Problem Set #1
ECE-595, Section II
Spring 2013, Adaptive Filtering
Date Assigned: 02/07/2013
Date Due: 02/21/2013

Background

In class, we discussed the popular LMS algorithm and several of its variants. In this MATLAB exercise, we will simulate several of the variants of the LMS:

- Normalized LMS algorithm (NLMS)
- Adaptive step-size LMS (from your text)
- Leaky LMS algorithm
- Adaptive step-size NLMS.
- Data-reuse NLMS algorithm

We specifically wish to study their performance in predicting a second-order autoregressive process, i.e., AR(2) process, where we will be deploying the LMS as an adaptive predictor. Towards this purpose, first write four MATLAB function `nlms.m`, `aslms.m`, `llms.m`, `asnlms.m`, `drnlms.m` that implement the different variants. I have uploaded some of these functions to the course webpage. For studying the tracking performance of the algorithms, generate a random sequence that has a different set of AR coefficients over two separate, non-overlapping halves of the same signal.

Comparison of Performance

Compare the performance of the standard LMS algorithm with that of the leaky LMS algorithm for two different step-size factors and two different leakage factors. Accomplish this by comparing the average tap-weights and the MSE, over 100 experiments. For the second part, compare the performance of the standard LMS with the two adaptive step-size algorithms and the NLMS. Again evaluate their performance by comparing the average tap-weight behavior and the MSE for both small and larger step-sizes for both: (a) low SNR but stationary and (b) time-varying AR coefficients and higher SNR environments. Document your observations in terms of the advantages or problems of the different approaches.

1 MATLAB functions

1.1 LMS Algorithm

```
%****************************************************
%  MATLAB Implementation of the LMS Algorithm
%  AUTHOR: Balu Santhanam
%  DATE  : 03/25/03
%
%  SYNOPSIS :
%  [filco,y,MSE] = lms_co(u,d,mu,L,init)
%****************************************************
function [filco,y,MSE] = lms_co(u,d,mu,L,init)
if nargin == 4
   init = zeros(1,L);
elseif nargin < 4
```

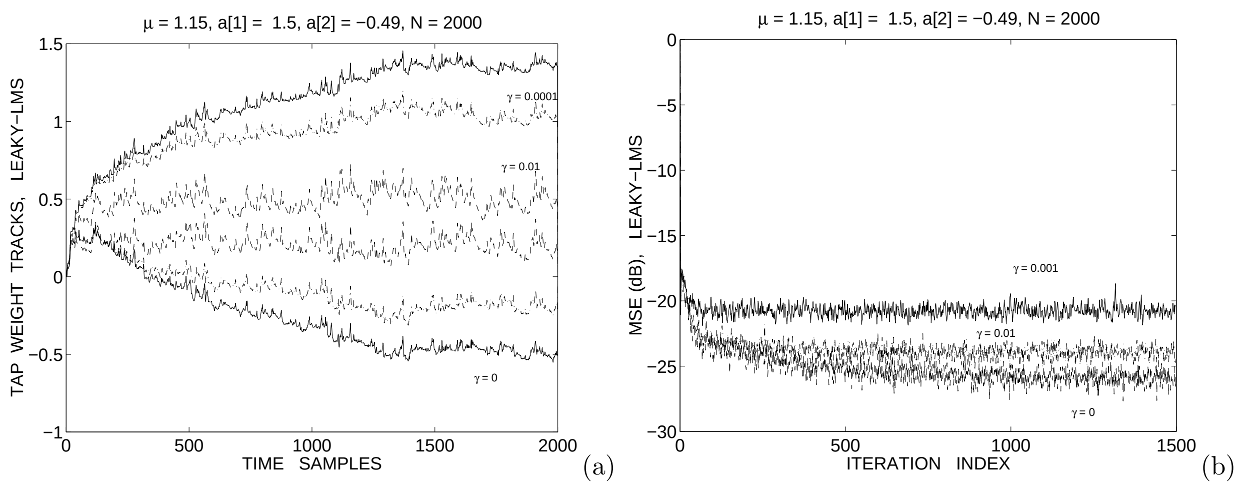

Figure 1: Comparison of the leaky-LMS with the LMS: (a) tap-weight tracks of the leaky-LMS for $\mu = 1.15$ for different γ factors and (b) corresponding MSE curves for different leakage factors. The leaky-LMS produces a biased solution while the LMS algorithm, i.e., $\gamma = 0$ will converge to the optimal Wiener solution.

```
   error('Insufficient number of input parameters')
end
X = convmtx(u,L).'; % Form Toeplitz convolution matrix
filco(1,:) = init(:).'; n = length(u);
for p = 2:n-L+1
    y(p) = filco(p-1,:)*X(p,:).';
    e(p) = d(p) - y(p);
    filco(p,:) = filco(p-1,:) + mu*e(p)*conj(X(p,:));
    MSE(p) = e(p)^2;
end
%*****************************************************
```

1.2 Leaky LMS Algorithm

```
%*****************************************************
%  MATLAB Implementation of the LMS Algorithm
%  AUTHOR: Balu Santhanam
%  DATE  : 03/25/03
%
%  SYNOPSIS :
%  [filco,y,MSE] = leak_lms(u,d,mu,gam,L,init)
%*****************************************************
function [filco,y,MSE] = leak_lms(u,d,mu,gam,L,init)
if nargin == 5
   init = zeros(1,L);
elseif nargin < 5
   error('Insufficient number of input parameters')
end
X = convmtx(u,L).'; % Form Toeplitz convolution matrix
filco = zeros(length(d),L);
filco(1,:) = init(:).'; n = length(u);
MSE = zeros(1,length(d)); MSE(1) = 1;
for p = 2:n-L+1
    y(p) = filco(p-1,:)*X(p,:).';
    e(p) = d(p) - y(p);
    filco(p,:) = (1-mu*gam)*filco(p-1,:) + mu*e(p)*conj(X(p,:));
```

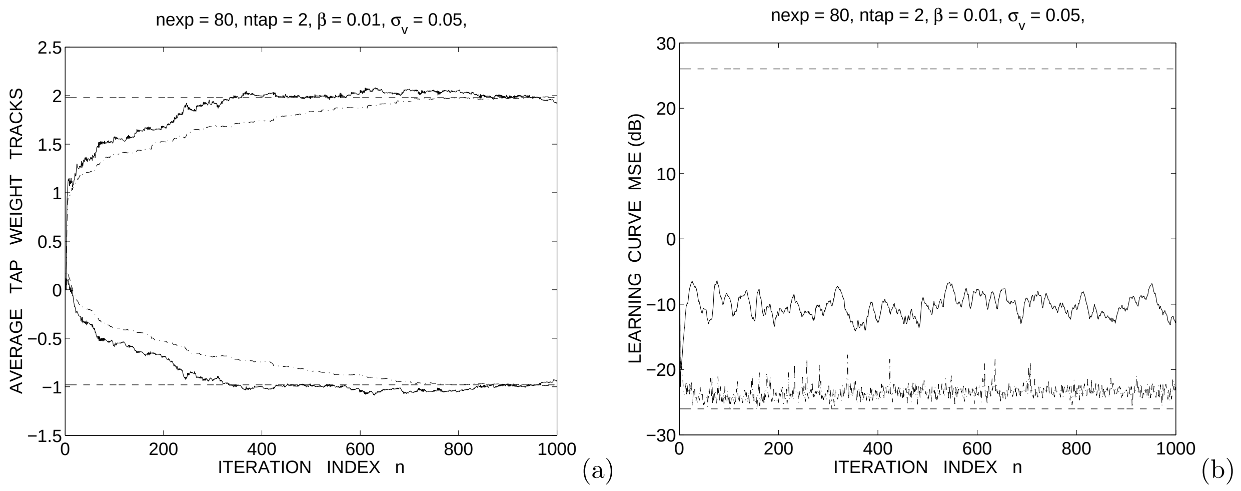

Figure 2: Performance of the NLMS: (a) average tap-weight tracks for the NLMS algorithm obtained by averaging over 80 experiments, and (b) learning curves for different step-size parameters.

```
    MSE(p) = e(p)^2 + gam*norm(filco(p,:),2)^2;
end
%----------------------------------------------------------
```

1.3 Normalized LMS Algorithm

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This function is an implementation of the
%   NLMS algorithm and returns the tap weights
%
%    AUTHORS  :  Santhanam Balasubramaniam
%    DATE     :  04/19/94
%
%    USAGE    : [filco,y,MSE] = nlms_co(x,d,alfa,bet,ntap)
%    filco    : Filter Coefficients matrix
%    err      : Estimation error
%    MSE      : Mean Square Error (dB)
%    x        : Observation Process (1,N)
%    d        : desired process (1,N)
%    ntap     : Number of taps in the filter
%    alfa    : Step size parameter (0 - 2)
%    bet     : Offset parameter (typ: 0.01)
%    y        : Filtered output
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [filco,y,MSE] = nlms_co(x,d,alfa,bet,ntap)
format lon
x = x(:).'; d = d(:).';
N = length(x); X = convmtx(x,ntap).';
W(1,:) = zeros(1,ntap); [M,N] = size(X);
out = zeros(1,N); MSE(1) = 1;
for k = 2:1:M-ntap+1
    mu = alfa / ((X(k,:) * X(k,:).') + bet);
    out(k) = W(k-1,:)*X(k,:).';
    error(k) = d(k) - out(k);
    W(k,:) = W(k-1,:) + mu*conj(X(k,:))*error(k);
    MSE(k) = error(k)^2;
end
```

```
filco = W;
y = out;
return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

1.4 Variable Step Size NLMS

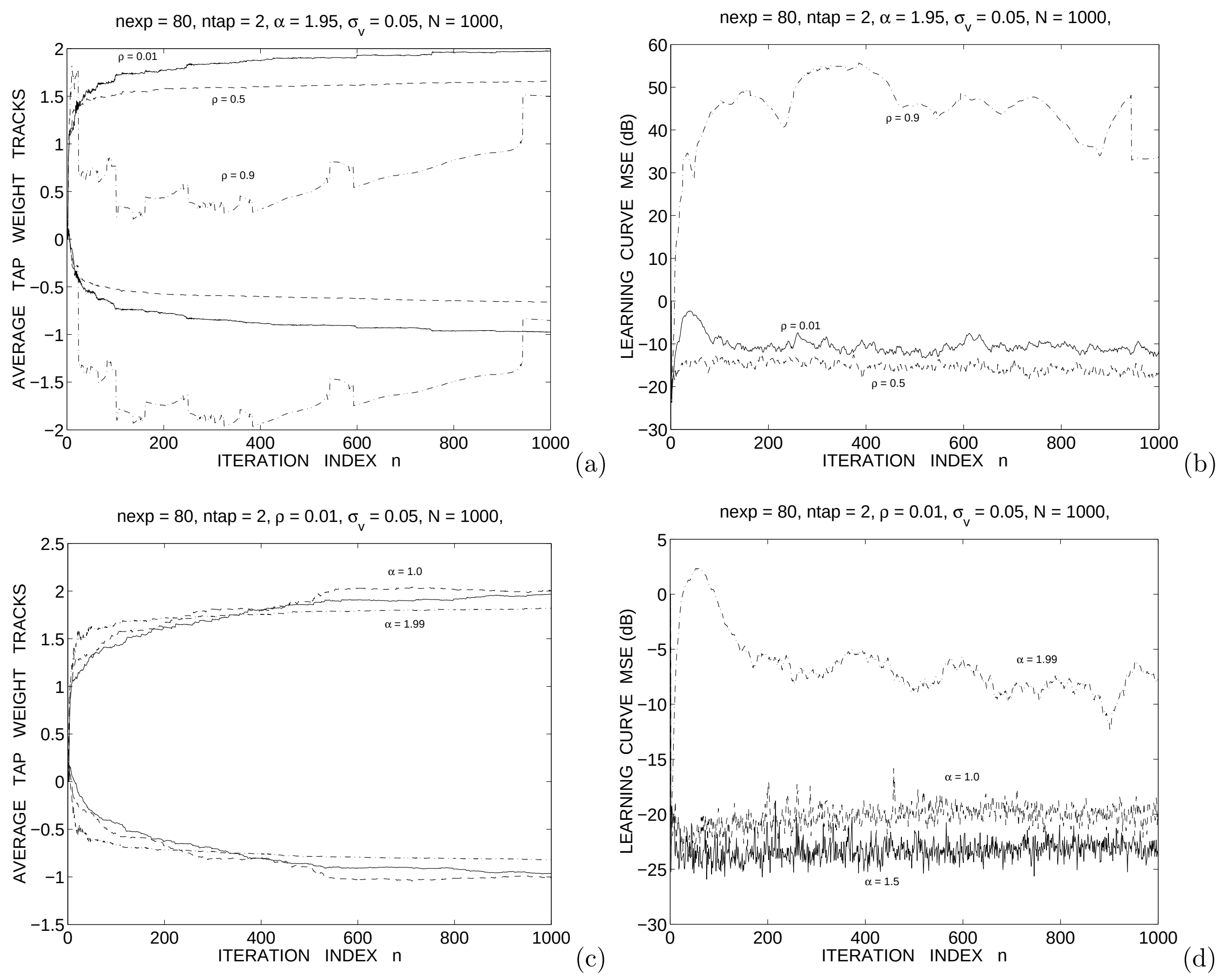

Figure 3: Performance of the variable step-size NLMS: (a,b) average tap-weights and learning curves for a fixed α for different learning rates, (c,d) average tap-weights and learning curves for a fixed learning rate for different α. Note the divergence of the tap-weights when the learning rate is too high.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This function is an implementation of the
%   generalized NLMS algorithm (SPL-04)
%
%    AUTHORS  :  Santhanam Balasubramaniam
%    DATE     :  10/05/05
%
%    USAGE [filco,out,MSE,eta] = gnlms(x,d,alpha,rho,ntap);
%
%    ifreq1,ifreq2 : estimated IF tracks
%    eta           : learning rate
%    MSE           : Instantaneous Mean Square Error (dB)
```

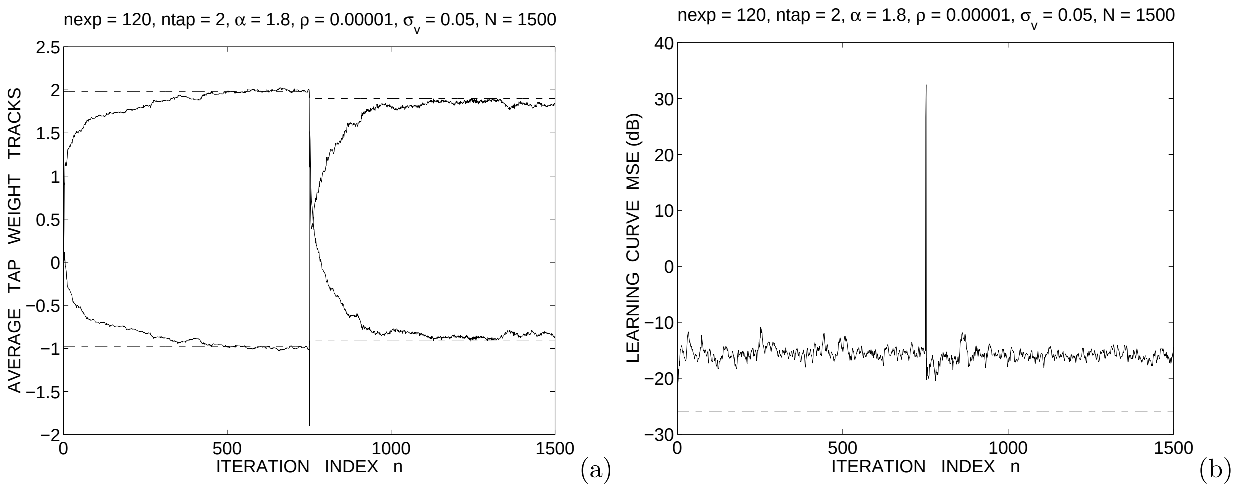

Figure 4: Tracking performance of the variable step-size NLMS: (a) average tap-weight tracks for the NLMS algorithm obtained by averaging over 120 experiments, and (b) learning curves for different step-size parameters. The AR coefficients over the first half of the observations are different from the AR coefficients during the second half. Note the switch in the tap-weights at the mid-point of the signal duration and the eventual readaptation.

```
%    ntap          : Number of taps in the filter
%    alpha         : Step size parameter (0 - 2)
%    rho           : Offset parameter learning rate (typ: 0-1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [filco,out,MSE,eta] = gnlms(x,d,alpha,rho,ntap)
format long
x = x(:); d = d(:);
N = length(x); X = convmtx(x,ntap);
% Initialization
W(1,:) = zeros(1,ntap);
[M,N] = size(X); beta(1) = 0.01;
out = zeros(1,N); MSE(1) = 1;
for k = 2:1:M-ntap+1
    out(k) = W(k-1,:)*X(k,:).';
    e(k) = d(k) - out(k);
    cfactor(k) = e(k)*e(k-1)*X(k,:)*X(k-1,:).';
    num = rho*alpha*cfactor(k);
    den = X(k-1,:)*X(k-1,:).' + beta(k-1);
    beta(k) = beta(k-1) - num/den;
    clear num den cfactor out
    eta(k) = alpha / ((X(k,:) * X(k,:).') + beta(k));
    W(k,:) = W(k-1,:) + eta(k)*conj(X(k,:))*e(k);
    MSE(k) = e(k)^2;
end
filco = W;
clear beta e
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

1.5 Data Reuse Normalized LMS Algorithm

In situations, where the signal environment is correlated, noisy and non-stationary, the data-reuse LMS may be beneficial. Simulate an observation process that has a high degree of correlation between samples. Write a MATLAB function `drnlms.m` that implements the data-reuse version of the NLMS. Compare the performance of the NLMS with and without data reuse for $L = 3$.