LMS Adaptive Filter

B Swaroop Reddy and Dr G V V Sharma*

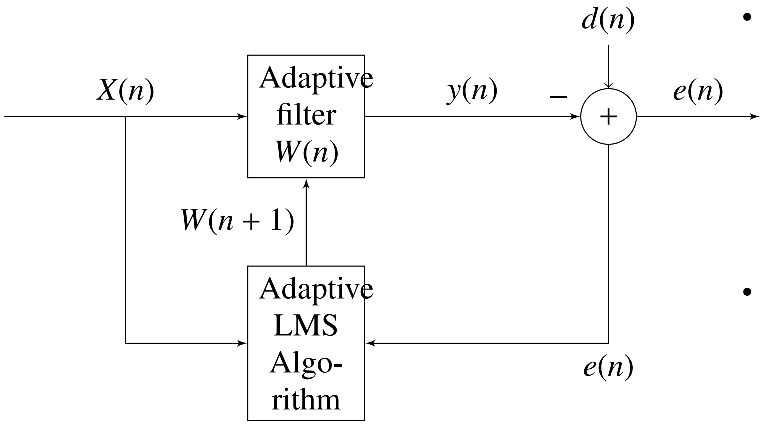

Fig. 1.0: The Adaptive LMS filter

Contents

***Abstract*—This manual explains about how the Adaptive Least-Mean-Square Algorithm can be used for noise-cancellation in the speech signal and as an Equalizer in a communication system.**

1 Least Mean Squares(LMS) Algorithm

Consider the following block diagram(Fig.1.0) for the structure of the adaptive LMS filter. Where

*The author is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in. All content in this manual is released under GNU GPL. Free and open source.

•

$$X(n) = \begin{bmatrix} X(n) \\ X(n-1) \\ X(n-2) \\ .. \\ .. \\ X(n-M+1) \end{bmatrix}_{MX1} \tag{1.0.1}$$

•

$$W(n) = \begin{bmatrix} w_1(n) \\ w_2(n) \\ w_3(n) \\ .. \\ .. \\ w_{n-M+1}(n) \end{bmatrix}_{MX1} \tag{1.0.2}$$

Problem 1.1. Write the expression for the error from Fig1.0

Solution:

$$\begin{aligned} e(n) &= d(n) - y(n) \\ &= d(n) - W^T(n)X(n) \\ &= d(n) - X^T(n)W(n) \end{aligned}$$

Problem 1.2. Write the expression for the Mean-Squared error(MSE) from Problem1.1.

Solution:

$$\begin{aligned} J &= E[e^2(n)] \\ &= E[(d(n) - W^T(n)X(n))^2] \\ &= E[(d(n) - X^T(n)W(n))^T(d(n) - X^T(n)W(n))] \\ &= E[d(n)d(n)] - W^T(n)E[X(n)d(n)] - E[d(n)X^T(n)]W(n) + W \\ &= r_{dd} - W^T(n)r_{xd} - r_{xd}^T W(n) + W^T(n)RW(n) \end{aligned}$$

Problem 1.3. Find the optimal solution from the cost function in Problem1.2

Solution: By Equating the derivative of J w.r.to W(n) to zero

$$
\begin{aligned}
\frac{\partial J(n)}{\partial W(n)} &= 0 \\
0 - r_{xd} - r_{xd} + 2RW(n) &= 0 \\
RW(n) &= r_{xd} \\
W(n) &= R^{-1} r_{xd}
\end{aligned}
$$

This is called Wiener-Hopf equation.

$$W_*(n) = R^{-1} r_{xd}$$

This is the Wiener optimal solution.

Problem 1.4. Do you think the above solution is Adaptive?If Yes give the explanation? If not how can you make the above solution adaptive?

Solution: The solution is non adaptive, since we are assuming that R and r_{xd} are known.
By making the above solution independent of the covariance and cross covariance and the solution will become adaptive
Depending on the data update the weights

$$
\begin{aligned}
NewWeightsW(n+1) = OldWeightsW(n) + \\
(functionofnewdataX(n) \\
andthedesiredsignald(n))
\end{aligned}
$$

Typical approaximation is based on the samples X(n)
Approximate $E[e^2(n)]$ based on samples i.e.

$$E[e^2(n)] \simeq \frac{1}{N} \sum_{i=0}^{N-1} e^2(n)$$

Widrow's idea : Approaximate $E[e^2(n)]$ by just one sample i.e.

$$
\begin{aligned}
E[e^2(n)] &\simeq e^2(n) \\
&= (d(n) - W^T(n)X(n))^2 \\
&= (d(n) - X^T(n)W(n))^T (d(n) - X^T(n)W(n)) \\
&= d(n)d(n) - W^T(n)X(n)d(n) - \\
&\quad d(n)X^T(n)W(n) + W^T(n)X(n)X^T(n)W(n)
\end{aligned}
$$

Where

- X(n) is most recent data
- W(n) is weight at time instant n
- d(n) desired signal at time instant n

Differentiate $e^2(n)$ and equate it to zero-

$$
\begin{aligned}
\frac{\partial e^2(n)}{\partial W(n)} &= 0 \\
0 - 2X(n)d(n) + 2X(n)X^T(n)W(n) &= 0 \\
X(n)X^T(n)W(n) &= X(n)d(n)
\end{aligned}
$$

We cann't solve the above equation. Because $X(n)X^T(n)$ is rank one matrix,thus cann't be inverted.

Iterative Scheme(Gradient Descent Algorithm):

$$
\begin{aligned}
NewWeights = OldWeights + Additional \\
termdependentonthe - vegradient \\
W(n+1) = W(n) + \bar{\mu}[-\nabla_{W(n)} J(n)]
\end{aligned}
$$

Where

- $J(n) = e^2(n)$ is the cost function
- $\nabla_{W(n)} J(n)$ is the gradient of J(n) w.r.to to W(n)
- In this case J(n) is convex, it will converge to Global optimum.

Summary of the LMS Adaptive Algorithm :

$$
\begin{aligned}
\nabla_{W(n)} J(n) &= \frac{\partial J(n)}{\partial W(n)} \\
&= \frac{\partial e2(n)}{\partial W(n)} \\
&= -2X(n)d(n) + 2X(n)X^T(n)W(n) \\
W(n+1) &= W(n) + \bar{\mu}[-\nabla_{W(n)} J(n)] \\
W(n+1) &= W(n) + \bar{\mu}[-(-2X(n)d(n) + 2X(n)X^T(n)W(n))] \\
W(n+1) &= W(n) + \mu X(n)[d(n) - X^T W(n)] \\
W(n+1) &= W(n) + \mu X(n)e(n)
\end{aligned}
$$

Where $\mu = 2\bar{\mu}$

2 Convergence of LMS Algorithm

2.1 Convegenge in Mean for W(n)

Problem 2.1. How can we choose the value of μ if LMS algorithm converges in mean i.e. $\lim_{n\to\infty} \tilde{W}(n) = 0$, where $\tilde{W}(n) = W(n) - W_*$.

Solution: $0 < \mu < \dfrac{2}{M(SignalPower)}$

Problem 2.2. Prove the result of the Problem 2.1.

Solution:

$$\begin{aligned}\tilde{W}(n+1) &= W(n+1) - W_* \\ &= W(n) + \mu X(n)[d(n) - X^T(n)W(n)] - W_* \\ &= \tilde{W}(n) + \mu X(n)[d(n) \\ &- X^T(n)(\tilde{W}(n) + W_*)] \\ &= \tilde{W}(n) + \mu X(n)[d(n) \\ &- X^T(n)\tilde{W}(n) + X^T(n)W_*] \\ &= [I - \mu X(n)X^T(n)]\tilde{W}(n) \\ &+ \mu X(n)[d(n) - X^T(n)W_*]\end{aligned}$$

Taking the expectation on both sides

$$\begin{aligned}E[\tilde{W}(n+1)] &= E[\tilde{W}(n)] - \mu E[X(n)X^T(n)\tilde{W}(n)] \\ &+ \mu E[X(n)d(n) - X(n)X^T(n)W_*] \\ \\ &= E[\tilde{W}(n)] - \mu R E[\tilde{W}(n)] + \mu[r_{xd} - RW_*] \\ &= E[\tilde{W}(n)] - \mu R E[\tilde{W}(n)] + 0 \\ &= [I - \mu R]E[\tilde{W}(n)]\end{aligned}$$

Since R is Symmetric and positive semidefinite, $R = U\Lambda U^T$, Where $U = [u_1, u_2, ..., u_M]$ and $\lambda_i > 0$ for i =1,2,3,...,M

$$\begin{aligned}E[\tilde{W}(n+1)] &= [I - \mu U\Lambda U^T]E[\tilde{W}(n)] \\ &= [UIU^T - \mu U\Lambda U^T]E[\tilde{W}(n)] \\ &= U[I - \mu\Lambda]U^T E[\tilde{W}(n)] \\ E[U^T\tilde{W}(n+1)] &= [I - \mu\Lambda]E[U^T\tilde{W}(n)]\end{aligned}$$

$$= \begin{bmatrix} 1-\mu\lambda_1 & 0 & 0 & .... & 0 \\ 0 & 1-\mu\lambda_2 & 0 & .... & 0 \\ . & . & . & .... & . \\ . & . & . & .... & . \\ . & . & . & .... & . \\ 0 & 0 & 0 & .... & 1-\mu\lambda_M \end{bmatrix} E[U^T\tilde{W}(n)]$$

$$\begin{aligned}&| 1 - \mu\lambda_i |< 1, i = 1, 2, 3, ..., M \\ &-1 < 1 - \mu\lambda_i < 1 \\ &0 < \mu < \frac{2}{\lambda_i} \\ &0 < \mu < \frac{2}{\lambda_{max}} < \frac{2}{\lambda_i}\end{aligned}$$

Estimating λ_i

$$\begin{aligned}\lambda_{max} &= Mr(0) \\ &= ME[x(n)x(n)] \\ &= \frac{M}{N}\sum_{i=0}^{N-1} x^2(n) \\ &= M(SignalPower)\end{aligned}$$

$$0 < \mu < \frac{2}{M(SignalPower)}$$

2.2 Convegenge in Mean-square sense

Problem 2.3. How can we choose the value of μ if LMS algorithm converges in mean-square sense.

Solution: $0 < \mu < \dfrac{1}{M(SignalPower)}$

Problem 2.4. Prove the result of the Problem 2.3.

Solution:

$$\begin{aligned}J(n) &= E[e^2(n)] \\ &= E[(d(n) - W^T(n)X(n))^2] \\ &= E[(d(n) - W^T(n)X(n))^T(d(n) - W^T(n)X(n))] \\ &= E[(d(n) - W_*^T X(n) - W^T(n)X(n) + W_*^T X(n))^T \\ &(d(n) - W_*^T X(n) - W^T(n)X(n) + W_*^T X(n))] \\ &= E[(e_*^2(n) - \tilde{W}(n)X(n))^T(e_*^2(n) - \tilde{W}(n)X(n))] \\ &= E[e_*^2(n)] + E[\tilde{W}(n)X(n)X(n))^T\tilde{W}(n)] - \\ &E[\tilde{W}(n)X(n)e_*(n)] - E[e_*(n)X^T(n)(n)\tilde{W}(n)]\end{aligned}$$

Where

$$\begin{aligned}\tilde{W}(n) &= W(n) - W_* \\ e_*(n) &= d(n) - W_*X(n)\end{aligned}$$

$$\begin{aligned}&E[e_*(n)X^T(n)(n)\tilde{W}(n)] \\ &= E[(d(n) - W_*^T X(n)X^T(n)(n)\tilde{W}(n))] \\ &= E[d(n)X^T(n)(n)\tilde{W}(n)] - E[W_*^T X(n)X^T(n)(n)\tilde{W}(n)] \\ &= [r_{xd} - W_*^T R]\tilde{W}(n) = 0\end{aligned}$$

$$\begin{aligned}J(n) &= E[e_*^2(n)] + E[\tilde{W}(n)X(n)X(n))^T\tilde{W}(n)] \\ &= J_{min}(n) + J_{ex}(n)\end{aligned}$$

$$
\begin{aligned}
&E[\tilde{W}^T(n)X(n)X^T(n)\tilde{W}(n)] \\
&\quad = E[\sum_{i=1}^{M}\sum_{j=1}^{M}\tilde{W}_i(n)x_i(n)x_j(n)\tilde{W}_j(n)] \\
&\quad = \sum_{i=1}^{M}\sum_{j=1}^{M}E[\tilde{W}_i(n)x_i(n)]E[x_j(n)\tilde{W}_j(n)] \\
&\quad = E[\sum_{i=1}^{M}\sum_{j=1}^{M}\tilde{W}_i(n)E[x_i(n)x_j(n)]\tilde{W}_j(n)] \\
&\quad = E[\tilde{W}^T(n)R\tilde{W}(n)] \\
J(n) &= J_{min} + E[\tilde{W}^T(n)R\tilde{W}(n)]
\end{aligned}
$$

Since R is Symmetric and positive semidefinite, $R = U\Lambda U^T$, Where $U = [u_1, u_2, ..., u_M]$ and $\lambda_i > 0$ for i =1,2,3,...,M

$$
\begin{aligned}
J_{ex}(n) &= E[\tilde{W}^T(n)U)(U^T\tilde{W}(n))] \\
&= \sum_{i=1}^{M}\lambda_i E[\tilde{W}_u^i\tilde{W}_u^i] \\
&= \sum_{i=1}^{M}\lambda_i p_u^{ii}(n)
\end{aligned}
$$

$$
\begin{aligned}
&\tilde{W}(n+1) \\
&\quad = [I - \mu X(n)X^T(n)]\tilde{W}(n) + \mu X(n)[d(n) - X^T(n)W_*]
\end{aligned}
$$

Let $P(n) = E[\tilde{W}(n)\tilde{W}^T(n)]$
and $P_u(n) = U^T P(n)U$
$p_u^{ii}(n)$ is the ii^{th} diagonal entry of $P_u(n)$
$p_u^{ii}(n+1) = [1 - 2\mu\lambda_i]p_u^{ii}(n) + \mu^2 J_{min}\lambda_i$

$$
\begin{aligned}
|\, 1 - 2\mu\lambda_i \,| &< 1 \\
-1 < 1 - 2\mu\lambda_i &< 1 \\
0 < \mu &< \frac{1}{\lambda_i} \\
0 < \mu < \frac{1}{\lambda_{max}} &< \frac{1}{\lambda_i} \\
0 < \mu &< \frac{1}{M(SignalPower)}
\end{aligned}
$$

Problem 2.5. Find the value of the cost function at infinity i.e. $J(\infty)$

Solution:

$$
\begin{aligned}
p_u^{ii}(\infty) &= [1 - 2\mu\lambda_i]p_u^{ii}(\infty) + \mu^2 J_{min}\lambda_i \\
&= \frac{\mu J_{min}}{2} \\
J_{ex}(\infty) &= \sum_{i=1}^{M}\lambda_i p_u^{ii}(\infty) \\
&= \sum_{i=1}^{M}\lambda_i\frac{\mu J_{min}}{2} \\
&= \frac{\mu J_{min}}{2}\sum_{i=1}^{M}\lambda_i \\
&= \frac{\mu J_{min}}{2}tr(R) \\
&= \frac{\mu J_{min}}{2}M(SignalPower) \\
J(\infty) &= J_{min} + \frac{\mu J_{min}}{2}\sum_{i=1}^{M}\lambda_i
\end{aligned}
$$

Problem 2.6. How can you choose the value of μ from the convergence of both in mean and mean-square sense.

Solution:

1) For the convergence in mean, we require
$$0 < \mu < \frac{2}{M(SignalPower)}$$
2) For the convergence in mean-square sense, we require
$$0 < \mu < \frac{1}{M(SignalPower)}$$

$\therefore$ choose μ such that $0 < \mu < \dfrac{1}{M(SignalPower)}$

3 Adaptive Noise Cancellation

You will need two data streams: one corresponding to signal plus noise and the other corresponding to noise.

Consider the following M-th order FIR adaptive structure for noise cancellation.

Assume that $X(n)$ and S(n) are statistically independent.Let $e(n) = S(n) + X(n) - \hat{X}(n)$.Now suppose the adaptation rule is chosen such that $E[e^2(n)]$ is minimized.

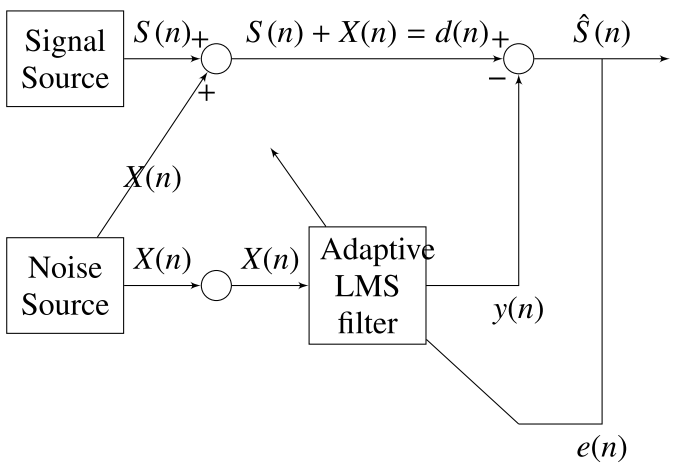

Fig. 3.0: LMS Noise Cancellation System

Problem 3.1. Download the speech and noise files from

The files are for speech plus noise, and only noise. You will notice that you cannot decipher the speech when you play the speech plus noise file; but upon proper implementation of the noise cancellation algorithm you will be able to decipher the speech.

Problem 3.2. Using the speech samples in Problem 2.1 write a python script for the Adaptive noise cancellation using LMS algorithm.

Solution:

```
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf


M = 10  # No. of weights of the
    filter
w = np.zeros(M)  # Initializing
    the filter weights
mu = 0.0001  # step size

#Reading the speech and noise data
d,fs = sf.read('Record-001.wav')
v,fs = sf.read('Record-002.wav')
#print(v)
#print(d)
#print(fs)
#print np.var(v)

#In case of number of samples data
    is not equal
if(len(d) <= len(v)):
        N = len(d)
else:
        N = len(v)
#print(N)

e = np.zeros(N) # error signal
y = np.zeros(N) # Output of the
    filter
u = np.concatenate((np.zeros(M-1),
    v))

# LMS Algorithm
for i in range(N):
        x = np.flipud(u[i:i+M])
        #print(np.shape(z))
        #d = np.flipud(x[i:i+
            filtlen])
        y[i] = np.sum(w*x)
        e[i] = d[i] - y[i]
        #print(e[i])
        w += mu*e[i]*x   #Update
            weights

S_hat = e
sf.write('output_signal_lms1.wav',
    S_hat,fs)
```

Problem 3.3. The output of the python script in Problem **??** is the speech signal output_ signal_ lms.wav.Listen to the the speech signal. What do you observe?

4 Adaptive Equalization

LMS algorithm for channel equalization using training signals:
Consider the following block diagram for the adaptive channel equalizer using training sequences.

In the above we are assuming that the transmitted symbols are binary (bpsk) with zero mean $b_n = \pm 1$. These are passed through a channel with impulse

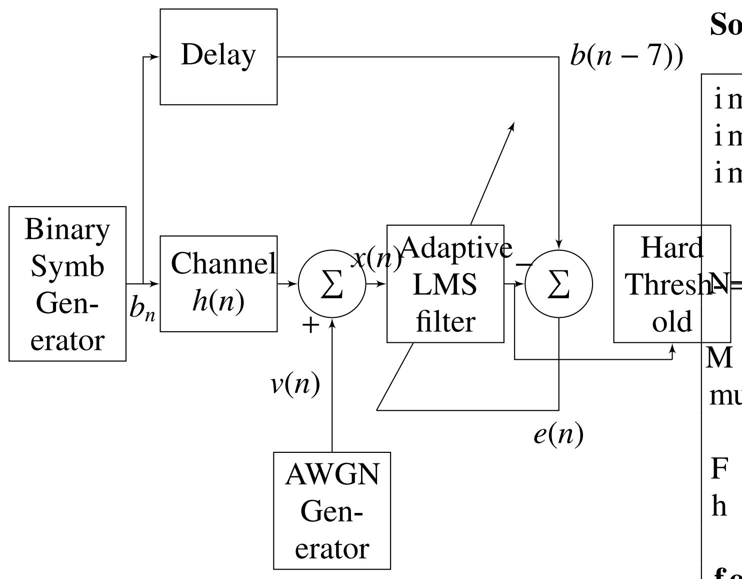

Fig. 4.0: LMS Equalization System

response

$$h(n) = \begin{cases} \frac{1}{2}[1 + cos(\frac{2\pi}{F}(n-2))] & n = 1,2,3 \\ 0 & otherwise \end{cases}$$

and corrupted by additive white Gaussian noise (AWGN) with zero mean and variance σ^2. The delayed symbols that are fed at the output of the adaptive filter act as training signals.The channel impulse response is an 3-point FIR filter with a raised cosine type of structure.Parameter F controls the eigen value spread of $R = E[X(n)X^T(n)$.
Note:

$$x(n) = \sum_{k=1}^{3} h(k)b_{n-k} + v(n)$$

You will need two random number generators; one for the symbols b n and the other for the AWGN.For AWGN assume $\sigma^2 = 0.001$. For b_n use a binary random number generator where 1 and -1 are generated with equal probability. You could do this by using a uniform random number generator, uniform over [0.5, 0.5]. Whenever you get a negative number, set it to -1 and whenever you get a positive number set it to +1.

Problem 4.1. Write a python script to plot the mean square error curve (Iterations Vs MSE) for the Equalization structure shown in fig:3.0 Run the experiment for two values of F, say $F = 3$ and $F = 3.2$.

Solution:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy


N=int(1000)  # Number of input
    samples
M = 7  # No. of taps
mu = 0.075 # Step size

F = 3.0
h = np.zeros(4)  # Channel Impulse
     response
for n in range(1,len(h)):
        h[n] = 0.5*(1+np.cos(2*np.
              pi*(n-2)/F))
print h

#b=np.zeros(N)
#b_hat=np.zeros(N)
ip=np.random.randint(2,size=N)
b=2*ip-1
#print b
bh = np.convolve(h,b)
#print bh

e = np.zeros(N) # Error signal
y = np.zeros(N) #Output signal
J = np.zeros(N) #Mean Square Error
w = np.zeros(M) # Filter
    Coefficients

noise = np.sqrt(0.001)*np.random.
    randn(len(bh))
#print noise
d = bh+noise
#print d

#LMS algorithm for Equalization
for j in range(M,N):
        x = np.flipud(d[j-M:j])

        y[j] = np.dot(w,x)

        e[j] = b[j-7] - y[j]

        w += mu*e[j] * x
        J[j] = (e[j])**2.0


#print w
plt.plot(J)
plt.grid()
plt.ylabel('MSE')
plt.xlabel('Iterations')
plt.title('Mean Square Error Curve
    ,F=3.0, $\mu$ =0.075 ,delay=7 ')
plt.savefig('Learning_curve.eps')
plt.show()
```

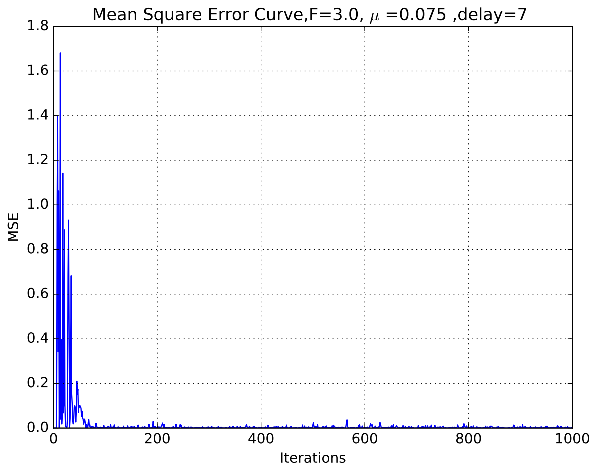

Fig. 4.1

Problem 4.2. Repeat the problem 4.1 with different number of tap weights in the FIR LMS adaptive filter. For example you could consider 9, or 11 tap weights.Estimate the amount of delay you need based on the number of tap weights you are using. Comment.

Problem 4.3. Repeat the problem 4.1 with different μ=0.025,0.05,0.0075. What do you observe?

Problem 4.4. Repeat the problem 4.1 with different variance for AWGN. What do you observe?

Problem 4.5. At the output of the adaptive filter have a hard threshold unit to classify the output as 1 or -1. This is done since we know that the transmitted symbols were 1 or -1. Now write a python script to compute the percentage bits that are in error at convergence (this will give you the bit error rate (BER)) with different SNRs. Plot the SNR Vs BER curve.

Solution:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import erfc

def lms(N,M,mu,sigma,bh,b):
        J = np.zeros(N)
        J_avg = np.zeros(N)
        e = np.zeros(N)
        y = np.zeros(N)
        noise = sigma*np.random.
            randn(len(bh))
        d = bh+noise
        w = np.zeros(M)
        for j in range(M,N):
                u = np.flipud(d[j-
                    M:j])
                y[j] = np.dot(w,u)
                e[j] = b[j-7] - y[
                    j]
                w += mu*e[j] * u
                J[j] += (e[j])
                    **2.0


        return y,J,w


N=int(1e6)
M = 7 # No. of taps
mu = 0.025 # Step size
itr = 50
F = 3.0
snrlen=11
Eb_N0_dB=np.arange(0,snrlen)
sigma=np.sqrt(0.5*(10**(-Eb_N0_dB
    /10.0)))
h = np.zeros(4)
for n in range(1,len(h)):
        h[n] = 0.5*(1+np.cos(2*np.
            pi*(n-2)/F))
print h
b=np.zeros(N)
b_hat=np.zeros(N)
simBersoft=np.zeros(snrlen)
nErr=np.zeros(snrlen)

for i in range(snrlen):
        ip=np.random.randint(2,
                    size=N)
            b=2*ip-1
            bh = np.convolve(h,b)
            y,J,w = lms(N,M,mu,sigma[i
                    ],bh,b)
            #print w
            b_hat = 2*(y>=0)-1
            nErr[i] = 0
            for j in range(M,N):
                        if b[j-M] != b_hat
                            [j] :
                                nErr[i]
                                    +=1


theoryBer=0.5*erfc(np.sqrt((10**(
    Eb_N0_dB/10.0))))

simBersoft=nErr/float(N)
#print simBersoft
plt.plot(Eb_N0_dB,theoryBer,'r',
    Eb_N0_dB,simBersoft,'b')
plt.legend(['theory','Practical'],
    loc=1)
#plt.plot(Eb_N0_dB,simBersoft,'b')
plt.yscale('log')
plt.ylabel('BitErrorRate')
plt.xlabel('Eb/N0␣in␣dB')
plt.title('BER␣for␣BPSK␣in␣raised␣
    cosine␣pulse␣channel␣with␣LMS␣
    Adaptive␣Equalizer␣')
plt.grid()
plt.savefig('SNR_Vs_BER.eps')
plt.show()
```

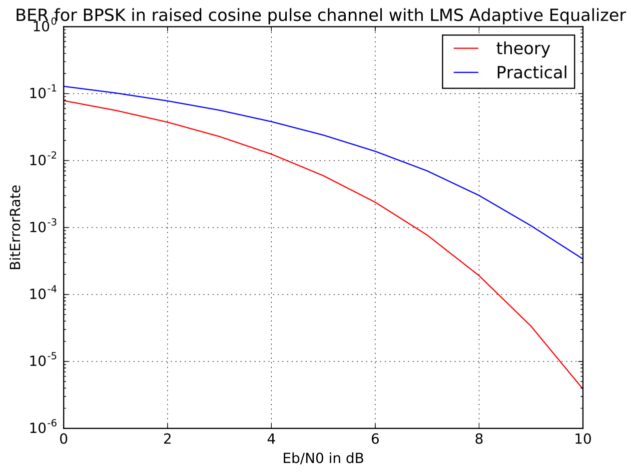

Fig. 4.5

Problem 4.6. The channel is no longer as given by the cosine function. But, now let the channel be modeled by an FIR filter of length 10, and each FIR coefficient be of unit magnitude. Repeat the above channel equalization problem for the equalizer under a narrow band channel.Plot the bit error rate.

Problem 4.7. The data symbols are not binary (BPSK) but are QPSK. You can generalize the method used for generating BPSK to a method for generating QPSK. Plot the bit error rate. Do this for both the cases: (a) the FIR channel specified above, and (b) any other narrow band channel of your choice.