EE4TK4 Project 2

Computer Simulation on Matched Filters for Detection of BSPK and BFSK Signals

March 29, 2007

Diego Benavides 0241824

1. Background

Using the binary transmission system shown below this project examines matched filter simulations for BPSK and BFSK schemes.

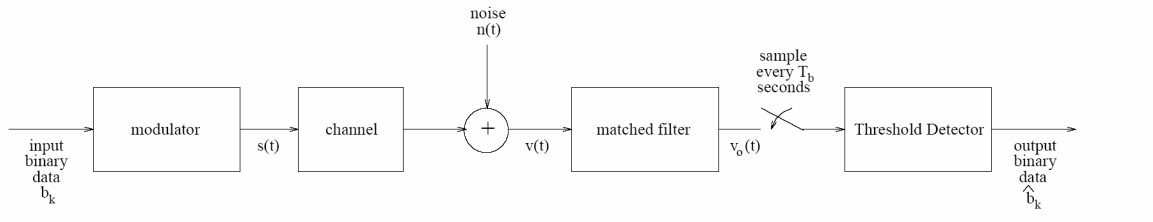

Figure 1: Binary Transmission System

The system input is a binary sequence b_k of bit duration T_b using the following modulation schemes.

BFSK Modulated Signal

$$s(t)=\begin{cases} s_1(t)=A\cos(\omega_1 t) & \text{if } b_k=1 \\ s_2(t)=A\cos(\omega_2 t) & \text{if } b_k=0 \end{cases}$$

Where we have the following frequencies:

$$\omega_1=\frac{2\pi(n_0+n_1)}{T_b}$$

$$\omega_2=\frac{2\pi(n_0-n_1)}{T_b}$$

Additionally, we define:

$$n_0=4$$

$$n_1=1$$

BPSK Modulated Signal

$$s(t)=\begin{cases} s_1(t)=+A\cos(\omega_c t) & \text{if } b_k=1 \\ s_2(t)=-A\cos(\omega_c t) & \text{if } b_k=0 \end{cases}$$

The carrier frequency is as follows:

$$\omega_c=4\frac{2\pi}{T_b}$$

The modulated signal is transmitted through the channel and corrupted by noise, assumed to be white Gaussian with power spectral density $S_n(\omega) = \frac{\eta}{2}$. The optimum receiver is composed of a matched filter, sampler and threshold detector. The signal which arrives at the receiver, containing noise, is filtered to produce $v_o(t)$, which is then compared to a pre-determined threshold V_T according to the rule:

$$\hat{b}_k = \begin{cases} 1 & \text{if } v_o(T_b) > V_T \\ 0 & \text{if } v_o(T_b) < V_T \end{cases}$$

2. Rectangular Pulse Bandwidth (Null-To-Null)

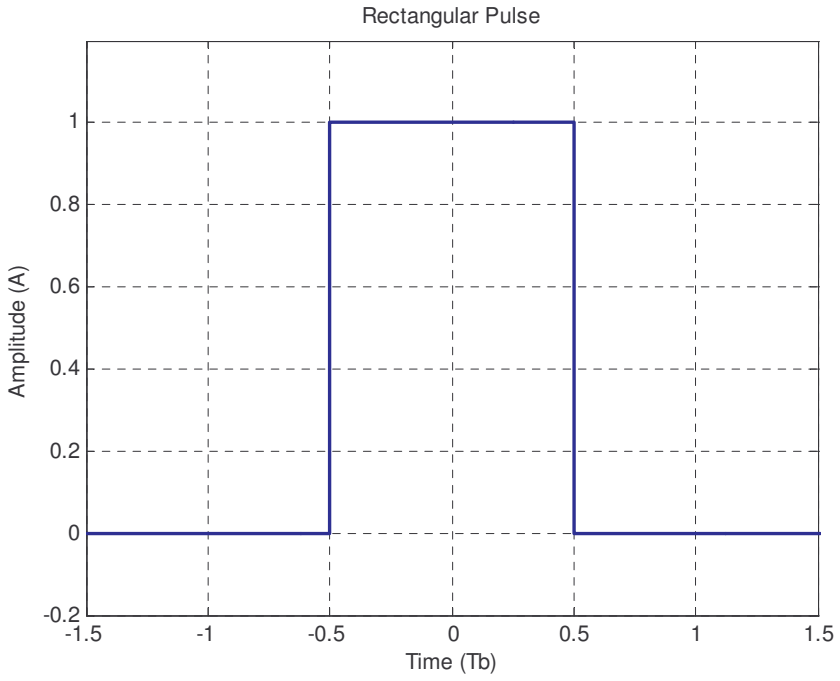

Figure 2: Rectangular Pulse

The frequency spectrum required for representation of a perfect rectangular pulse of width T_b is infinite. This is due to the fact that a rectangular pulse has a Fourier representation of a *sinc* pulse in the frequency domain, which is defined along the entire spectrum. The following calculation demonstrates this in practice.

$$x(t) = A\left[u\left(t+\frac{T_b}{2}\right) - u\left(t-\frac{T_b}{2}\right)\right]$$

$$X(\omega) = F\{x(t)\}$$

$$= \int_{-\infty}^{\infty} A\left[u\left(t+\frac{T_b}{2}\right) - u\left(t-\frac{T_b}{2}\right)\right] e^{-j\omega t}\,dt$$

$$= A\left[\int_{\frac{-T_b}{2}}^{\infty} e^{-j\omega t}\,dt - \int_{\frac{T_b}{2}}^{\infty} e^{-j\omega t}\,dt\right]$$

$$= \frac{2A}{\omega}\left[\frac{1}{2j}\left(e^{j\omega\frac{T_b}{2}} - e^{-j\omega\frac{T_b}{2}}\right)\right]$$

$$X(\omega) = \frac{2A}{\omega}\sin\left(\omega\frac{T_b}{2}\right)$$

If we consider the *sinc* pulse as a low-pass signal, that is, one whose main spectral content is centered about zero, we can take the bandwidth as half the width of the main spectral lobe. In other words, we need to find roots of $X(\omega)$.

$$0 = \frac{2A}{\omega}\sin\left(\omega\frac{T_b}{2}\right)$$

$$\sin^{-1}(0) = \omega\frac{T_b}{2} + 2\pi k_0$$

$$\pi k_1 = \omega\frac{T_b}{2} + 2\pi k_0$$

$$\pi(k_1 - 2k_0) = \omega\frac{T_b}{2}$$

$$\frac{2\pi k}{T_b} = \omega$$

We now have an adequate relationship to define the bandwidth. For the half width of the main lobe, we use the first root (k=1) and find that the bandwidth is $\frac{2\pi}{T_b}$.

3. Noise Power

Gaussian noise power is equivalent to its area as represented in the frequency domain. Since we have a constant spectral density of $\frac{\eta}{2}$, we need only take the product of this with our valid range of frequency, which will be limited by the sampling frequency $\omega_s = \frac{2\pi}{T_s}$.

The formulation for the noise power is thus as follows:

$$\rho_n = \frac{\eta}{2}\left(\frac{2\pi}{T_s}\right)$$
$$= \frac{\eta\pi}{T_s}$$

4. Signal to Noise Ratio

An effective signal to noise ratio is defined in the following form, since the noise bandwidth is much larger than the effective bandwidth of signal pulses:

$$\rho = \frac{A^2 T_b}{2\eta}$$

When expressed in dB, the amplitude of the modulate pulse (A) is computed as follows:

$$\rho_{dB} = 10\log 10\left[\frac{A^2 T_b}{2\eta}\right]$$
$$A = \sqrt{\frac{2\eta 10^{\frac{\rho_{dB}}{10}}}{T_b}}$$

5. Signal Generation Using FSK and PSK

Using the aforementioned techniques for FSK and PSK modulation, we are able to generate the following transmission signals.

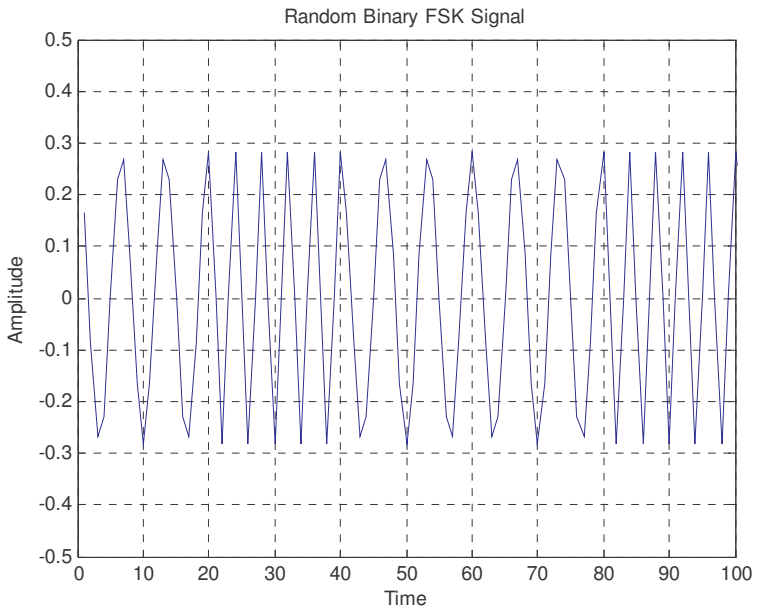

Figure 3: Binary FSK Signal

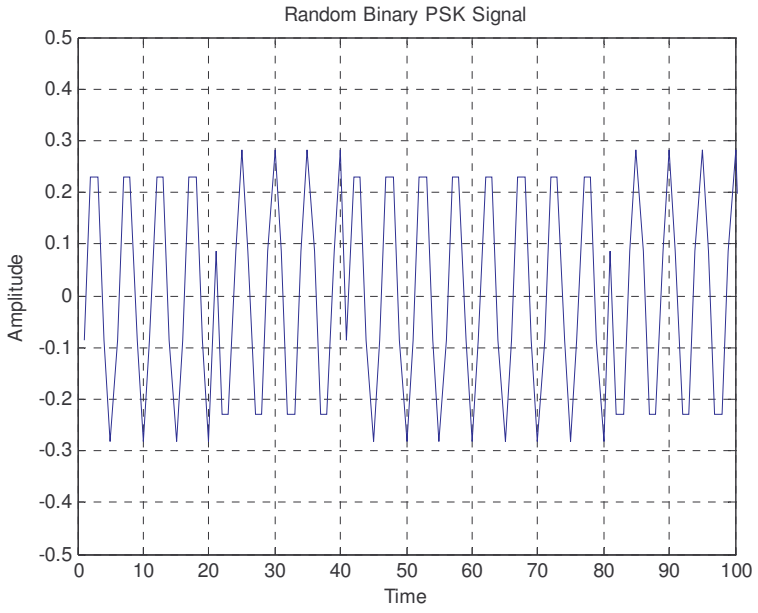

Figure 4: Binary PSK Signal

6. Shift-Keying Signals with Gaussian Noise

Adding zero-mean Gaussian white noise to the previously generated signals yields the following results.

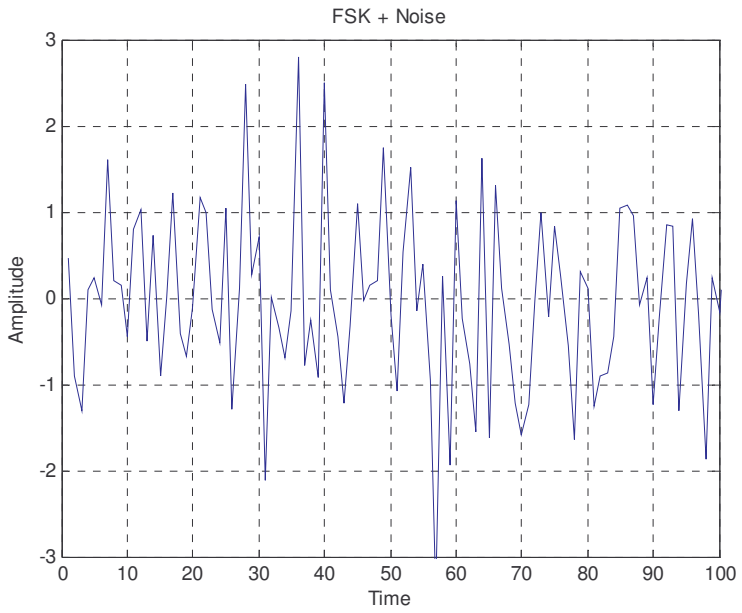

Figure 5: Noisy FSK Signal

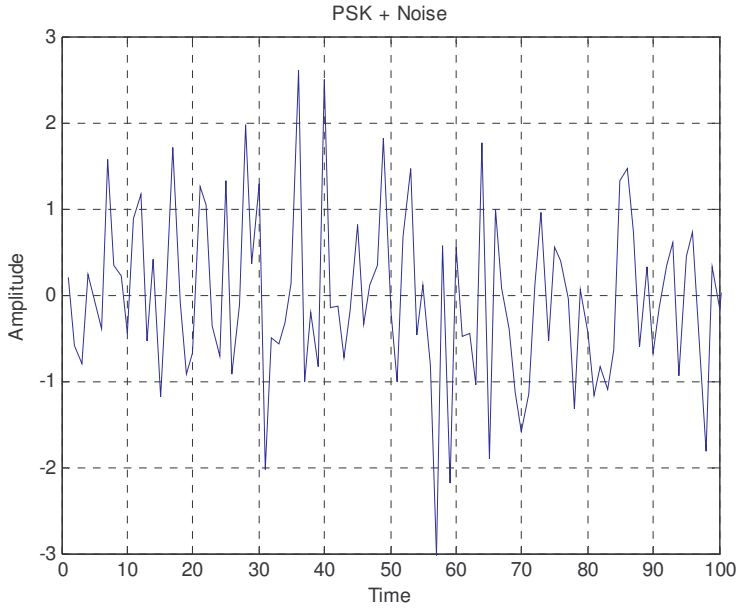

Figure 6: Noisy PSK Signal

7. Filtering and Decoding

The optimum receiver is of the form $h(t) = p(T-t)$, where $p(t) = s_1(t) - s_2(t)$. This filter type was implemented in MATLAB. The output produced by the filter was

subsequently used to decode the noisy modulated signal messages. This experiment was performed for 20 realizations of noise, over a spectrum of signal to noise ratios ranging from -4dB to +4dB.

8. Results and Discussion

It was found in this experiment that as the signal to noise ratio increases, the amount of error present in the optimum receiver is greatly reduced. This is of course expected as noise power will play less of a role in affecting the signal arriving at the receiver. Additionally, from the graphs which follow, it can be seen that the performance of PSK against FSK reduces further the probability of bit error detection by as much as 9%. This is especially evident at lower values of signal to noise ratios. Furthermore, the power required by phase shift-keying is less than that of frequency shift-keying modulation and detection schemes.

The following graphs demonstrate the results of this experiment.

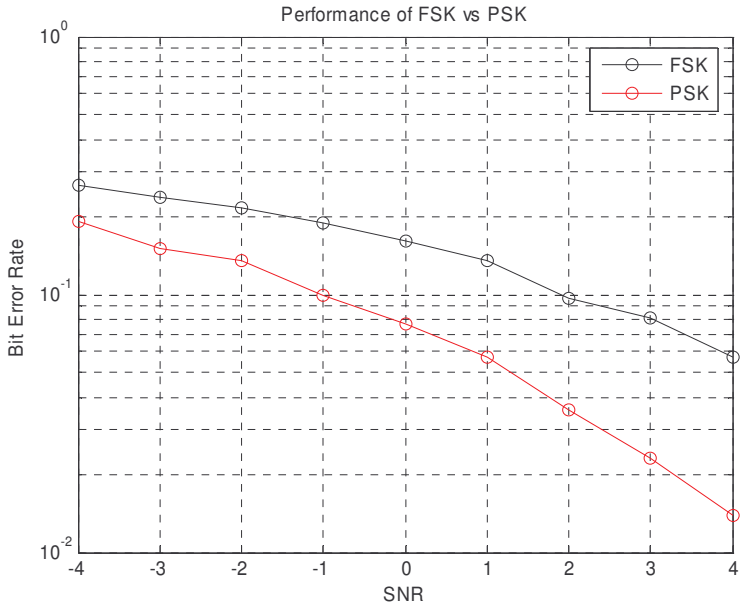

Figure 7: Bit Error Rates of FSK and PSK vs SNR

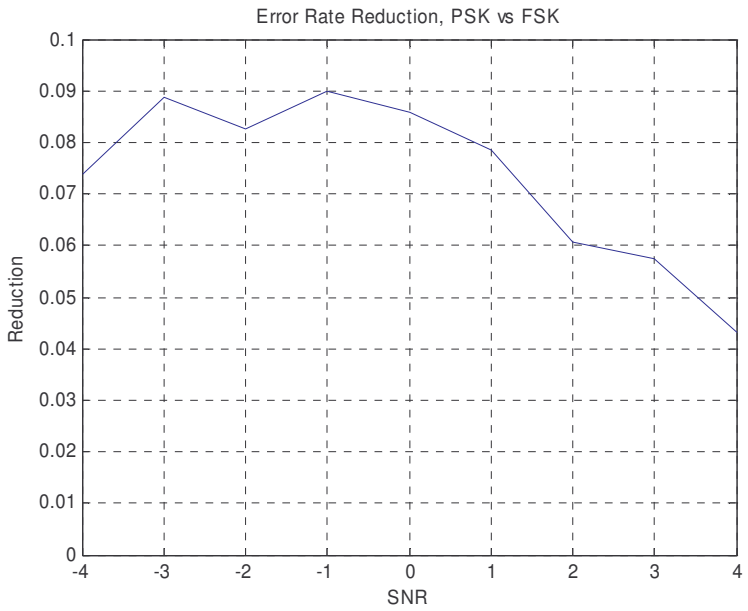

Figure 8: Error Rate Difference, FSK-PSK

Finally, these two graphs demonstrate the proximity of these experimental results to the theoretical error rates of PSK and FSK modulation schemes. The curves follow the same trends and are within small margins of each other. Thus, the experiment results are found to agree very closely with our theoretical expectations.

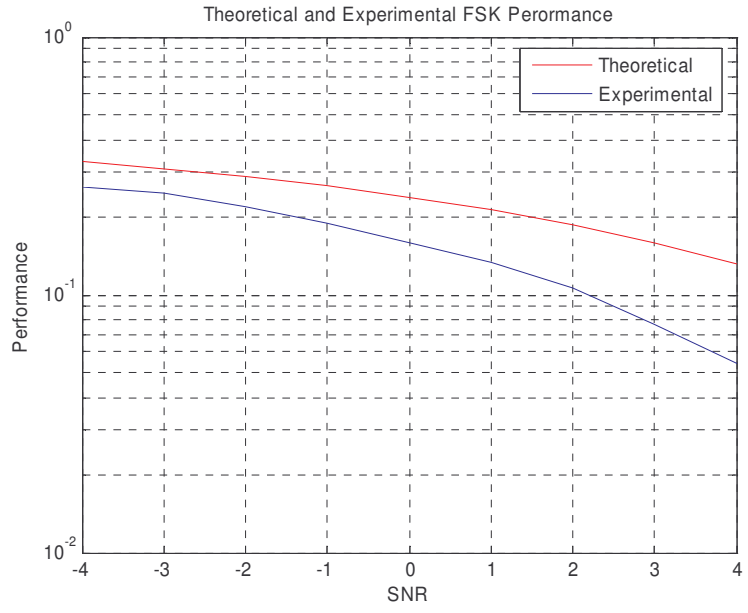

Figure 9: Theoretical and Experimental FSK Performance

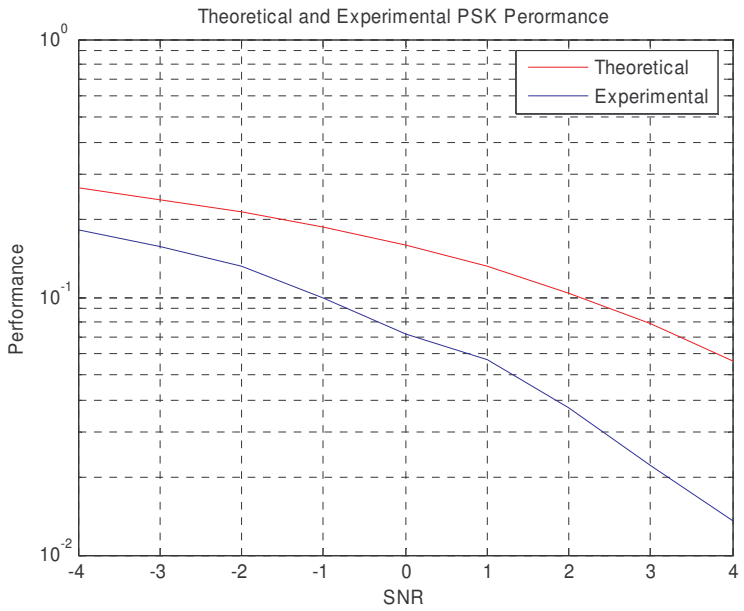

Figure 10: Theoretical and Experimental PSK Performance

9. Appendix

MATLAB Code

```
%*************************************************************************
%** Computer Simulation On Matched Filters for BPSK and BFSK Detection   **
%** Diego Benavides 0241824                                              **
%** March 29, 2007                                                       **
%*************************************************************************

%*************************************************************************
%** Initialize workspace ************************************************
close all;
clear;
clc;
%*************************************************************************

%*************************************************************************
%** Constants and Variable Setup ****************************************
toPlot=0;                       % set to 1 to show some sample graphs

n0=4;                           % used for w1, bfsk
n1=1;                           % used for w2, bfsk

N=20;                           % defined in project outline
Ts=1;                           % sampling period of binary sequence
Tb=N*Ts;                        % period of binary sequence

eta=2;                          % noise spectral content

SNRe=-4:4;                      % SNR: -4dB to 4dB

t=0:N-1;                        % timing for simulation
bitLength=500;                  % chosen abritrarily


vi=1;                           % virtual index

% number of bit errors
err_fsk=0;
err_psk=0;

% decoded output sequences
bko_fsk=zeros(1,bitLength);
bko_psk=zeros(1,bitLength);
%*************************************************************************
```

```
%************************************************************************
%** Set up shift keying frequencies ***************************************
w1=2*pi*(n0+n1)/Tb;             % first bfsk frequency
w2=2*pi*(n0-n1)/Tb;             % second bfsk frequency

wc=n0*2*pi/Tb;                  % carrier frequency, bpsk
%************************************************************************

bk=round(rand(1,bitLength));    %1. generate random binary sequence
A=sqrt(2*eta/Tb*10.^(SNRe/10)); %3. compute A for varying SNR levels

for SNR_loop=1:length(SNRe)
    for TRIAL_loop=1:N          % 20 independent trials
    err_fsk=0;
    err_psk=0;

    noise=randn(1,bitLength*N); %2. generate zero-mean white Gaussian noise
    vi=1;

    %************************************************************************
    %** Use A to setup signal functions ***********************************
    s1_fsk=A(SNR_loop)*cos(w1*t);   % bfsk 1
    s2_fsk=A(SNR_loop)*cos(w2*t);   % bfsk 0

    s1_psk=A(SNR_loop)*cos(wc*t);   % bpsk 1
    s2_psk=-A(SNR_loop)*cos(wc*t);  % bpsk 0

    d=length(s1_fsk);               % signal duration
    %************************************************************************

    for i=1:bitLength             %4. generate FSK and PSK mod signals
        if(bk(i)==1)              % virtual index
            s_fsk(vi:vi+(N-1))=s1_fsk(1:N);
            s_psk(vi:vi+(N-1))=s1_psk(1:N);
        else
            s_fsk(vi:vi+(N-1))=s2_fsk(1:N);
            s_psk(vi:vi+(N-1))=s2_psk(1:N);
        end
        vi=vi+N;
    end

    v_fsk=s_fsk+noise;              %5. generate noisy signals
    v_psk=s_psk+noise;

    if(toPlot==1)                   % display preliminary output if desired
        figure(1)
```

```
        plot(s_fsk);
        axis([0,100,-1/2,1/2]);
        title('Random Binary FSK Signal');
        ylabel('Amplitude');
        xlabel('Time');
        set(gcf,'Color',[1 1 1])
        grid on

        figure(2)
        plot(s_psk);
        axis([0,100,-1/2,1/2]);
        title('Random Binary PSK Signal');
        ylabel('Amplitude');
        xlabel('Time');
        set(gcf,'Color',[1 1 1])
        grid on

        figure(3)
        plot(v_fsk);
        axis([0,100,-3,3]);
        title('Random Binary FSK Signal');
        ylabel('Amplitude');
        xlabel('Time');
        set(gcf,'Color',[1 1 1])
        grid on

        figure(4)
        plot(v_psk);
        axis([0,100,-3,3]);
        title('Random Binary PSK Signal');
        ylabel('Amplitude');
        xlabel('Time');
        set(gcf,'Color',[1 1 1])
        grid on
    end

    %6. create matched filters and corresponding signal through each
    mf_fsk=fliplr(s1_fsk-s2_fsk);   % BFSK output
    mf_psk=fliplr(s1_psk-s2_psk);   % BPSK output

    vo_fsk=conv(v_fsk,mf_fsk);
    vo_psk=conv(v_psk,mf_psk);

    % adjustment for detection
    vo_fsk=vo_fsk(1:bitLength*N);
    vo_psk=vo_psk(1:bitLength*N);
```

```
    % sample the output signal
    for m=N:N:(bitLength*N)
        vo_fsk(m/N)=vo_fsk(m);
        vo_psk(m/N)=vo_psk(m);
    end

    %7. threshold detection
    for m=1:bitLength*N;
        if(vo_fsk(m)<=0)                % PSK detection
            bko_fsk(m)=0;
        else
            bko_fsk(m)=1;
        end

        if(vo_psk(m)<=0)                % PSK detection
            bko_psk(m)=0;
        else
            bko_psk(m)=1;
        end
    end

    %8. find bit error rates
    for m=1:bitLength;
        if(bk(m)~=bko_fsk(m))
            err_fsk=err_fsk+1;
        end

        if(bk(m)~=bko_psk(m))
            err_psk=err_psk+1;
        end
    end

    ber_fsk(TRIAL_loop)=err_fsk/bitLength;
    ber_psk(TRIAL_loop)=err_psk/bitLength;
    end

    fprintf(' Bit Error Rates, SNR=%ddB\n',SNRe(SNR_loop));
    fprintf(' FSK: \t%.4f\n',mean(ber_fsk));
    fprintf(' PSK: \t%.4f\n',mean(ber_fsk));

    performance_fsk(SNR_loop)=mean(ber_fsk);
    performance_psk(SNR_loop)=mean(ber_psk);

end
```

```
%************************************************************************
%** Overal results ******************************************************
figure(5)
semilogy(SNRe,performance_fsk,'-ok');
hold on
semilogy(SNRe,performance_psk,'-or');
title('Performance of FSK vs PSK');
ylabel('Bit Error Rate');
xlabel('SNR');
legend('FSK','PSK');
set(gcf,'Color',[1 1 1])
grid on

figure(6)
plot(SNRe,performance_fsk-performance_psk)
title('Error Rate Reduction, PSK vs FSK');
ylabel('Reduction')
xlabel('SNR')
axis([-4,4,0,0.1]);
set(gcf,'Color',[1 1 1])
grid on

figure(7)
semilogy(SNRe,1/2*erfc(1/2*sqrt(Tb/eta*A.^2)),'r');
hold on
semilogy(SNRe,performance_psk);
title('Theoretical and Experimental PSK Perormance');
ylabel('Performance')
xlabel('SNR')
legend('Theoretical','Experimental');
set(gcf,'Color',[1 1 1])
grid on

figure(8)
semilogy(SNRe,1/2*erfc(1/2*sqrt(Tb/eta/2*A.^2)),'r');
hold on
semilogy(SNRe,performance_fsk);
title('Theoretical and Experimental FSK Perormance');
ylabel('Performance')
xlabel('SNR')
legend('Theoretical','Experimental');
set(gcf,'Color',[1 1 1])
grid on
%************************************************************************
```