ELECTRONOTES
1016 Hanshaw Road
Ithaca, NY 14850

APPLICATION NOTE NO. 406

March 15, 2014

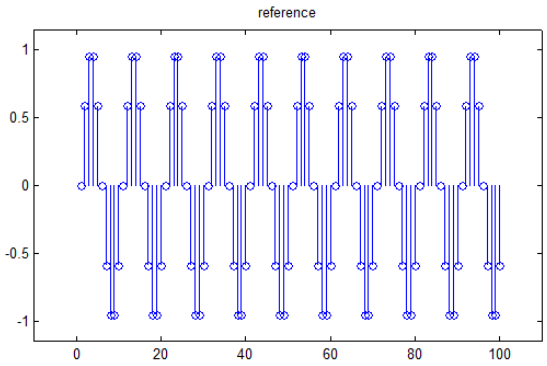
WIENER ALTERNATIVE TO LMS ALGORITHM

In this app note the purpose is first (and not too essentially) to update some adaptive filter code presented previously [2,3]. More importantly, we also want to give the code for an alternative calculation method (Wiener solution) [1], which although said code is 20 years old and used in teaching, I believe it was not published here. The previous presentations [1-3] gave a wealth of information and examples of adaptive filtering. Here the main message is that the Wiener solution gives the correct solution as a one-time calculation, the same solution as the iterative LMS algorithm. Nothing too much that is new here.

Here is the ***adapt.m*** program that uses the LMS algorithm:

```
function [y,e,w]=adapt(d,x,L,mu,W0)
%--------------------------------------------------------------------%
%  function [y,e,w]=adapt(d,x,L,mu,W0)                               %
%                                                                    %
%      LMS ALGORITHM ADAPTIVE FILTER SIMULATOR (adapt.m)             %
%                                                                    %
%  INPUTS:                                                           %
%  d is desired signal, x is reference signal, both the same length  %
%  L is the length of the adaptive linear combiner                   %
%  mu is the convergence factor (try about 0.01 to start)            %
%  W0, when included, is an initial weight vector of length L        %
%                                                                    %
%  OUTPUTS:                                                          %
%  y is the output of adaptive linear combiner (correlated signal)   %
%  e is the error (uncorrelated signal)                              %
%  w is the final weight vector                                      %
%--------------------------------------------------------------------%
%                       1          2          3              L-1     %
%      x             -----      -----      -----            -----    %
%      ___________  |  -1 |___|  -1 |___|  -1 |___    ___|  -1 |__  %
%                \    | z   | | | z   | | | z   | |       | z   |  |  %
%                 \    -----  |  -----  |  -----  |        -----   |  %
%                  \ W0    W1 \        / W2       / W3              /  %
%                   \           -----------      /                /   %
%                    \-------|    SUM    |----/           <-----/    %
%                            -----------                             %
%                                 |                                  %
%                                 |--------------------------- y     %
%                               (-)|                                 %
%                             (+)  -----                             %
%  d --------------------------| SUM |------------------------ e     %
%                               -----                                %
%                                                                    %
%     LMS Algorithm    Wj(n+1) = Wj(n) + 2*mu*e(n)*xj(n)             %
%                                                                    %
%                                          B. Hutchins   Fall 1993   %
%                                               revised March 2014   %
%--------------------------------------------------------------------%
```

```
if exist('W0')==1 w=W0;   % check for user-given initial weights
else w=zeros(1,L);        % else set taps to zero
end

xj=zeros(1,L);            % initialize linear combiner delay line

for n = 1:length(d)           %  for each iteration
     xj=[x(n),xj(1:L-1)];     %  move data up line
     y(n)=xj*w';              %  compute y
     e(n)=d(n)-y(n);          %  compute e
     w=w+2*mu*e(n).*xj;       %  update taps w(n+1)=w(n)+2mux(n)e(n)
end                           %


% Done Calculations - Rest Just Display
hm=-0.1*length(x);
hp=1.1*length(x);
vm=1.2*min(x);
vp=1.2*max(x);

figure(1);
clf                           %  plot the four signals
subplot(221)                  %
stem(x)                       %  x
axis([hm hp vm vp])
title('reference')            %
subplot(222)
stem(y)                       %  y
axis([hm hp vm vp])
title('y - correlated')       %
subplot(223)
stem(d)                       %  d
axis([hm hp vm vp])
title('desired')              %
subplot(224)
stem(e)                       %  e
axis([hm hp vm vp])
title('error - uncorrelated') %
figure(1)

clf
figure(2)                     %    now plot the impulse and freq. resp.
subplot(221)                  %    of the final taps.  This is strictly
stem([0:L-1],w)               %    valid information only if the error is zero.
axis([-0.1*(L-1) 1.1*(L-1) 1.2*min(w) 1.2*max(w)]) % . . . . .
title('tap weights at end')   %    If the error is very small,
subplot(223)                  %    and/or if mu is very small,
H=freqz(w,1,500);             %    the result may be valid if thought of
plot([0:.001:.499],abs(H),'r')%    as an instantaneous frequency response
axis([-.05 0.55 -.1 1.1*max(abs(H)) ])
grid
title('magnitude at end')
subplot(224)
plot([0:.001:.499],angle(H),'r')
axis([-.05 0.55 1.1*min(angle(H)) 1.1*max(angle(H)) ])
grid
title('phase at end')
figure(2)
```



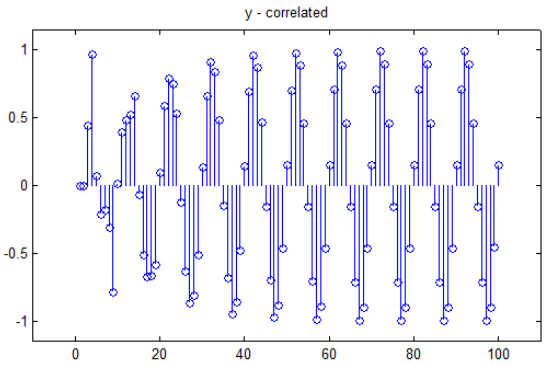
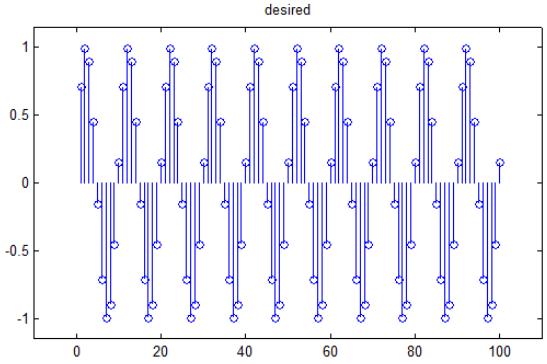
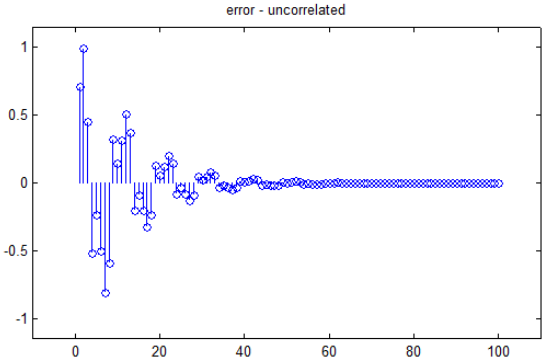
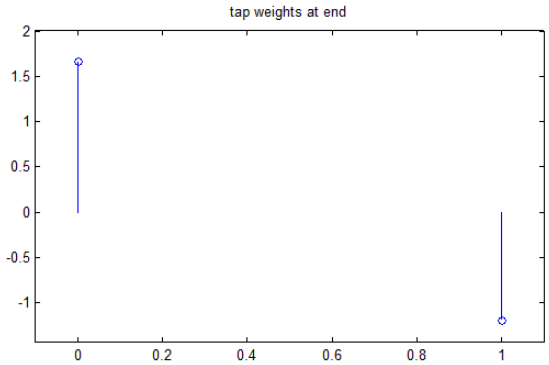


Fig. 1







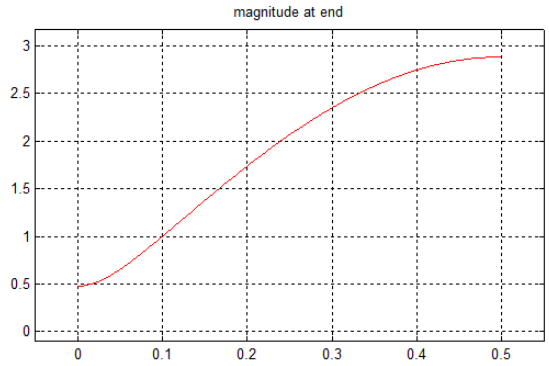
x = sin(2*pi*[0:99]/10)

d = sin(2*pi*[0:99]/10 + pi/4)

[y,e,w]=adapt(d,x,2,.4)

w = 1.6802 -1.2029

Fig. 2



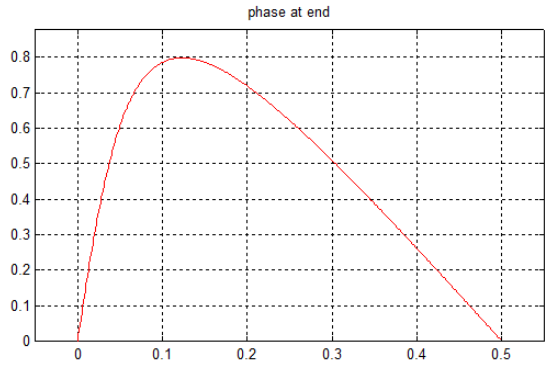


Fig. 1 and Fig. 2 give the results of our demonstration, and as we have said, the references [1-3] have similar and many more examples. We start with an input signal x and a desired signal d (see diagram at top of ***adapt.m*** program, and the upper right of Fig. 2) that are samples of a sinusoidal waveform with frequency 0.1, differing in phase by 45°. We have further chosen a length L = 2 for the adaptive linear combiner – ALC) and a convergence factor of mu=0.4. Note that the iterative LMS algorithm (see y growing, error e decreasing) converges and shows that d and x (as modified by the ALC) can cancel. Note the two tap weights. So we see from Fig. 1 that the algorithm does work, and from Fig. 2 we see why. The ALC becomes stationary enough that it can be considered a pseudo-fixed FIR filter, and we compute the magnitude and phase response of the two weights w. These are shown in Fig. 2, and at the frequency of interest, 0.1, the magnitude response is 0.9999 and the phase is 45.0011°, which is nearly exactly what we needed. [If we had chosen a longer ALC, we would have still gotten a similar cancellation, but there would have been three tap weights giving a different solution except at the critical frequency of 0.1. Here we rigged things so that we get the one answer the Wiener calculation gives. Again see the first two references.]

Now here is the ***wiener.m*** program.

```
function w=wiener(d,x,L)
% function w=wiener(d,x,L)
% WIENER FILTER DEMO            wiener.m
%   d = desired input (typically one cycle of periodic sequence)
%   x = reference (typically one cycle, same length as d)
%   L = number of adaptive filter taps
% B. Hutchins                                              Fall 1993
%                                                 revised March 2014

Lx=length(x);           %  basic length
R=zeros(L,L);           %  zero R
P=zeros(1,L);           %         and P


for row=1:L
    for col=1:L
         for v=0:(Lx-1)/2
             R(row,col)=R(row,col)+x(row+v)*x(col+v);
         end
    end
    for v=1:Lx/2
           P(row)=P(row)+x(v)*d(row+v-1);
    end
end
R                               %   Input Correlation
P                               %   Cross Correlation
w=inv(R)*P'                     %   Wiener Weight Vector


wr=flipud(w);
for v=1:Lx-1                        %   compute
     y(v+L-1)=x(v:v+L-1)*wr;        %           the
     e(v+L-1)=d(v+L-1)-y(v+L-1);    %              error
end                                 %
```

```
% Computations Done - Rest Just Display
hm=-0.1*length(x);
hp=1.1*length(x);
vm=1.2*min(x);
vp=1.2*max(x);

figure(1);
clf                          %  plot the four signals
subplot(221)                 %
stem(x)                      %  x
axis([hm hp vm vp])
title('reference')           %
subplot(222)
stem(y)                      %  y
axis([hm hp vm vp])
title('y - correlated')      %
subplot(223)
stem(d)                      %  d
axis([hm hp vm vp])
title('desired')             %
subplot(224)
stem(e)                      %  e
axis([hm hp vm vp])
title('error - uncorrelated') %
figure(1)

clf
figure(2)                    %  now plot the impulse and freq. resp.
subplot(221)                 %  of the taps.
stem([0:L-1],w)
axis([-0.1*(L-1) 1.1*(L-1) 1.2*min(w) 1.2*max(w)])
title('tap weights at end')
subplot(223)
H=freqz(w,1,500);
plot([0:.001:.499],abs(H),'r')
axis([-.05 0.55 -.1 1.1*max(abs(H)) ])
grid
title('magnitude at end')
subplot(224)
plot([0:.001:.499],angle(H),'r')
axis([-.05 0.55 1.1*min(angle(H)) 1.1*max(angle(H)) ])
grid
title('phase at end')
figure(2)
```

In this case we used the same x and d as above, and the command line was then:

w=wiener(d,x,2)

with results:

R = 25.0000 20.2254
20.2254 25.0000

P = 17.6777 3.9109

w = 1.6804 -1.2030

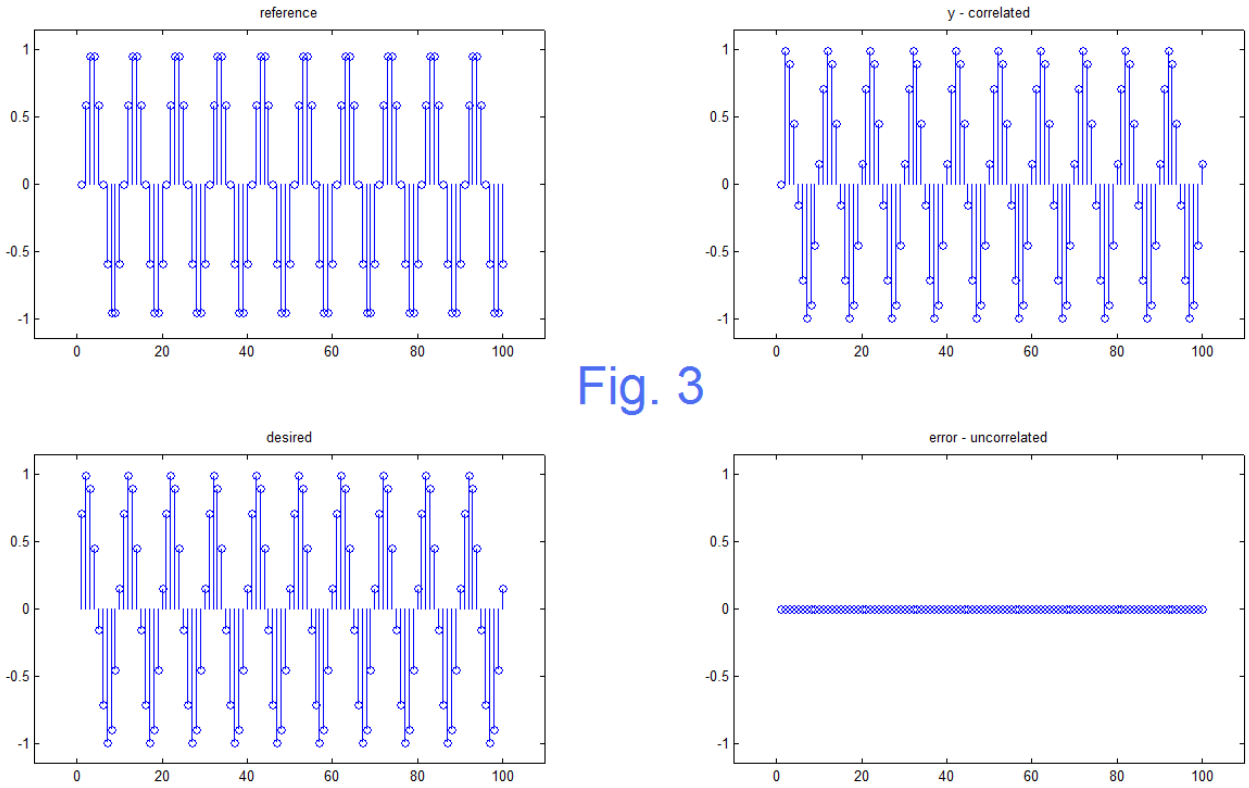


Fig. 3

Here there is no algorithm to converge. Rather the autocorrelation (R) and cross-correlations (P) are computed from available data. Here, we used all the data possible, which is about half the signal, so that the correlations overlap. The Wiener weight vector is the classic:

$$w = R^{-1}P$$

and this is virtually identical to the LMS algorithm. If fact, the result is so nearly exact that the plot of the results corresponding to ***wiener.m*** perfectly overlaps Fig. 2 for the LMS and need not be separately presented here.

REFERENCES:

[1] B. Hutchins, “An Introduction to Adaptive Filters of Several Types,” ***Electronotes***, Vol. 16, No. 170, Feb. 1988, pp 3-46

[2] B. Hutchins, “A New Adaptive Filter Simulator,” ***Electronotes***, Vol. 17, No. 178, July 1991, pp 35-46

[3] B. Hutchins, “A New Adaptive Filter Simulator Program,” Electronotes Application Note AN-327, Nov. 1993.