EEE 558 Project ISI Channels

Beibei LIU (1210467866)

November 30, 2016

1 Introduction

Equalization is the reversal of distortion incurred by a signal transmitted through a channel. MMSE(Minimum Mean Square Error) equalizer is one of the linear filters. It designs the filter to minimize $E[|e|^2]$, where e is the error signal, which is the filter output minus the transmitted signal.

In this experiment, we are going to simulate transmission of the BPSK signals, comparing the performances of MMSE equalizers of different lengths and under different channel models.

1.1 channel model

We consider the transmission of a block of $L+1$ unknown information symbols $s[l], l = 0, ..., L$, and assume $s[l]$ is known for $l < 0$ and $l > L$.

The transmitted symbol sequence vector is

$$\mathbf{s} = [s[0], s[1], ..., s[L]]$$

Consider the following ISI channel

$$x[l] = \sum_{k=0}^{K} h[k]s[l-k] + v[l], \quad l = 0, ..., L \tag{1}$$

where $h[k]$ depends on the transmit filters g_T and receive filters g_R as well as the multipath propagation channel c, i.e. $h[k] = g_T[k] * c[k] * g_R^*[-k]$. $s[l-k] \in \{+1, -1\}$ are i.i.d. BPSK symbols. $v[l]$ are i.i.d. complex-Gaussian AWGN with zero-mean and variance σ_v^2, independent of $s[l]$.

We assume $g_T[k]$, $c[k]$ and $g_R[k]$ are causal, then $h[k] = 0$ holds when $k < 0$. Let K be the length of channel impulse response. We assume that $h[k] \approx 0$ holds when $k > K$.

It is often the case that the channel coefficients $h[k], k = 1...K$ are unknown. Since it is the equalizer that we are interested in, not the channel, it is desirable to avoid estimating the channel, and go directly to estimating the best equalizer coefficients.

1.2 good channel and bad channel

As explained above, we assume the channel impulse response is perfectly known. And in our experiment, $h[k], k = 0, 1, 2$ are fixed, instead of generated randomly (in simulation). Channel fading is not considered.

We try two different channels, one "bad" channel with severe ISI and another "good channel" which is more "delta-like". Performances of the two channels will be compared. Note that the energy of channel $\sum_k |h[k]|^2 = 1$ is normalized.

2 MMSE (Minimum Mean Square Error) linear equalizer

Consider a noncausal equalizer so that

$$\hat{s}[l] = \sum_{m=-K_g}^{K_g} g[m]x[l-m] \tag{2}$$

where $\hat{s}[l]$ is the equalizer output, $2K_g + 1$ is the equalizer length, $g[m]$ is the equalizer coefficient and $x[l-m]$ is the equalizer input. In our experiment, we compare the performance of two different values of equalizer length K_g. One should expect that if K_g is increased, the performance would be better, at the expense of computational complexity. The equalizer is non-causal in general since the equalizer output at time l depends on the channel output $x[\cdot]$ at times greater than l.

2.1 best equalizer design

The goal is to specifying the equalize coefficients $g[m]$ for a (possibly non-causal) FIR equalizer, such that the mean squared error (MSE)

$$E[|s[l-d] - \hat{s}[l]|^2] \tag{3}$$

is minimum, where the expectation is with respect to the noise $v[l]$ and symbol $s[l]$ distributions, d is a delay parameter which can be optimized to improve performance. We assume $d = 0$ for convenience.

Please note that minimizing the MSE does not mean minimizing the BER. Since the BER is a very complicated function of the equalizer coefficients, we do not seek to minimize the BER. So henceforth, by "performance" we will mean MSE with the hope that smaller MSE will probably yield a smaller BER at the end (which is not always the case).

There is an orthogonality principle saying that the error vector of the optimal estimator (in a mean square error sense) is orthogonal to any possible estimator. Thus, in order to minimize the MSE, the error $s[l-d] - \hat{s}[l]$ must be orthogonal to the data $x[p]$ for all values of p and l

$$E[(s[l-d] - \hat{s}[l])x^*[p]] = 0, \quad p = 0, ..., L \tag{4}$$

Multiply both side s of equation 2 by $x^*[p]$, and take expectation over all values of p and l, we obtain

$$
\begin{aligned}
E[\hat{s}[l]x^*[p]] &= \sum_{m=-K_g}^{K_g} g[m]E[x[l-m]x^*[p]] \\
E[(s[l-d]-\hat{s}[l])x^*[p]] + E[\hat{s}[l]x^*[p]] &= \sum_{m=-K_g}^{K_g} g[m]E[x[l-m]x^*[p]] \\
E[s[l-d]x^*[p]] &= \sum_{m=-K_g}^{K_g} g[m]E[x[l-m]x^*[p]] \qquad (5)
\end{aligned}
$$

In this way, by virtue of orthogonality principle, we derive the relation of s and x from the relation of $\hat{s}$ and s. Relation of s and x is useful in the following section.

2.2 correlation

Defining the auto-correlation $r_{xx}[k] = E[x[l]x^*[l+k]]$ and cross-correlation $r_{sx}[k] = E[s[l]x^*[l+k]]$, then equation 5 can be re-express as

$$
\begin{aligned}
E[s[l-d]x^*[p]] &= \sum_{m=-K_g}^{K_g} g[m]E[x[l-m]x^*[p]] \\
r_{sx}[l-d-p] &= \sum_{m=-K_g}^{K_g} g[m]r_{xx}[l-m-p] \\
r_{sx}[d-q] &= \sum_{m=-K_g}^{K_g} g[m]r_{xx}[m-q] \qquad (q := l-p)) \qquad (6)
\end{aligned}
$$

Note that equation 6 is actually a linear system of equations: For each value of q, one obtains a linear equation depending on the equalizer coefficients. Since there are $2K_g+1$ equalizer coefficients, we need at least as many equations. Hence we need to substitute $2K_g+1$ different values of q in equation 6 and solve it for the equalizer coefficients.

Take the values of q ranging from $-K_g$ to K_g

$$
\begin{aligned}
r_{sx}[d-(-K_g)] &= \sum_{m=-K_g}^{K_g} g[m]r_{xx}[m-(-K_g)] \\
r_{sx}[d-(-K_g+1)] &= \sum_{m=-K_g}^{K_g} g[m]r_{xx}[m-(-K_g+1)] \\
... &= ... \\
r_{sx}[d-0] &= \sum_{m=-K_g}^{K_g} g[m]r_{xx}[m-0] \\
... &= ... \\
r_{sx}[d-K_g] &= \sum_{m=-K_g}^{K_g} g[m]r_{xx}[m-K_g]
\end{aligned}
$$

We can put equation 6 in matrix form

$$\mathbf{R}_{xx}\mathbf{g} = \mathbf{r}_{sx} \tag{7}$$

where $\mathbf{R}_{xx}$ is a $(2K_g+1)\times(2K_g+1)$ matrix whose qth row and mth column is $r_{xx}[m-q]$

$$\mathbf{R}_{xx}[m-q] = \begin{bmatrix} r_{xx}[(-K_g)-(-K_g)] & r_{xx}[(-K_g+1)-(-K_g)] & ... & r_{xx}[(K_g-(-K_g)] \\ r_{xx}[(-K_g)-(-K_g+1)] & r_{xx}[(-K_g+1)-(-K_g+1)] & ... & r_{xx}[(K_g-(-K_g+1)] \\ r_{xx}[(-K_g)-(-K_g+2)] & r_{xx}[(-K_g+1)-(-K_g+2)] & ... & r_{xx}[(K_g-(-K_g+2)] \\ ... & ... & ... & ... \\ r_{xx}[(-K_g)-K_g] & r_{xx}[(-K_g+1)-K_g] & ... & r_{xx}[(K_g-K_g)] \end{bmatrix}$$

which can be simplified to

$$\mathbf{R}_{xx}[m-q] = \begin{bmatrix} r_{xx}[0] & r_{xx}[1] & r_{xx}[2] & ... & r_{xx}[2K_g-1] & r_{xx}[2K_g] \\ r_{xx}[-1] & r_{xx}[0] & r_{xx}[1] & ... & r_{xx}[2K_g-2] & r_{xx}[2K_g-1] \\ r_{xx}[-2] & r_{xx}[-1] & r_{xx}[0] & ... & r_{xx}[2K_g-3] & r_{xx}[2K_g-2] \\ r_{xx}[-3] & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & r_{xx}[1] \\ r_{xx}[-2K_g] & r_{xx}[-2K_g+1] & r_{xx}[-2K_g+2] & ... & r_{xx}[-1] & r_{xx}[0] \end{bmatrix}$$

$\mathbf{g}$ is a $(-2K_g+1)\times 1$ column vector

$$\mathbf{g} = \begin{bmatrix} g[-K_g] & ... & g[-K_g] \end{bmatrix}$$

$\mathbf{r}_{sx}$ is also a column vector of the same size with $\mathbf{g}$

$$\mathbf{r}_{sx} = \begin{bmatrix} r_{sx}[d+K_g] & ... & r_{sx}[d-K_g] \end{bmatrix}$$

Then we can solve $\mathbf{g}$ by inverting matrix $\mathbf{R}_{xx}$ and multiplying it with $\mathbf{r}_{sx}$

$$\mathbf{g} = \mathbf{R}_{xx}^{-1}\mathbf{r}_{sx} \tag{8}$$

where the calculations of $\mathbf{R}_{xx}$ and $\mathbf{r}_{sx}$ will be specified in the following session.

2.3 Obtaining correlations from known channels

From the discussion above we know how to find the equalization coefficients vector $\mathbf{g}$ if the auto-correlations $\mathbf{R}_{xx}$ and the cross-correlations $\mathbf{r}_{sx}$ are given. The following part shows how to obtain those correlations.

It is clear that the correlations depend on the channel since the expectations are not with respect to the channel. So if we knew (or estimated) the channel, we could easily get the auto and cross-correlations needed to get $\mathbf{g}$.

To calculate these expectations, we will assume that the noise is independent from the symbols, and that the symbols are i.i.d., which are both reasonable assumptions. The latter assumption might not be so exact in cases where coding or other kinds of dependence is introduced to the input symbols. In addition to these, of course, the noise samples are also i.i.d.

Recall the ISI channel model in equation 1. Here additionally, take $l = l + \tau$, we have

$$x[l] = \sum_{k=0}^{K} h[k]s[l-k] + v[l], \quad l = 0, ..., L$$
$$x[l+\tau] = \sum_{k=0}^{K} h[k]s[l+\tau-k] + v[l+\tau], \quad l = \tau, ..., L+\tau$$

Therefore we obtain

$$\begin{aligned}
r_{xx}[\tau] &= E[x[l]x^*[l+\tau]] \\
&= E\left[\left(\sum_{k=0}^{K} h[k]s[l-k] + v[l]\right)\left(\sum_{k=0}^{K} h[k]s[l+\tau-k] + v[l+\tau]\right)^*\right] \\
&= E\Big[\left(\sum_{k=0}^{K} h[k]h^*[k]s[l-k]s^*[l+\tau-k]\right) + \left(v[l]\sum_{k=0}^{K} h^*[k]s^*[l+\tau-k]\right) \\
&+ \left(v^*[l+\tau]\sum_{k=0}^{K} h[k]s[l-k]\right) + (v[l]v^*[l+\tau])\Big]
\end{aligned}$$

Since symbols and noise are independent, their cross-correlations equal zero. Then the second and third terms are zero.

$$\begin{aligned}
r_{xx}[\tau] &= E\left[\left(\sum_{k=0}^{K} h[k]h^*[k]s[l-k]s^*[l+\tau-k]\right) + (v[l]v^*[l+\tau])\right] \\
&= E\left(\sum_{k=0}^{K} h[k]h^*[k]\right) E\left(\sum_{k=0}^{K} s[l-k]s^*[l+\tau-k]\right) + |v[l]|^2\delta[\tau] \\
&= E\left(\sum_{k=0}^{K} h[k]h^*[k]\right) E\left(\sum_{k=0}^{K} |s[l-k]|^2\delta[\tau]\right) + |v[l]|^2\delta[\tau]
\end{aligned}$$

Since $|s[l-k]|^2 = 1$ and $|v[l]|^2 = \sigma_v^2$ as assumed, we have

$$\begin{aligned}
r_{xx}[\tau] &= \sum_{k=0}^{K} h[k]h^*[k]\delta[\tau] + \sigma_v^2\delta[\tau] \\
&= \sum_{k=0}^{K} h[k]h^*[k+\tau] + \sigma_v^2\delta[\tau] \qquad (9)
\end{aligned}$$

where $\delta[\tau]$ is the discrete Kronecker delta function which is zero when τ is nonzero and one when τ is zero. Additionally, the energy of channel $\sum_k |h[k]|^2 = 1$ is also normalized.

In our experiment, only $h[0], h[1]$ and $h[2]$ are nonzero

$$\begin{aligned}
r_{xx}[-2] &= h[2]h^*[0] \\
r_{xx}[-1] &= h[1]h^*[0] + h[2]h^*[1] \\
r_{xx}[0] &= h[0]h^*[0] + h[1]h^*[1] + h[1]h^*[1] + \sigma_v^2\delta[\tau] \\
r_{xx}[-1] &= h[1]h^*[0] + h[2]h^*[1] \\
r_{xx}[-2] &= h[2]h^*[0] \qquad (10)
\end{aligned}$$

Similarly, for cross-correlation

$$
\begin{aligned}
r_{sx}[\tau] &= E[s[l]x^*[l+\tau]] \\
&= E\left[s[l]\left(\sum_{k=0}^{K} h[k]s[l+\tau-k] + v[l+\tau]\right)^*\right] \\
&= E\left[s[l]\left(\sum_{k=0}^{K} h^*[k]s^*[l+\tau-k] + v^*[l+\tau]\right)\right] \\
&= E\left(\sum_{k=0}^{K} h^*[k]s[l]s^*[l+\tau-k]\right) + E\left(s[l]v^*[l+\tau]\right)
\end{aligned}
$$

Since the symbols and noise are independent, the second term equals zero. For the first term, the expectation is with respect to l (since τ is fixed when calculating correlations), the sum of $h^*[k]$ can be moved out of the expectation. Then we can get

$$
r_{sx}[\tau] = \sum_{k=0}^{K} h^*[k]E\left[s[l]s^*[l+\tau-k]\right]
$$

$E\left[s[l]s^*[l+\tau-k]\right]$ only equals 1 when $\tau - k = 0, i.e. k = \tau$, otherwise equals zero.

$$
r_{sx}[\tau] = h^*[\tau] \tag{11}
$$

In our experiment, nonzero terms are

$$
\begin{aligned}
r_{sx}[0] &= h^*[0] \\
r_{sx}[1] &= h^*[1] \\
r_{sx}[2] &= h^*[2]
\end{aligned}
$$

Then if the channel impulse response h and noise variance σ_v^2 are known, the auto-correlation and cross-correlation can be computed through equation 9 and equation 11, respectively. Thus, we get the equalization coefficients vector $\mathbf{g}$ through equation 8.

3 Experiment results and analysis

We assume fixed $h[k], k = 0, 1, 2$. Take channel impulse responses for "bad", "medium" and "good" channels as

$$
\begin{aligned}
bad: h[0] &= \sqrt{0.083} + \sqrt{0.083}i, h[1] = \sqrt{0.334} + \sqrt{0.334}i, h[2] = \sqrt{0.083} + \sqrt{0.083}i \\
medium: h[0] &= \sqrt{0.075} + \sqrt{0.075}i, h[1] = \sqrt{0.35} + \sqrt{0.35}i, h[2] = \sqrt{0.075} + \sqrt{0.075}i \\
good: h[0] &= \sqrt{0.05} + \sqrt{0.05}i, h[1] = \sqrt{0.4} + \sqrt{0.4}i, h[2] = \sqrt{0.05} + \sqrt{0.05}i
\end{aligned}
$$

which satisfying the energy normalization constraint $h^2[0] + h^2[1] + h^2[2] = 1$.

When evaluating performance, in figure 1, figure 2, figure 3, channel conditions are fixed(bad, medium and good channels, respectively). For a given channel, we plot the Bit Error Rate(in log) against the SNR (in dB), with respect to different equalizer length $2Kg + 1 = 5, 11, 21$. We can see as the length of equalizer increases, the performance improves.

In figure 4, figure 5, figure 6, equalizer lengths are fixed ($Kg = 2, 5, 10$, respectively). For a given equalizer, we plot the Bit Error Rate(in log) against the SNR (in dB), with respect to different channel conditions (bad, medium and good). In addition, in order to see the performance improvement with equalization, we plot both the BER of the communication system without equalizer and BER of the communication system with equalizer. It can be seen clearly that, for a given BER, the required SNR decreases a lot.

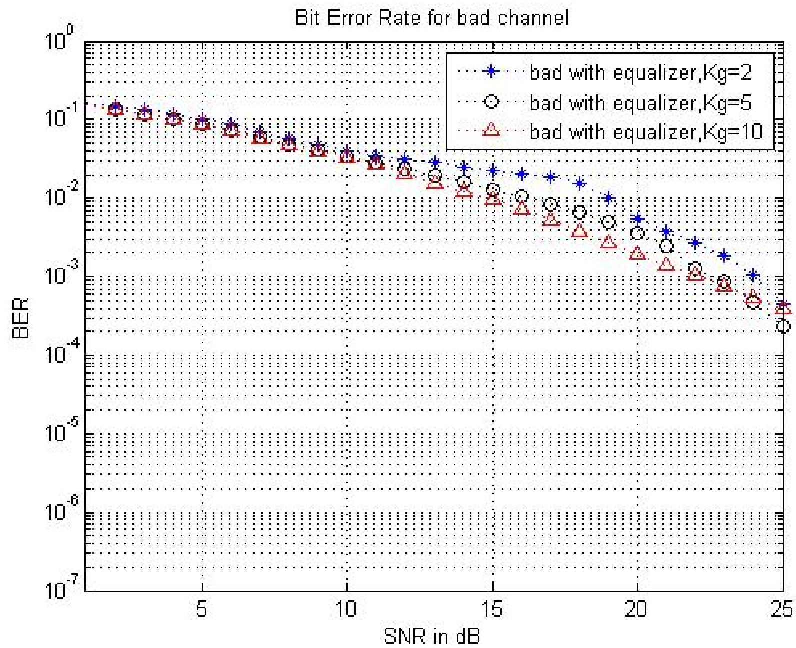

Figure 1: Bit Error Rate for bad channel

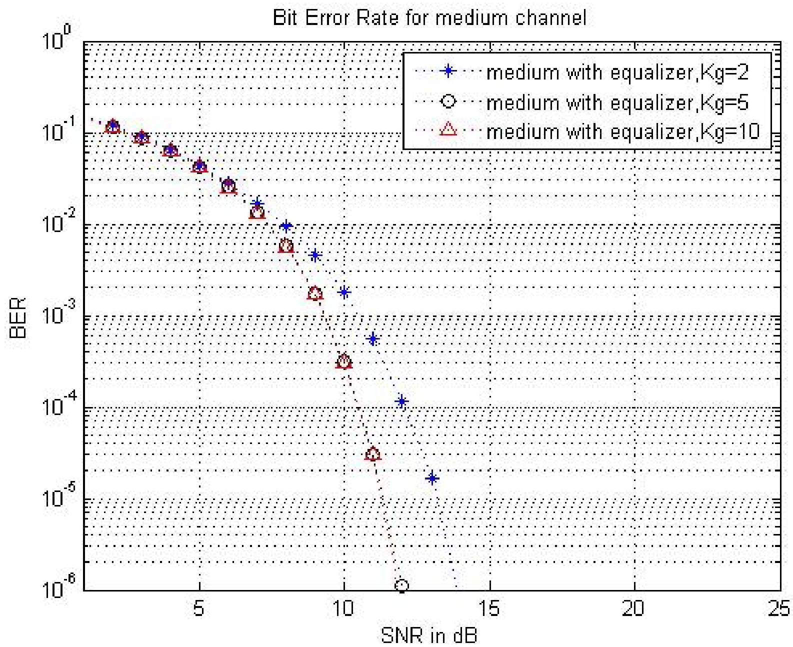

Figure 2: Bit Error Rate for medium channel

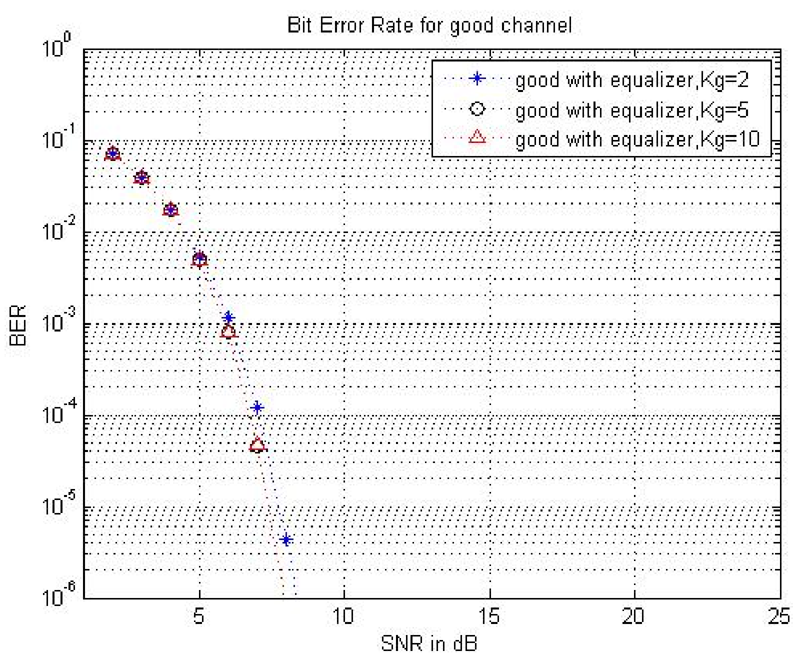

Figure 3: Bit Error Rate for good channel

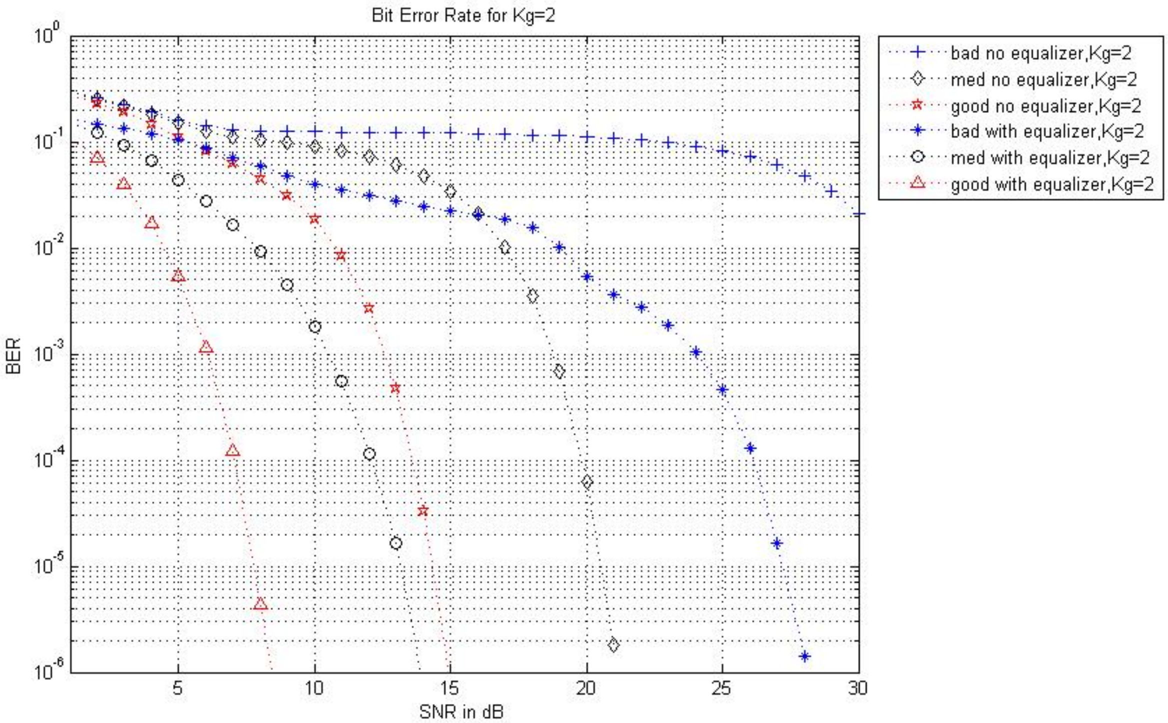

Figure 4: Bit Error Rate when $Kg = 2$

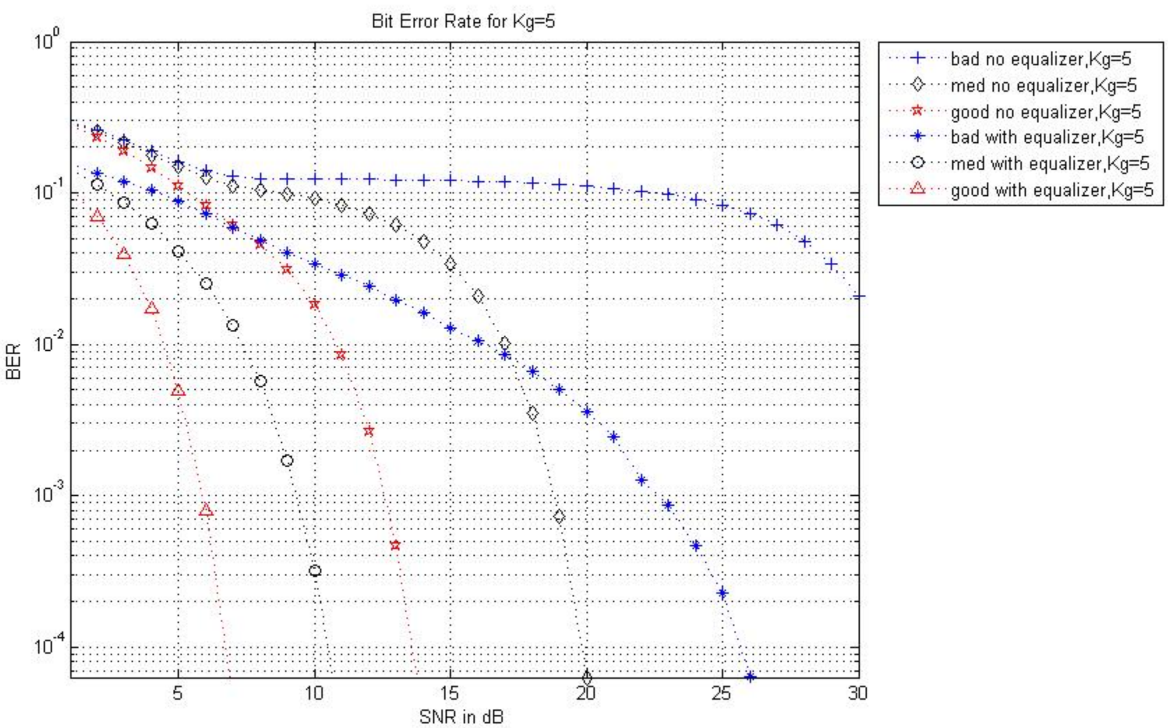

Figure 5: Bit Error Rate when $Kg = 5$

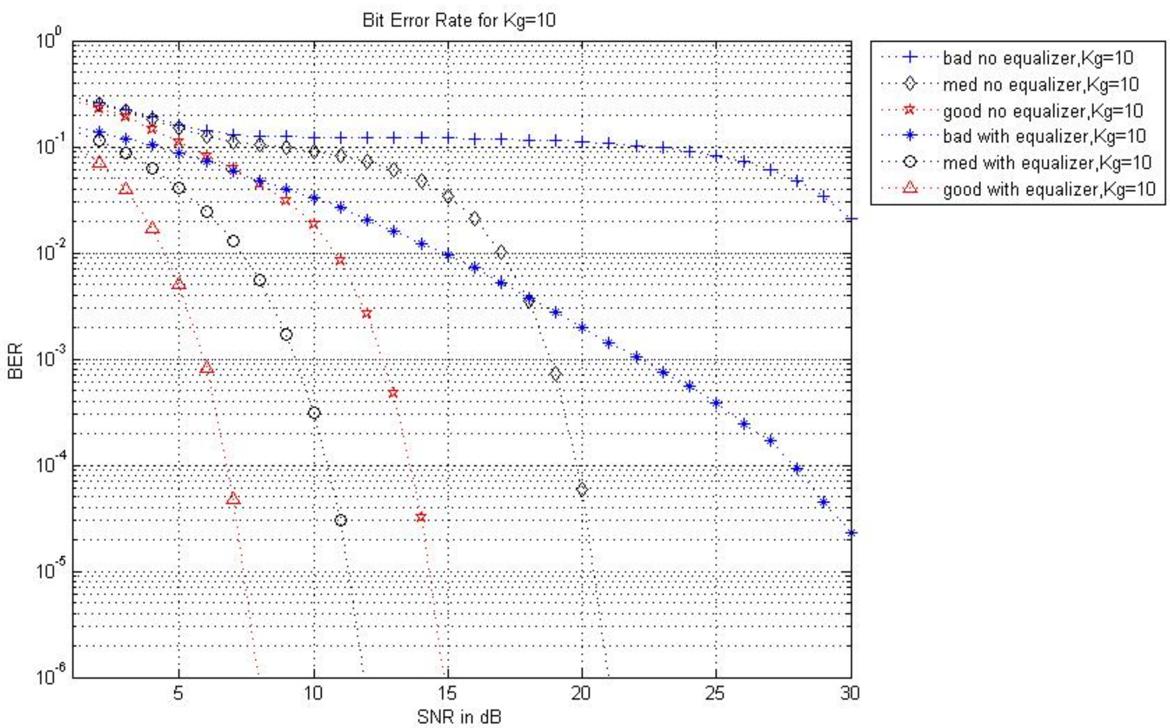

Figure 6: Bit Error Rate when $Kg = 10$

4 attachment: Matlab code

```
% EEE 558 wireless communications project 2: ISI channels
% edited by Beibei LIU (1210467866)
% date:11/20/2016

%%%%%%%%%%%%%%%% main %%%%%%%%%%%%%%%%%%
clear;
%---------- the SNR range (in dB) ----------
MAX_snr_dB=30;
snr_in_dB=0:1:MAX_snr_dB;
%-------- number of loops to run ------
LOOP = 10000;
%---------- initialization before running for loops ---------------
Pe_MMSE_bad_tmp = zeros(length(snr_in_dB),LOOP);
Pe_MMSE_med_tmp = zeros(length(snr_in_dB),LOOP);
Pe_MMSE_good_tmp = zeros(length(snr_in_dB),LOOP);
Pe_MMSE_equal_bad_tmp = zeros(length(snr_in_dB),LOOP);
Pe_MMSE_equal_med_tmp = zeros(length(snr_in_dB),LOOP);
Pe_MMSE_equal_good_tmp = zeros(length(snr_in_dB),LOOP);
Pe_MMSE_bad = zeros(1,length(snr_in_dB));
Pe_MMSE_med = zeros(1,length(snr_in_dB));
Pe_MMSE_good = zeros(1,length(snr_in_dB));
Pe_MMSE_equal_bad = zeros(1,length(snr_in_dB));
Pe_MMSE_equal_med = zeros(1,length(snr_in_dB));
Pe_MMSE_equal_good = zeros(1,length(snr_in_dB));

%---------------- two-layer for cycle ------------------
for i = 1:length(snr_in_dB)
    %-------- probability of error initialization ---------
    avg_bad = 0; avg_med = 0; avg_good = 0;
    avg_equal_bad = 0; avg_equal_med = 0; avg_equal_good = 0;

    for j = 1:LOOP
        %----------- call function:pj2_MMSE: without equalization ------
        % return: probability of error for bad, medium and good chnannel
        [Pe_MMSE_bad_tmp(i,j),Pe_MMSE_med_tmp(i,j),Pe_MMSE_good_tmp(i,j)]=
            pj2_MMSE(snr_in_dB(i));

        %------ call function:pj2_MMSE_equal: with equalization -----
        % return: probability of error for bad, medium and good chnannel
        [Pe_MMSE_equal_bad_tmp(i,j),Pe_MMSE_equal_med_tmp(i,j),
            Pe_MMSE_equal_good_tmp(i,j)]=pj2_MMSE_equal(snr_in_dB(i));

        %----- sum of Pe for all inner loops --------
        avg_bad = avg_bad + Pe_MMSE_bad_tmp(i,j);
        avg_med = avg_med + Pe_MMSE_med_tmp(i,j);
```

```
            avg_good = avg_good + Pe_MMSE_good_tmp(i,j);

            avg_equal_bad = avg_equal_bad + Pe_MMSE_equal_bad_tmp(i,j);
            avg_equal_med = avg_equal_med + Pe_MMSE_equal_med_tmp(i,j);
            avg_equal_good = avg_equal_good + Pe_MMSE_equal_good_tmp(i,j);
        end
        %------ devide the sum to get average probability of error ------
        Pe_MMSE_bad(i) = avg_bad/LOOP;
        Pe_MMSE_med(i) = avg_med/LOOP;
        Pe_MMSE_good(i) = avg_good/LOOP;

        Pe_MMSE_equal_bad(i) = avg_equal_bad/LOOP;
        Pe_MMSE_equal_med(i) = avg_equal_med/LOOP;
        Pe_MMSE_equal_good(i) = avg_equal_good/LOOP;
end;

%==================== plot ========================
%---------- without equalizer -----------
figure(1);
semilogy(snr_in_dB, Pe_MMSE_bad,'+b:');
hold on
semilogy(snr_in_dB, Pe_MMSE_med,'dk:');
hold on
semilogy(snr_in_dB, Pe_MMSE_good,'pr:');
hold on
%legend('bad channel,Kg=5','medium channel,Kg=5','good channel,Kg=5');
grid on;
xlim([1 MAX_snr_dB]);
xlabel('SNR in dB');
ylabel('BER');

%---------- with equalizer -----------
% figure(2);
semilogy(snr_in_dB, Pe_MMSE_equal_bad,'*b:');
hold on
semilogy(snr_in_dB, Pe_MMSE_equal_med,'ok:');
hold on
semilogy(snr_in_dB, Pe_MMSE_equal_good,'^r:');
legend('bad no equalizer,Kg=2','medium no equalizer,Kg=2','good no equalizer,Kg
    =2','bad with equalizer,Kg=2','medium with equalizer,Kg=2','good with
    equalizer,Kg=2');
grid on;
xlim([1 MAX_snr_dB]);
xlabel('SNR in dB');
ylabel('BER');
title('Bit Error Rate');
```

```
%%%%%%%%%%%%%% pj2_MMSE_equal:with equalization %%%%%%%%%%%%%%
function [Pe_MMSE_equal_bad,Pe_MMSE_equal_med,Pe_MMSE_equal_good]=pj2_MMSE_equal
    (snr_in_dB)
% ---------------- parameters ----------------------
SNR=10.^(snr_in_dB/10);    % SNR:snr in linear
E = 1;                     % E:energy per signal bit(symbol,same in BPSK)
sgma = E/SNR;              % sgma:standard deviation of noise

%K = 3;                    % K:length of the non-zero channel impulse response
        % please note:due to no access to h(0), we change K=2 to K=3 instead
Kg = 10;                   % 2*K_g+1:equalizer length

% -------------- original signal ----------------
L = 1000;                           % L:simulation length (transmitted signal length)
rnd = rand(1,L);                    % rnd:random numbers ranging from 0 to 1
s = 1*(rnd>0.5)+(-1)*(rnd<=0.5);    % s: {+1,-1} sequence,original signal

% ------- channel impulse response (fixed, length=3)--------
% ---- bad channel (with severe ISI)
h_bad(1) = sqrt(0.083)+1j*sqrt(0.083);
h_bad(2) = sqrt(0.334)+1j*sqrt(0.334);
h_bad(3) = sqrt(0.083)+1j*sqrt(0.083);
% --- medium channel:
h_med(1) = sqrt(0.075)+1j*sqrt(0.075);
h_med(2) = sqrt(0.35)+1j*sqrt(0.35);
h_med(3) = sqrt(0.075)+1j*sqrt(0.075);
% ---- good channel (more delat-like)
h_good(1) = sqrt(0.05)+1j*sqrt(0.05);
h_good(2) = sqrt(0.4)+1j*sqrt(0.4);
h_good(3) = sqrt(0.05)+1j*sqrt(0.05);

% ------------ noise ---------
v = (randn(1,L)+1j*randn(1,L))*sgma/sqrt(2);          % zero-mean, variance=1
    Gaussian noise

% ------- signal going through channe ---------
x_bad_1 = conv(s, h_bad, 'same');
x_med_1 = conv(s, h_med, 'same');
x_good_1 = conv(s, h_good, 'same');

x_bad = x_bad_1 + v;
x_med =x_med_1+v;
x_good = x_good_1+v;

% **************************** equalizer *******************************
% ============================ Rxx bad ================================
r_xx_bad(1) = h_bad(3)*h_bad(1)';                  % actually r_xx_bad(-2)
```

```
r_xx_bad(2) = h_bad(2)*h_bad(1)'+h_bad(3)*h_bad(2)';    % actually r_xx_bad(-1)
r_xx_bad(3) = h_bad(1)*h_bad(1)'+h_bad(2)*h_bad(2)'+h_bad(3)*h_bad(3)'+sgma;    %
    actually r_xx_bad(0)
r_xx_bad(4) = h_bad(1)*h_bad(2)'+h_bad(2)*h_bad(3)';    % actually r_xx_bad(1)
r_xx_bad(5) = h_bad(1)*h_bad(3)';                       % actually r_xx_bad(2)

R_xx_bad = zeros(2*Kg+1,2*Kg+1);
for ii = -Kg:Kg
    for jj = -Kg:Kg
        switch ii-jj
            case -2
                R_xx_bad(ii+Kg+1,jj+Kg+1) = r_xx_bad(1);
            case -1
                R_xx_bad(ii+Kg+1,jj+Kg+1) = r_xx_bad(2);
            case 0
                R_xx_bad(ii+Kg+1,jj+Kg+1) = r_xx_bad(3);
            case 1
                R_xx_bad(ii+Kg+1,jj+Kg+1) = r_xx_bad(4);
            case 2
                R_xx_bad(ii+Kg+1,jj+Kg+1) = r_xx_bad(5);
        end
    end
end

% ------------------------ rsx bad ------------------------
r_sx_bad = zeros(1,2*Kg+1);
r_sx_bad(Kg+1) = h_bad(1)';    % actually r_sx_bad(0)
r_sx_bad(Kg+2) = h_bad(2)';    % actually r_sx_bad(1)
r_sx_bad(Kg+3) = h_bad(3)';    % actually r_sx_bad(2)

% ======================== Rxx med ========================
r_xx_med(1) = h_med(3)*h_med(1)';                       % actually r_xx_bad(-2)
r_xx_med(2) = h_med(2)*h_med(1)'+h_med(3)*h_med(2)';    % actually r_xx_bad(-1)
r_xx_med(3) = h_med(1)*h_med(1)'+h_med(2)*h_med(2)'+h_med(3)*h_med(3)'+sgma^2;
    % actually r_xx_bad(0)
r_xx_med(4) = h_med(1)*h_med(2)'+h_med(2)*h_med(3)';    % actually r_xx_bad(1)
r_xx_med(5) = h_med(1)*h_med(3)';                       % actually r_xx_bad(2)

R_xx_med = zeros(2*Kg+1,2*Kg+1);
for ii = -Kg:Kg
    for jj = -Kg:Kg
        switch ii-jj
            case -2
                R_xx_med(ii+Kg+1,jj+Kg+1) = r_xx_med(1);
            case -1
                R_xx_med(ii+Kg+1,jj+Kg+1) = r_xx_med(2);
            case 0
```

```
                R_xx_med(ii+Kg+1,jj+Kg+1) = r_xx_med(3);
            case 1
                R_xx_med(ii+Kg+1,jj+Kg+1) = r_xx_med(4);
            case 2
                R_xx_med(ii+Kg+1,jj+Kg+1) = r_xx_med(5);
        end
    end
end

% ---------------------- rsx med ------------------------
r_sx_med = zeros(1,2*Kg+1);
r_sx_med(Kg+1) = h_med(1)';     % actually r_sx_med(0)
r_sx_med(Kg+2) = h_med(2)';     % actually r_sx_med(1)
r_sx_med(Kg+3) = h_med(3)';     % actually r_sx_med(2)

% ========================= Rxx good ================================
r_xx_good(1) = h_good(3)*h_good(1)';                     % actually r_xx_bad(-2)
r_xx_good(2) = h_good(2)*h_good(1)'+h_good(3)*h_good(2)';    % actually r_xx_bad
    (-1)
r_xx_good(3) = h_good(1)*h_good(1)'+h_good(2)*h_good(2)'+h_good(3)*h_good(3)'+
    sgma^2;    % actually r_xx_bad(0)
r_xx_good(4) = h_good(1)*h_good(2)'+h_good(2)*h_good(3)';    % actually r_xx_bad
    (1)
r_xx_good(5) = h_good(1)*h_good(3)';                     % actually r_xx_bad(2)

R_xx_good = zeros(2*Kg+1,2*Kg+1);
for ii = -Kg:Kg
    for jj = -Kg:Kg
        switch ii-jj
            case -2
                R_xx_good(ii+Kg+1,jj+Kg+1) = r_xx_good(1);
            case -1
                R_xx_good(ii+Kg+1,jj+Kg+1) = r_xx_good(2);
            case 0
                R_xx_good(ii+Kg+1,jj+Kg+1) = r_xx_good(3);
            case 1
                R_xx_good(ii+Kg+1,jj+Kg+1) = r_xx_good(4);
            case 2
                R_xx_good(ii+Kg+1,jj+Kg+1) = r_xx_good(5);
        end
    end
end

% ---------------------- rsx good -----------------------
r_sx_good = zeros(1,2*Kg+1);
r_sx_good(Kg+1) = h_good(1)';     % actually r_sx_good(0)
r_sx_good(Kg+2) = h_good(2)';     % actually r_sx_good(1)
```

```
r_sx_good(Kg+3) = h_good(3)';      % actually r_sx_good(2)

% ========= calculate equalizer coefficients g ==================
g_bad = inv(R_xx_bad)*r_sx_bad.';
g_med = inv(R_xx_med)*r_sx_med.';
g_good = inv(R_xx_good)*r_sx_good.';

% ------- the received signal going through equalizer -----------

s_equal_bad = conv(x_bad,g_bad,'same');
s_equal_med = conv(x_med,g_med,'same');
s_equal_good = conv(x_good,g_good,'same');

% --------- decision --------------
% --- initialization ----
s_hat_bad = zeros(1,length(SNR));
s_hat_med = zeros(1,length(SNR));
s_hat_good = zeros(1,length(SNR));
for ii = 1:L
    if real(s_equal_bad(ii)) > 0
        s_hat_bad(ii) = 1;
    else
        s_hat_bad(ii) = -1;
    end

    if real(s_equal_med(ii)) > 0
        s_hat_med(ii) = 1;
    else
        s_hat_med(ii) = -1;
    end

    if real(s_equal_good(ii)) > 0
        s_hat_good(ii) = 1;
    else
        s_hat_good(ii) = -1;
    end
end

% --------------- BER ----------------
ave_err_bad = 0;
for ii = 1:L-1
    if s_hat_bad(ii+1)~=s(ii)
        ave_err_bad = ave_err_bad + 1;
    end
end
Pe_MMSE_equal_bad = ave_err_bad/L;
```

```
ave_err_med = 0;
for ii = 1:L-1
    if s_hat_med(ii+1)~=s(ii)
        ave_err_med = ave_err_med + 1;
    end
end
Pe_MMSE_equal_med = ave_err_med/L;

ave_err_good = 0;
for ii = 1:L-1
    if s_hat_good(ii+1)~=s(ii)
        ave_err_good = ave_err_good + 1;
    end
end
Pe_MMSE_equal_good = ave_err_good/L;
```