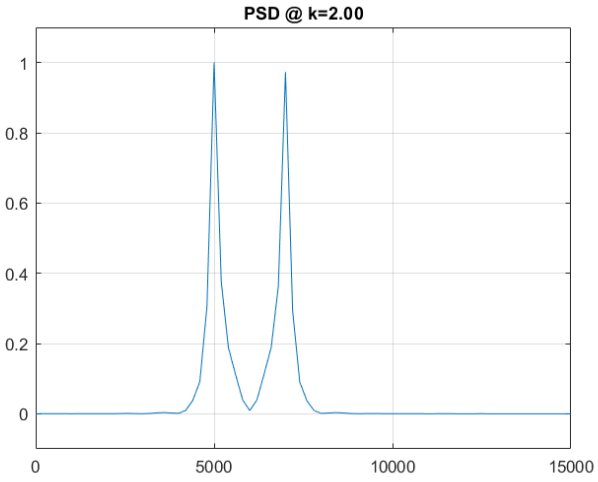
## Code snippet for BPSK modulation:

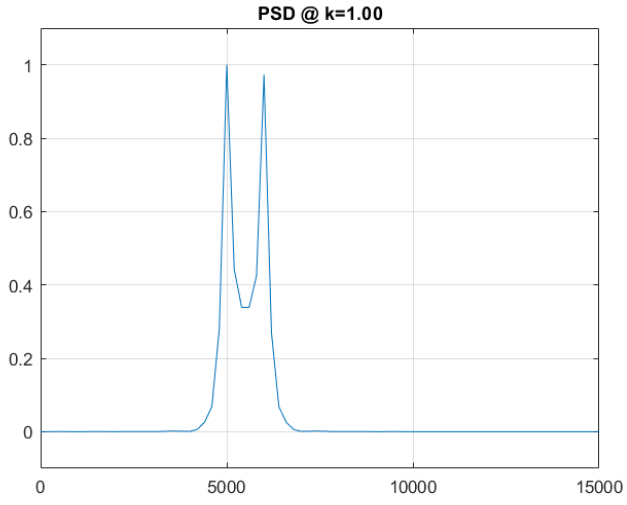
```
clc;
clear all;
close all;
N=5000;
A0=10;
A1=10;
ph1 = 0;
ph2 = pi;
fc=5e3;
Rb = 1000;
Tb = 1/Rb;
Fs = 20*fc;
Ts = 1/Fs;
k = Fs*Tb;


bits = randi([0,1], 1, N);
bits = repelem(bits, k);
t= 0:Ts:N*Tb-Ts; %linspace(0, 0.5, N*k);


figure;
subplot(211);
stairs(t,bits)
xlim([0, 0.01]);
ylim([-0.2, 1.2]);
title("Bit Stream")
grid on;


cbits = 1.-bits;
y = bits.*(A1*sin(2*pi*fc*t+ph1)) + cbits.*(A0*sin(2*pi*fc*t+ph2));
% figure;
subplot(212);
plot(t, y);
xlim([0, 0.01]);
ylim([-12, 12]);
title("Modulation")
grid on;


figure;
[pxx f] = pwelch(y, 500, 200, 500, Fs);
pxx = pxx/max(pxx);
plot(f, pxx);
xlim([0, 40000]);
ylim([-0.1, 1.1]);
title("PSD")
grid on;
```

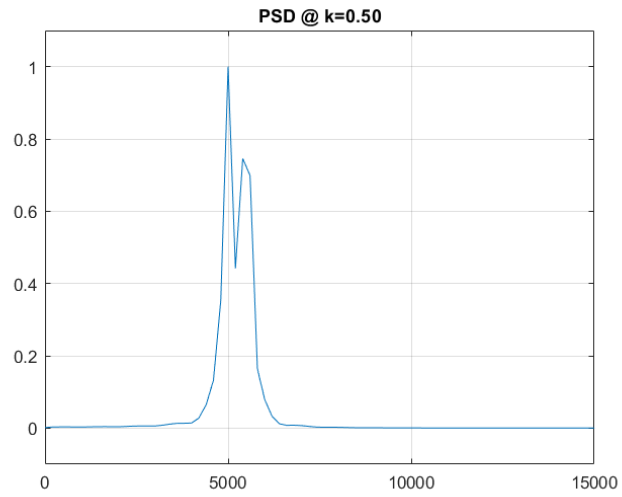
BPSK signal inverts the phase when the opposite bit stream is received. The power spectral density (PSD) of BPSK is wider than that of BASK signal.

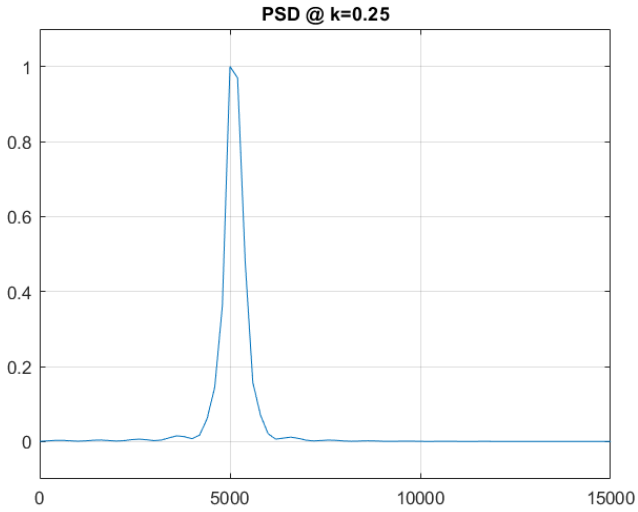
**(b) pwelch():** This function provides the power spectral density of a discrete time signal via Welch's averaged periodogram method. It divides the signal into segments equal in length to the length of window . The modified periodograms are computed using the signal segments multiplied by the vector, window . If window is an integer, the signal is divided into segments of length window. It takes 3 arguments such as ***window size, num of overlap, num of fft bin.***

**(c) BFSK** with various frequency distributions:




PSD @ k=0.50
1
0.8
0.6
0.4
0.2
0
0
5000
10000
15000


PSD @ k=0.25
1
0.8
0.6
0.4
0.2
0
0
5000
10000
15000

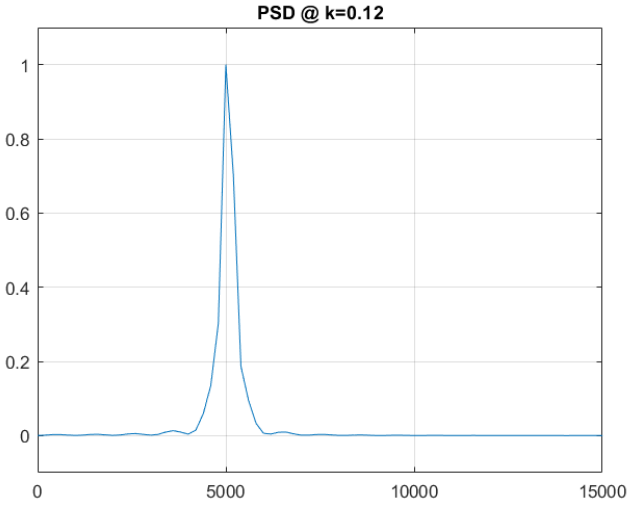

PSD @ k=0.12
1
0.8
0.6
0.4
0.2
0
0
5000
10000
15000

Fig: BPSK Modulation power spectral density using |f1-f2| = k/Tb for k=1, 2, ½, ¼, 1/8, respectively

We can see that reducing the k value makes the frequency gap smaller. Normally bigger differences give lower BER with a cost of larger bandwidth.

**Lab Task 2:**

**(a) Demodulation:**

**BASK:**

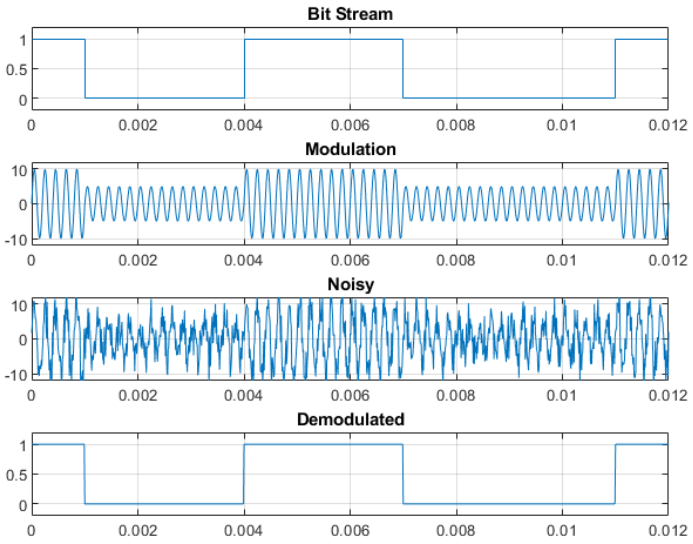



Fig: (a) original bit stream (b) modulated signal without noise (c) mixer output (d) demodulated output signal

**Code snippet:**

```
figure;
subplot(411);
stairs(t,bits)
xlim([0, 0.012]);
ylim([-0.2, 1.2]);
title("Bit Stream")
grid on;

subplot(412);
```

```
plot(t, y_ask);
xlim([0, 0.012]);
ylim([-12, 12]);
title("Modulation")
grid on;

subplot(413);
plot(t, yn_ask0);
xlim([0, 0.012]);
ylim([-12, 12]);
title("Noisy")
grid on;

subplot(414);
plot(t, repelem(yd_ask, k));
xlim([0, 0.012]);
ylim([-0.2, 1.2]);
title("Demodulated")
grid on;                        % amplitude for bit 0
```

In the absence of additive noise, the mixer output is a sin-squared signal, and after the integrator and threshold, the original message signal is decoded without errors.

**Code Snippet:**

```
mc = awgn(modulated,2,'measured');     % received (signal+noise) 2dB
```

Adding 2dB additive white Gaussian noise causes distortion in the modulated signal, and this causes errors in the received bit sequence as can be seen from the graph above.

**BFSK:**

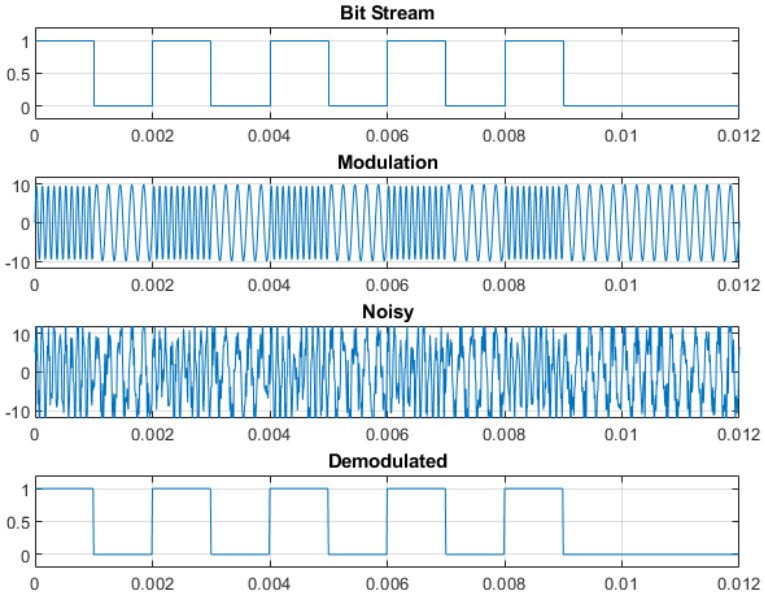



Fig: (a) Original message signal (b) Frequency modulated signal (c) Mixer 1 output (d) Mixer 2 output (e) demodulated bit stream after integration and compare

## Code Snippet:

```
figure;
subplot(411);
stairs(t,bits)
xlim([0, 0.012]);
ylim([-0.2, 1.2]);
title("Bit Stream")
grid on;

subplot(412);
plot(t, y_fsk);
xlim([0, 0.012]);
ylim([-12, 12]);
title("Modulation")
grid on;

subplot(413);
plot(t, yn_fsk0);
xlim([0, 0.012]);
ylim([-12, 12]);
title("Noisy")
grid on;
```

```
subplot(414);
plot(t, repelem(yd_fsk, k));
xlim([0, 0.012]);
ylim([-0.2, 1.2]);
title("Demodulated")
grid on;                % comparator output for decoded bit stream
```

Multiplying frequency modulated signal with the two separate modulating frequencies result in two different characteristics, and after integration and comparison, the original bit stream is recovered.

As we can see, the demodulated output in presence of 0dB white Gaussian noise does not have errors in the sampled length. It crudely proves that BFSK has better noise performance than BASK but at the cost of a higher bandwidth.

**BPSK:**

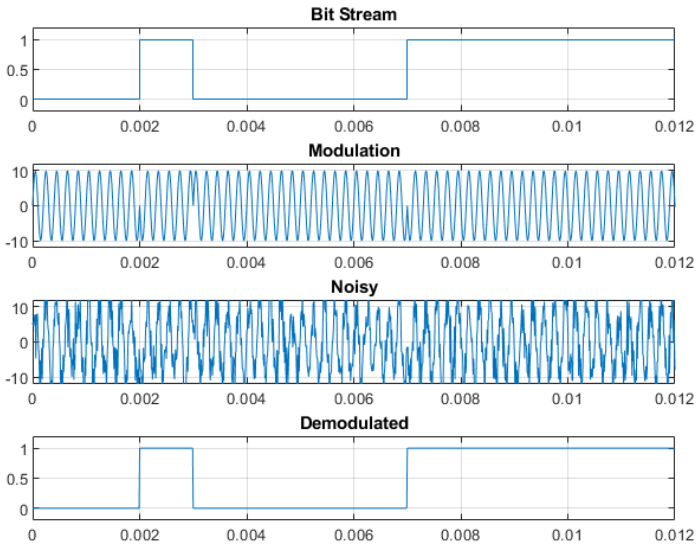



Fig: (a) Original message signal (b) Phase modulated signal (c) Mixer 1 output (d) Demodulated bit stream after integration and compare

**Code Snippet:**

```
figure;
subplot(411);
stairs(t,bits)
xlim([0, 0.012]);
ylim([-0.2, 1.2]);
title("Bit Stream")
grid on;

subplot(412);
plot(t, y_psk);
xlim([0, 0.012]);
ylim([-12, 12]);
title("Modulation")
grid on;

subplot(413);
plot(t, yn_psk0);
xlim([0, 0.012]);
ylim([-12, 12]);
title("Noisy")
grid on;

subplot(414);
plot(t, repelem(yd_psk, k));
xlim([0, 0.012]);
ylim([-0.2, 1.2]);
title("Demodulated")
grid on;
```

BPSK mixer output signal has a positive sin-square shape for binary 1 and negative sin-square for binary 0, as a result a simple threshold of 0 at integrator can recover original message.

## (b) BER vs SNR performance:

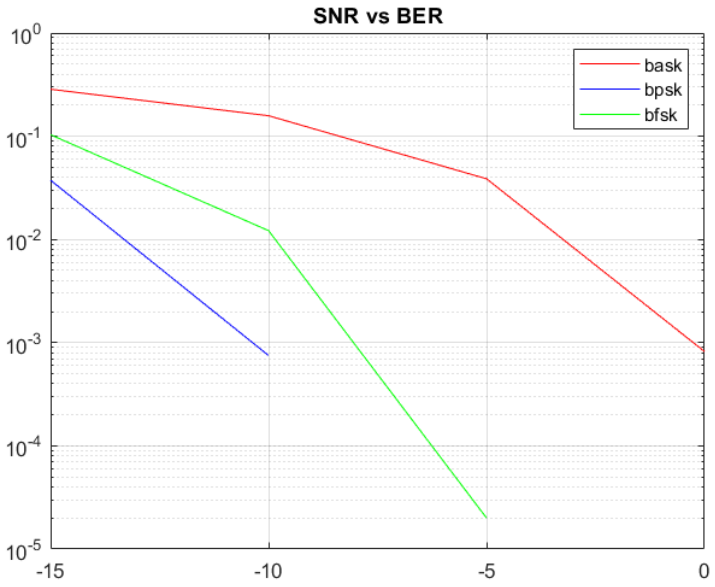



Fig: BER vs SNR performance for BASK, BFSK, BPSK modulation

## Code Snippet:

```
% Modulation
clc;
clear all;
close all;
N=1e5;
A0=5;
A1=10;
ph1 = 0;
ph2 = pi;
fc1=5e3;
fc2=2*fc1;
Rb = 1000;
Tb = 1/Rb;
Fs = 20*fc1;
Ts = 1/Fs;
k = Fs*Tb;
vth_ask = (A0 + A1)/4;
vth_psk = 0;


b = randi([0,1], 1, N);
bits = repelem(b, k);
t= 0:Ts:N*Tb-Ts; %linspace(0, 0.5, N*k);
% figure;
% subplot(211);
% stairs(t,bits)
% xaxis([0, 0.005]);
cbits = 1.-bits;
mc1 = sin(2*pi*fc1*t+ph1);
mc2 = sin(2*pi*fc1*t+ph2);
mc3 = sin(2*pi*fc2*t+ph1);
y_ask = bits.*(A1*mc1) + cbits.*(A0*mc1);
```

```
y_psk = bits.*(A0*mc2) + cbits.*(A0*mc1);
y_fsk = bits.*(A0*mc3) + cbits.*(A0*mc1);
% figure;
% subplot(212);
% plot(t, y_ask);

SNR = [-15, -10, -5, 0, 5];

yint_ask = [];
yint_psk = [];
yint_fsk = [];

yber_ask = [];
yber_psk = [];
yber_fsk = [];

count = 0;
for i=SNR
    count=count+1;
    yn_ask = awgn(y_ask, i, 'measured');
    yn_ask = yn_ask.*mc1;

    yn_psk = awgn(y_psk, i, 'measured');
    yn_psk = yn_psk.*mc1;

    yn_fsk = awgn(y_fsk, i, 'measured');
    yn_fsk1 = yn_fsk.*mc1;
    yn_fsk2 = yn_fsk.*mc3;


    for j=1:N
        yint_ask(j) = sum(yn_ask(k*(j-1)+1:k*j))/(k);
        yint_psk(j) = sum(yn_psk(k*(j-1)+1:k*j))/(k);
        yint_fsk1(j) = sum(yn_fsk1(k*(j-1)+1:k*j))/(k);
        yint_fsk2(j) = sum(yn_fsk2(k*(j-1)+1:k*j))/(k);
    end
    yd_ask = double(yint_ask>vth_ask);
    yd_psk = double(yint_psk<vth_psk);
    yd_fsk = double(yint_fsk1<yint_fsk2);

    %ber
    yber_ask(count) = sum(abs(yd_ask-b))/N;
    yber_psk(count) = sum(abs(yd_psk-b))/N;
    yber_fsk(count) = sum(abs(yd_fsk-b))/N;
end

figure;
semilogy(SNR, yber_ask,'r'); hold on;
semilogy(SNR, yber_psk,'b'); hold on;
semilogy(SNR, yber_fsk,'g'); hold on;
title("SNR vs BER");
legend('bask','bpsk','bfsk');
grid on;
% ylim([0,1]);
```

From the BER vs SNR curves, it can be said that BASK has the worst noise performance. Then comes BFSK and then comes BPSK. So, BASK < BFSK < BPSK in terms of performance.