EE526 Project 2

Kevin Mack, Clarkson University 03/08/2018

Problem Definition:

This project inspects three communication schemes (BPSK, OOK, and QPSK) in order to determine the achievable BER for each scheme under different noise conditions. The theoretical BER for each SNR value will be compared to the simulated results (Monte-Carlo Simulation) for each scheme. In the end, all schemes will be compared over the same range of SNR values.

Binary Phase Shift Keying (BPSK):

For BPSK the message, $\mathbf{X} \in [-1, 1]$ and $\mathbf{N} \sim \mathcal{N}(0, \sigma_n^2)$. This means that, assuming equal prior probabilities for each symbol, the Bayesian Decision Rule gives threshold at $x = 0$. The signal is given by,

$$H_0 : \mathbf{X} = \mathbf{N}, \tag{1}$$

$$H_1 : \mathbf{Y} = \mathbf{S} + \mathbf{N}. \tag{2}$$

In this case, the theoretical value for the BER is given by,

$$P_e = Q\left(\sqrt{2\gamma_b}\right) \tag{3}$$

where $Q(.)$ is the q-function, and γ_b is the SNR per bit. In order to test this theoretical value, a Monte-Carlo simulation is used to give an experimental result. Figure 1 represents the results of the Monte-Carlo solution, and Listing 1 below provides the code to generate these results.

Listing 1: Matlab Code – BPSK Simulation

```
%% BPSK Monte-Carlo Simulation
snr_max = 9.0; %dB
snr_min = 2.0; %dB
snr_step = 0.2;

snr_range = snr_min:snr_step:snr_max;

thresh_BPSK = 0;

num_transmissions = 1e6;

% set up figure
```

```matlab
figure
hold all
grid on
grid minor

error_theo = zeros(size(snr_range,2), 1);
error_sim = zeros(size(snr_range, 2), 1);
for i = 1:size(snr_range, 2)
    received_message = zeros(num_transmissions, 1);

    %calculate variance from SNR
    N_0 = 1/snr_range(i);%1;% noise variance
    sigma_sq = N_0/2;

    % generate random signal of 1's and 0's (P(0)=P(1)
    bits = randi([0 1], num_transmissions, 1);

    % for BPSK, the bits should be -1 and +1, so 0 -> -1
    X = bits;
    X(X == 0) = -1;

    % generate random gaussian noise
    Z = sqrt(sigma_sq).*randn(size(X, 1), 1);

    % add noise to signal
    Y = X + Z;

    % check signal against threshold
    ind = find(Y> thresh_BPSK);
    received_message(ind) = 1;

    % get simulation bit error rate (BER)
    errors = received_message(received_message ~= bits);
    error_sim(i) = size(errors,1)/size(X,1);

    % calculate theoretical BER
    error_theo(i) = qfunc(sqrt(2*snr_range(i)));
end

plot(snr_range, log10(error_sim), 'ko')
plot(snr_range, log10(error_theo), 'k--')
title('Detector Performance with Varying SNR (BPSK)')
xlabel('SNR [dB]')
ylabel('BER')
legend('Simulated BER', 'Theoretical BER')
```

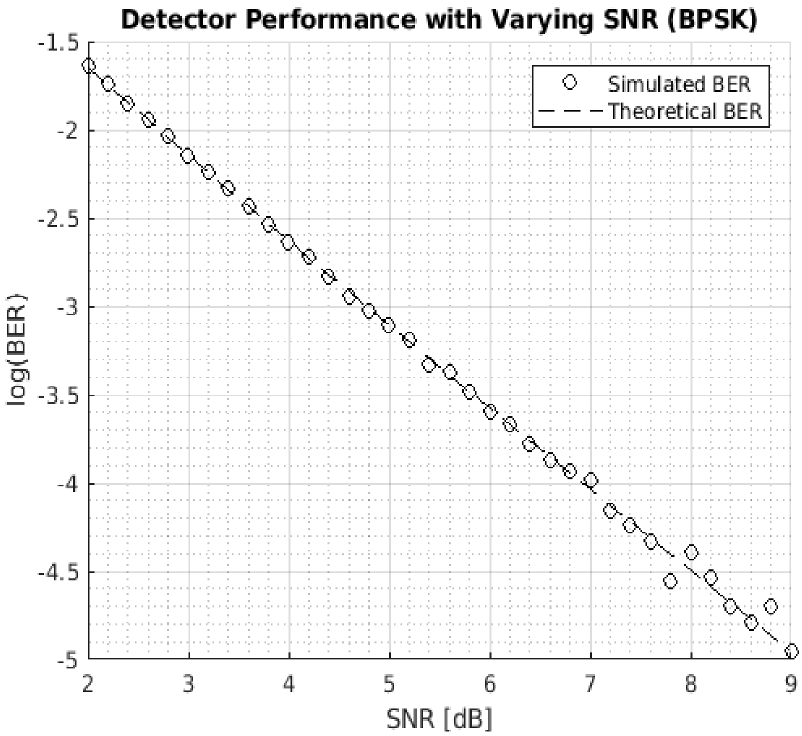

Figure 1: This figure depicts the results of a Monte-Carlo simulation ($1e^6$ iterations) to determine the relationship between SNR and BER the BPSK communication scheme. It is clear from the graph that the theoretical BER is comparable to the simulated BER.

On/Off Keying (OOK):

For the OOK communication scheme, the initial setup is the same as for BPSK. However, the only difference is that $\mathbf{X} \in [0, 1]$. This means that the threshold for detection is at $x = 0.5$, and that there is less 'space' between the on and off state. This will result in a degradation of BER performance as compared with BPSK, as can be seen in 2. The theoretical BER is given by,

$$P_e = Q(\sqrt{\gamma_b}), \tag{4}$$

The code to simulate the OOK setup is given in Listing 2 below.

Listing 2: Matlab Code – OOK Simulation

```
snr_max = 12.0; %dB
snr_min = 5.0; %dB
snr_step = 0.2;

range = snr_min:snr_step:snr_max;

thresh_OOK = 0.5;

num_transmissions = 1e6;

```

```
% set up figure
figure
hold all
grid on
grid minor

error_theo = zeros(size(range,2), 1);
for i = 1:size(range, 2)
    received_message = zeros(num_transmissions, 1);

    %calculate variance from SNR
    N_0 = 0.5/range(i);% noise variance
    sigma_sq = N_0/2;

    % thresh = some_equation_based_on_noise_variance;
    % generate random signal of 1's and 0's (P(0)=P(1)
    bits = randi([0 1], num_transmissions, 1);

    % for OOK, the bits should be 0 and +1
    X = bits;

    % generate random gaussian noise
    Z = sqrt(sigma_sq).*randn(size(X, 1), 1);

    % add noise to signal
    Y = X + Z;

    % check signal against threshold
    ind = find(Y > thresh_OOK);
    received_message(ind) = 1;

    % get simulation bit error rate (BER)
    errors = received_message(received_message ~= bits);
    error_sim(i) = size(errors,1)/size(X,1);

    % calculate theoretical BER
    error_theo(i) = qfunc(sqrt(range(i)));
end

plot(range, log10(error_sim), 'ko')
plot(range, log10(error_theo), 'k--')
title('Detector Performance with Varying SNR (OOK)')
xlabel('SNR [dB]')
ylabel('log(BER)')
legend('Simulated BER', 'Theoretical BER')
```

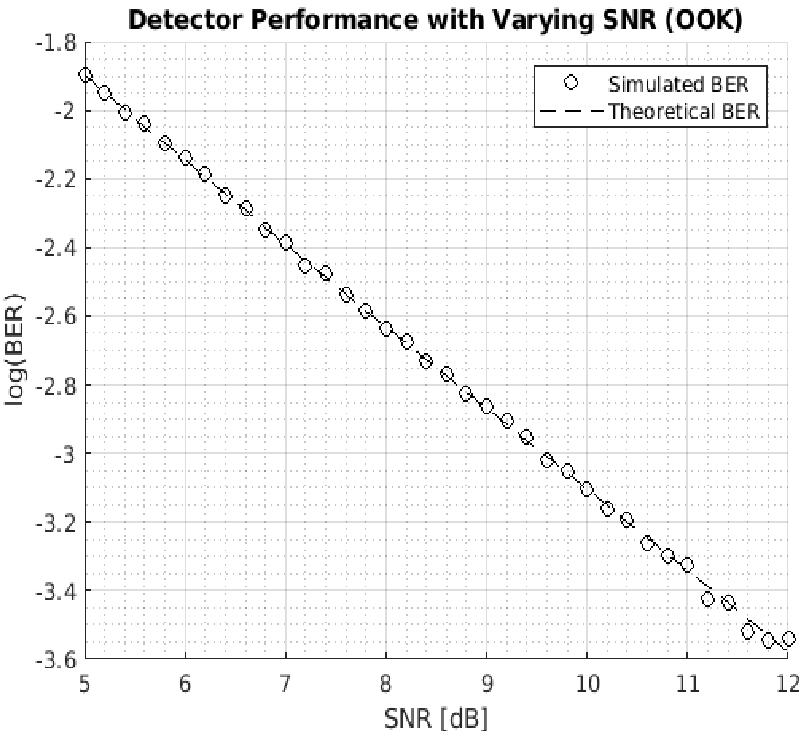

Figure 2: This figure depicts the effect of SNR on the BER for the OOK communication scheme. In this case, it is shown that the theoretical results agrees with the results of a Monte-Carlo simulation ($1e^6$ iterations)

.

Quadrature Phase-Shift Keying (QPSK):

In this section, the communication scheme will be identical to that of BPSK except that $\mathbf{X} \in \mathbb{C}$. This means that it is possible to send two bits of information per transmission (a real part and an imaginary part). Theoretically, the bit error rate is the same as BPSK, but this type modulation can allow for twice as much information to be sent. In order to test this, a Monte-Carlo simulation was done just as with BPSK and OOK. In this case, the noise added to this system was $N \sim \mathcal{CN}(0, \sigma_n^2)$. The results are given by Figure 3 and 4, and the code to generate them is given below in Listing 3. Figure 4 illustrates the data that the receiver would be processing in a noisy QPSK system.

Listing 3: Matlab Code – OOK Simulation

```
%% QPSK Monte-Carlo Simulation
%
%           |
%   10 x    |   x 00    (odd bit, even bit)
%           |
%   -------+------->   real part (I channel)
%           |
%   11 x    |   x 01
%           |
```

```

snr_max = 9.0; %dB
snr_min = 2.0; %dB
snr_step = 0.2;

snr_range = snr_min:snr_step:snr_max;

num_transmissions = 1e6;

% set up figure
figure
hold all
grid on
grid minor

len = 1/sqrt(2);
theta = zeros(num_transmissions, 1);
error_theo = zeros(size(snr_range,2), 1);
single_errors = zeros(size(snr_range, 1), 1);
double_errors = zeros(size(snr_range, 1), 1);
for i = 1:size(snr_range, 2)
    received_message = zeros(num_transmissions, 1);
    received  = zeros(num_transmissions, 1);
    % calculate variance from SNR
    N_0 = 1/snr_range(i);% noise variance
    sigma_sq = N_0/4;
    \begin{figure}[h!]
\begin{minipage}[b]{1.0\linewidth}
  \centering
  \centerline{\includegraphics[width=15cm, height=10cm]{EE526_Project2_OOK↩
      .png}}
% \vspace{2.0cm}
\end{minipage}
%
\caption{}
\label{fig:ook_ber}
\end{figure}
    % generate random signal of 0's, 1's, 2's, 3's of equal probability
    % that represent the 4 positions for symbos in QPSK
    bits = randi([0 3], num_transmissions, 1);

    % for QPSK, the bits should be
    X = bits;

    X(X == 0) = len + len*1i; % quadrant  1 = 00
    X(X == 1) = -len + len*1i; % qudrant  2 = 10
    X(X == 2) = -len + -len*1i;% quadrant 3 = 11
```

```
    X(X == 3) = len - len*1i;% quadrant  4 = 01

    % generate random complex gaussian noise
    Z = sqrt(sigma_sq).*randn(size(X, 1), 1) + sqrt(sigma_sq).*randn(size(X, 1), 1)*1i;

    % add noise to signal
    Y = X + Z;

    % check signal against threshold

    for j = 1:num_transmissions
        rec_bits = [real(Y(j)) imag(Y(j))];
        if(real(Y(j)) > 0 && imag(Y(j)) > 0 )
            % quadrant 1
            received_message(j) = 0;
        elseif (real(Y(j)) < 0 && imag(Y(j)) > 0 )
            % quadrant 2
            received_message(j) = 1;
        elseif(real(Y(j)) < 0 && imag(Y(j)) < 0 )
            % quadrant 3
            received_message(j) = 2;
        else
            %quadrant 4
            received_message(j) = 3;
        end
    end

    received(received_message == 0) = len + len*1i; % quadrant  1 = 00
    received(received_message == 1) = -len + len*1i; % qudrant  2 = 10
    received(received_message == 2) = -len + -len*1i;% quadrant 3 = 11
    received(received_message == 3) = len - len*1i;% quadrant  4 = 01

    %get simulation bit error rate (BER)

    diff = abs(received - X);
    double_errors(i) = size(diff(diff > 1.5), 1);
    single_errors(i) = size(diff(diff > 1 & diff < 1.5), 1);
    error_sim(i) = (single_errors(i) + 2*double_errors(i))/(2*size(X,1));

    %calculate theoretical BER
    error_theo(i) = qfunc(sqrt(2*snr_range(i)));
end

%plot
plot(snr_range, log10(error_sim), 'ko')
plot(snr_range, log10(error_theo), 'k--')
```

```
title('Detector Performance with Varying SNR (QPSK)')
xlabel('SNR [dB]')
ylabel('log(BER)')
legend('Simulated BER', 'Theoretical BER')
```

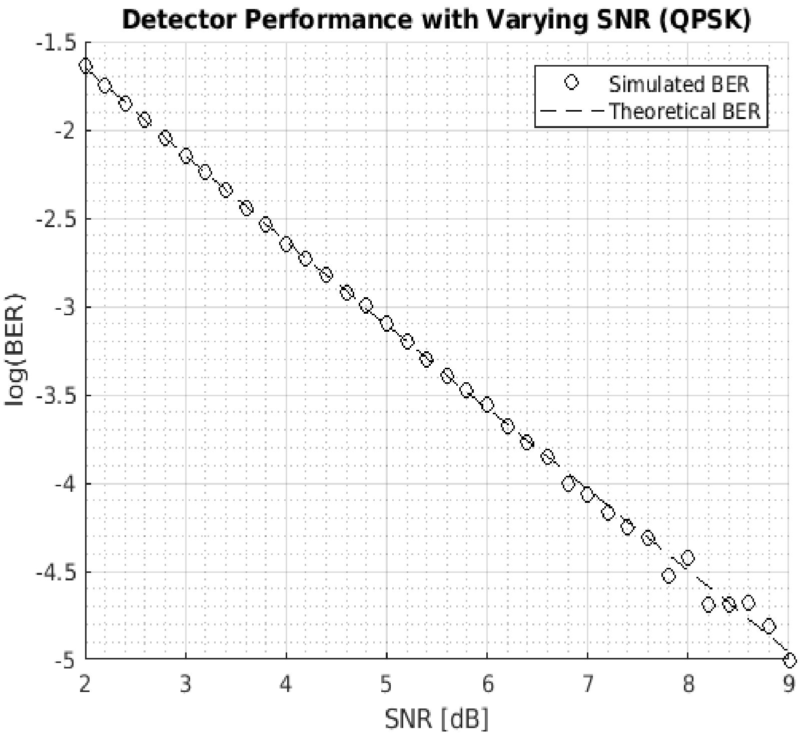

Figure 3: This figure represents the results of a Monte-Carlo simulation ($1e^6$ iterations) to verify the theoretical BER based on a given SNR value. As can be seen by the graph, the theoretical results agree with the experimental results.

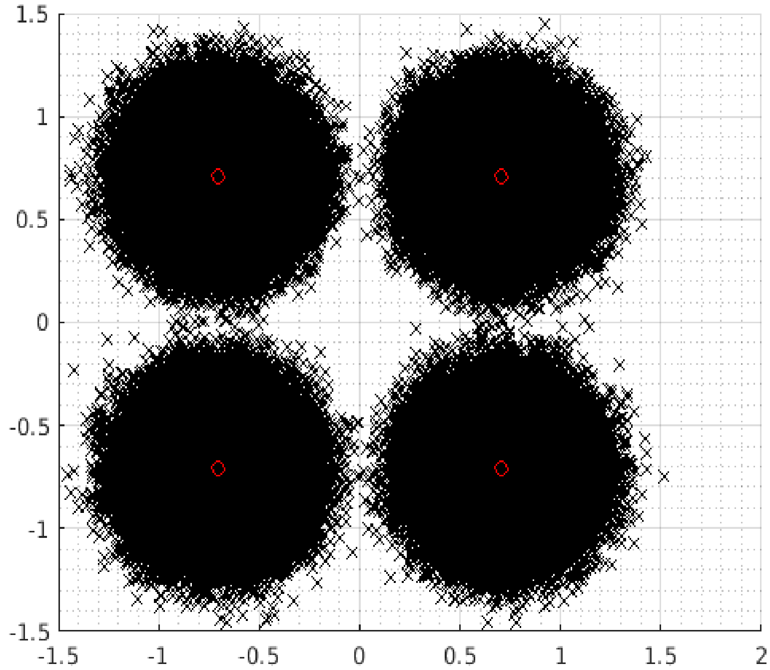

Figure 4: This figure represents the noisy data that is received by the detector. The distribution around the true signal (red circles) is due to the noise added by the system. Here it can be seen how too much of an error will confuse the detector and cause an error.

Summary and Conclusions:

In closing, the most efficient scheme is QPSK. This is due to the fact that it can transmit twice as much information as OOK and BPSK, while achieving the same BER as BPSK (which is better than OOK). The BER for all schemes over a range of SNR values is given by Figure 5. The figure clearly depicts that the OOK has the worst performance (in terms of transfer rate vs. BER), while QPSK and BPSK are the same.

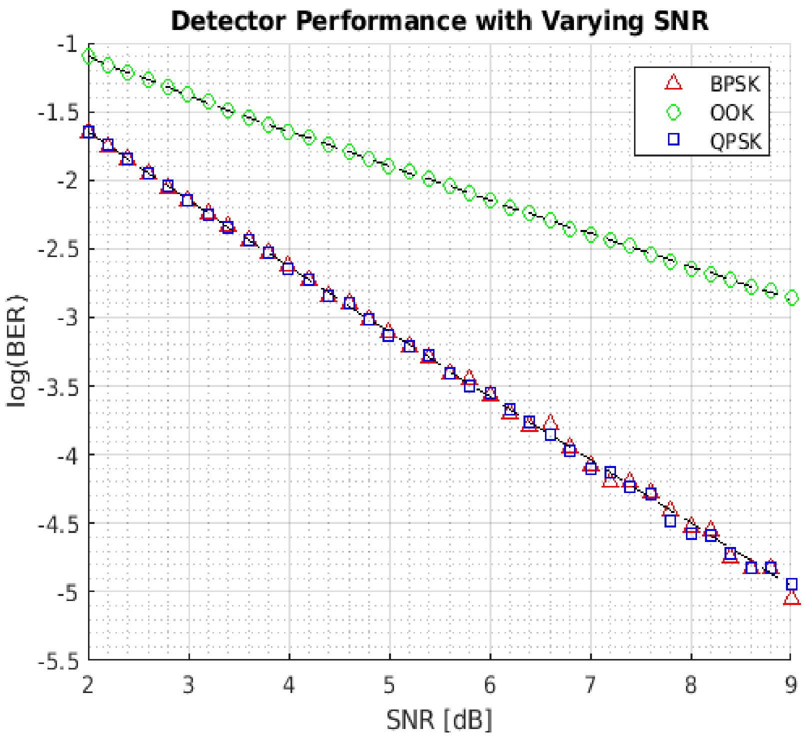

Figure 5: This figure represents the relationship between SNR and BER. Here it is clear that the BPSK and QPSK schemes outperform the OOK scheme in terms of BER. Also, it is clear that the BPSK and QPSK schemes have the same performance per bit.