EE 504

Homework #4
Due: April 28, 2004

P.1: (Hayes, Prob. 9.3) Newton's method is an iterative algorithm that may be used to find the minimum of a nonlinear function. Applied to the minimization of the mean-square error

$$\varepsilon(n) = \frac{1}{2}E\{e^2(n)\}$$

where $e(n) = d(n) - \mathbf{w}^T\mathbf{x}(n)$, Newton's method is

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mathbf{R}_x^{-1}\nabla\varepsilon(n)$$

where $\mathbf{R}_x$ is the autocorrelation matrix of $x(n)$. Introducing a step-size paramter μ, Newton's method becomes

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu\mathbf{R}_x^{-1}\nabla\varepsilon(n)$$

Comparing this to the steepest descent algorithm, we see that the step size μ is replaced with matrix, $\mu\mathbf{R}_x^{-1}$ which alters the descent direction.

a) For what values of μ is Newton's method stable ?

b) What is the optimum value of μ, i.e. for what value of μ is the convergence fastest ?

c) Suppose that we form an LMS version of Newton's method by replacing gradient with a gradient estimate $\hat{\nabla}\varepsilon(n) = \nabla\varepsilon(n)$. Derive the coefficient update equation that results from using this gradient estimate and describe how it differs from the LMS algorithm.

d) Derive an expression that describes the time evolution of $E\{\mathbf{w}_n\}$ using the LMS Newton algorithm derived in part c).

P.2: Interference cancellation
At some medical measurements, such as the EKG measurements, the 220V 50-Hz signal may leak into the sensors connected to the patient. A method of minimizing the effect of power line leakage is to use adaptive filters. In this experiment you will implement a simulation to to cancel out a sinosoid with unknown phase and magnitude, given a reference sinosoid.

a) Generate 200 samples of $r[n] = A\cos(\pi/10n + \phi)$ where $A = 4$ and $\phi = \pi/4$. ($r[n]$ is the powerline interference.)

b) Assume that we have access to the line voltage. The line voltage is, of course, out of phase and at different magnitude from $r[n]$. Let line voltage $x[n] = cos(\pi/10n)$. Analytically show that we can find $\mathbf{w}_1$ and $\mathbf{w}_2$ coeffs. such that $r[n] = \mathbf{w}_1 x[n] + \mathbf{w}_2 x[n-1]$.

c) Using the given llms.m file implement an adaptive filter with input $x[n]$, and desired signal $d[n] = r[n]$. Examine whether filter converges to the coefficients found at part b).

Try different step-size values. Plot a representetive learning curve (Matlab:plot(A)) and error curve (Matlab:semilogy(E)). Try different A and ϕ values. Convince yourself that the filter converges for all A and ϕ values.

d) What happens when we use a third order adaptive filter ? Try different initial conditions. Comment on your results.

P.3: (Hayes) **System Identification**
One of the many uses of adaptive filters is for system identification as shown in Figure 1. In this configuration, the same input is applied to the adaptive filter and to an unknown system, and the coefficients of the adaptive filter are adjusted until the difference between the outputs of the two systems is as small as possible. After adaptation, the system function of the unknown system can be approximated by the system function of the adaptive filter. Adaptive system identification can be used to model a system whose parameters are slowly varying, when the input and output signals are both available, for example in vibration studies of mechanical systems. In realistic situations, however the output of the unknown system will generally by distorted by additive noise.

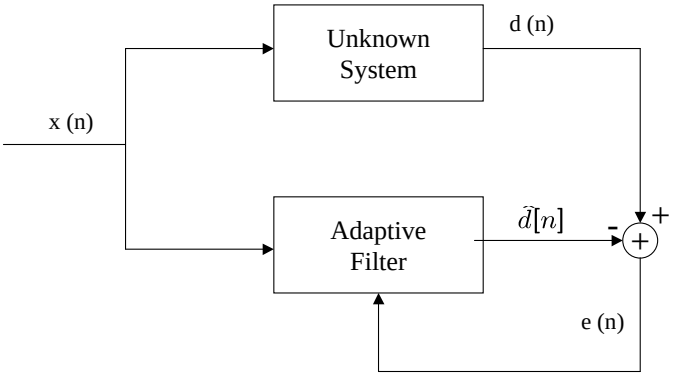

Figure 1: Adaptive Filter for system identification problem

In the first exercise, we look at how the adaptive system identification works in an ideal setting. More realistic conditions are then considered in Exercises 2 and 3.

Exercise 1 Let an unknown system be an FIR filter with the following impulse response,

$$h[n] = \delta(n) + 1.8\delta(n-1) + 0.81\delta(n-2)$$

Using an input signal $x(n)$ consisting of at least 100 samples of unit variance white Gaussian noise, create the reference signal $d(n)$ by passing $x(n)$ through the filter.

a) Determine the range of values for step size μ so that the adaptive filter will be convergent in the mean.

b) Use the given llms.m file to implement the adaptive filter. Set the initial values for the filter coefficients to zero. Set the step size $\mu = 0.5$ and use an adaptive filter of order $N = 4$. (Note you may find that the adaptive filter does not converge. If so, decrease μ until it does). Let the adaptive filter adapt and record the final setting.

c) If the experiment in part b) is performed K times and the error on the kth trial is

$\varepsilon_k(n)$ after n iterations, then the mean squared error may be approximated using

$$E\{\varepsilon(n)\} \cong \frac{1}{K}\sum_{k=1}^{K}\varepsilon_k(n)$$

With K=5, make a plot of the learning curves as a function of n using the approximation defined above. How many iterations are necessary for the mean square error to fall to the 10% of its peak value.

d) Calculate the excess mean square error and compare it to what you observed in your plot of the learning curve.

e) Repeat parts b) and c) for $\mu = 0.1$ and $\mu = 0.05$.

f) Repeat parts b) and c) with $N = 3$ and $N = 4$.

You should observe that the adaptive filter will converge to the proper solution when the unknown system is an FIR filter of order less than or equal to the order of the adaptive filter. The rate of the convergence depends on the step size -the larger the step size, the faster the convergence-. However, when noise is present, a small step size is used to mitigate the effects of the noise. This is explored in the following exercise.

Exercise 2 Consider the system identification problem in Exercise 1, but now suppose that the reference signal is corrupted by noise, i.e.

$$\tilde{d}(n) = d(n) + \gamma v(n)$$

where $v(n)$ is unit variance white Gaussian noise.

a) With $\gamma = 1$, use the LMS algorithm to model the system. Set the initial values to zero and use a step size of $\mu = 0.5$ and an adaptive filter of order 4. Let the filter adapt and record the coefficients. Repeat your experiment using N=5 and comment on your results.

b) Repeat part a) with $\gamma = 0.1$ and comment on how the accuracy of the model varies with the noise amplitude. Conduct some simple experiments to determine whether the modeling accuracy depends on the step size.

Exercise 3 An adaptive FIR filter may be used to model any unknown system. In this problem, we examine how well such a system can model an IIR system. The arrangement for the filter is the same as the one in Figure 1 that was used in the previous two exercise. Let the unknown system be an IIR filter with a system function

$$H(z) = \frac{1 + 0.5z^{-1}}{1 - 0.9z^{-1}} \tag{1}$$

Generate a 300-point Gaussian random sequence with unit variance. Use this signal for the input to both the unknown system and the adaptive filter.

a) Use the file llms.m to implement an adaptive filter with $\mu = 0.3$ and $N = 4$. Record the final filter coefficients. Plot the error as described in exercise 1. How many iterations are necessary for the filter to converge.

b) Repeat a)for different values for the filter length N. What seems to be a reasonable value to use ?

c) Add noise of amplitude A to the output of the unknown system and repeat part a) for $A = \{0.1, 0.3, 1.0\}$.

Matlab files can be downloaded from course website.

```
function [A,E]= llms(x,d,alpha,nord,A_in)
%LMS                          [mhh3 3/92]
%-- Widrow-Hoff LMS adaptive filtering algorithm
% Adapted from lms.m written by j.mcclellan
%
%
% [A,E] = llms(x,d,alpha,nord,a0)
%
%       x : input data to the adaptive filter
%       d : desired output vector
%   alpha : adaptive filtering update (step-size) parameter
%    nord : number of adaptive filter coefficients
%      a0 : (optional) initial guess for FIR filter coeffs
%           (row vector). If a0 is omitted, a0=0 is assumed
%     [A,E] = llms(x,d,alpha) uses A_in=0
%
%     The output matrix A contains filter coefficients
%         - The n'th row contains the filter coefficients at time n
%         - The m'th column contains the m'th filter coeff vs. time
%         - The output vector E contains the error sequences vs. time
%
%

X=convolm(x,nord,'<');
[M,N] = size(X);

if nargin < 5, A_in=zeros(1,N); end;
A_in = A_in(:).';

E(1) = d(1) - A_in*X(1,:).';
A(1,:) = A_in + alpha*E(1)*conj(X(1,:));

if M>1,
    for k=2:M,
        E(k) = d(k) - A(k-1,:)*X(k,:).';
        A(k,:) = A(k-1,:) + alpha*E(k)*conj(X(k,:));
    end;
end;

function H = convolm(x,num_zeros,pad)
%convolm                        [jMc 2/89]
% creates a convolution matrix, optionally padded with zeros
%    convolm(X,P) creates a convolution matrix from
%        the data vector x(1:N). The matrix will have p columns.
%    convolm(X,P,'<') will pad with 'p-1' zeros at the beginning of x
```

```
%    convolm(X,P,'>') will pad with 'p-1' zeros at the end of x
%    convolm(X,P,'<>') will pad both end of the signal x with p-1 zeros
%
% RECALL : H = [ h(i,j) ] where h(i,j) = x(p+i-j)
% see HANKEL, CIRCULANT, CONVMTX

N=length(x);
x=x(:);

if nargin==3
    z = zeros(num_zeros-1,1);
    if pad == '<>'
           N= N + 2*(num_zeros-1); t=[z; x; z];
       elseif pad == '<'
           N= N + num_zeros-1;   t=[z; x];
       elseif pad == '>'
           N= N + num_zeros-1;   t=[x; z];
       end;
else
   t=x;
end;

H= zeros(N-num_zeros+1,num_zeros);
for i=1:num_zeros
    H(:,i) = t(num_zeros-i+1:N-i+1);
end;
```