

Adaptive Filtering

by: Thomas Drumright

Spring 1998

Introduction

Digital signal processing (DSP) has been a major player in the current technical advancements such as noise filtering, system identification, and voice prediction. Standard DSP techniques, however, are not enough to solve these problems quickly and obtain acceptable results.

Adaptive filtering techniques must be implemented to promote accurate solutions and a timely convergence to that solution.

Adaptive Filtering System Configurations

There are four major types of adaptive filtering configurations; adaptive system identification, adaptive noise cancellation, adaptive linear prediction, and adaptive inverse system. All of the above systems are similar in the implementation of the algorithm, but different in system configuration. All 4 systems have the same general parts; an input $x(n)$, a desired result $d(n)$, an output $y(n)$, an adaptive transfer function $w(n)$, and an error signal $e(n)$ which is the difference between the desired output $u(n)$ and the actual output $y(n)$. In addition to these parts, the system identification and the inverse system configurations have an unknown linear system $u(n)$ that can receive an input and give a linear output to the given input.

Adaptive System Identification Configuration

The adaptive system identification is primarily responsible for determining a discrete estimation of the transfer function for an unknown digital or analog system. The same input $x(n)$ is applied to both the adaptive filter and the unknown system from which the outputs are compared (see figure 1). The output of the adaptive filter $y(n)$ is subtracted from the output of the unknown

system resulting in a desired signal $d(n)$. The resulting difference is an error signal $e(n)$ used to manipulate the filter coefficients of the adaptive system trending towards an error signal of zero.

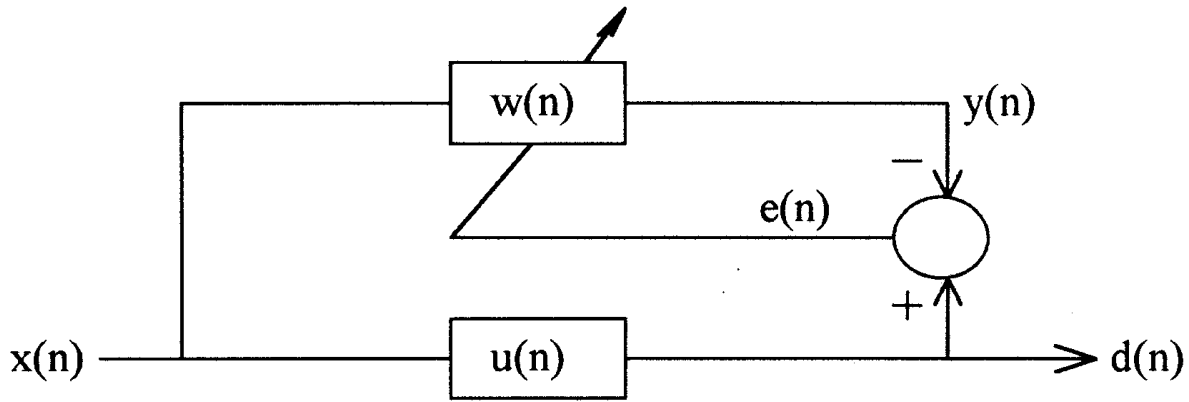


Figure 1. Adaptive System Identification Configuration

After a number of iterations of this process are performed, and if the system is designed correctly, the adaptive filter's transfer function will converge to, or near to, the unknown system's transfer function. For this configuration, the error signal does not have to go to zero, although convergence to zero is the ideal situation, to closely approximate the given system.

There will, however, be a difference between adaptive filter transfer function and the unknown system transfer function if the error is nonzero and the magnitude of that difference will be directly related to the magnitude of the error signal.

Additionally the order of the adaptive system will affect the smallest error that the system can obtain. If there are insufficient coefficients in the adaptive system to model the unknown system, it is said to be under specified. This condition may cause the error to converge to a nonzero constant instead of zero. In contrast, if the adaptive filter is over specified, meaning that

there are more coefficients than needed to model the unknown system, the error will converge to zero, but it will increase the time it takes for the filter to converge.

Adaptive Noise Cancellation Configuration

The second configuration is the adaptive noise cancellation configuration as shown in figure 2. In this configuration the input $x(n)$, a noise source $N_1(n)$, is compared with a desired signal $d(n)$, which consists of a signal $s(n)$ corrupted by another noise $N_0(n)$. The adaptive filter coefficients adapt to cause the error signal to be a noiseless version of the signal $s(n)$.

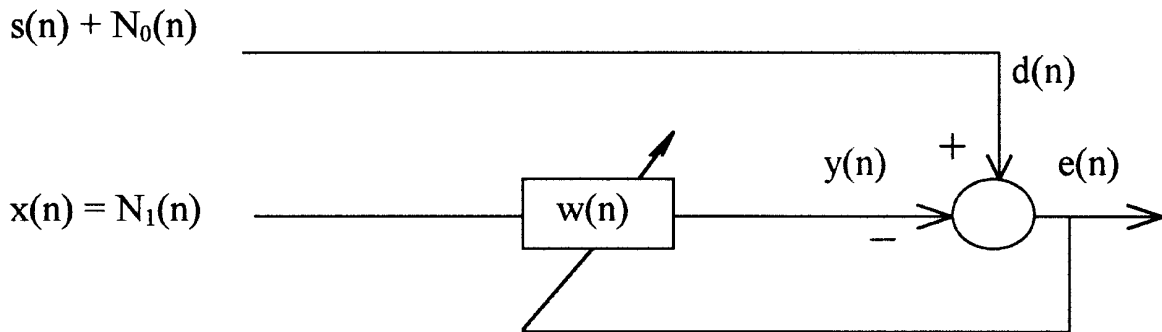


Figure 2. Adaptive Noise Cancellation Configuration

Both of the noise signals for this configuration need to be uncorrelated to the signal $s(n)$. In addition, the noise sources must be correlated to each other in some way, preferably equal, to get the best results.

Due to the nature of the error signal, the error signal will never become zero. The error signal should converge to the signal $s(n)$, but not converge to the exact signal. In other words, the difference between the signal $s(n)$ and the error signal $e(n)$ will always be greater than zero. The only option is to minimize the difference between those two signals.

Adaptive Linear Prediction Configuration

Adaptive linear prediction is the third type of adaptive configuration (see figure 3). This configuration essentially performs two operations. The first operation, if the output is taken from the error signal $e(n)$, is linear prediction. The adaptive filter coefficients are being trained to predict, from the statistics of the input signal $x(n)$, what the next input signal will be. The second operation, if the output is taken from $y(n)$, is a noise filter similar to the adaptive noise cancellation outlined in the previous section.

As in the previous section, neither the linear prediction output nor the noise cancellation output will converge to an error of zero. This is true for the linear prediction output because if the error signal did converge to zero, this would mean that the input signal $x(n)$ is entirely deterministic, in which case we would not need to transmit any information at all.

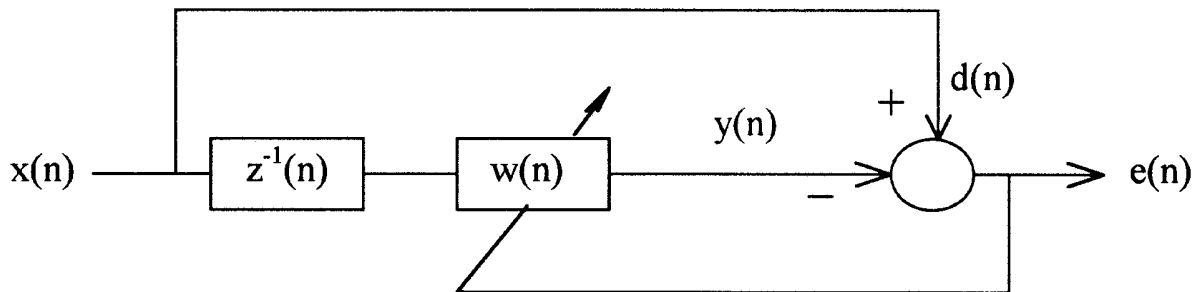


Figure 3. Linear Prediction Configuration.

In the case of the noise filtering output, as outlined in the previous section, $y(n)$ will converge to the noiseless version of the input signal.

Adaptive Inverse System Configuration

The final filter configuration is the adaptive inverse system configuration as shown in figure 4. The goal of the adaptive filter here is to model the inverse of the unknown system $u(n)$. This is particularly useful in adaptive equalization where the goal of the filter is to eliminate any spectral changes that are caused by a prior system or transmission line. The way this filter works is as follows. The input $x(n)$ is sent through the unknown filter $u(n)$ and then through the adaptive filter resulting in an output $y(n)$. The input is also sent through a delay to attain $d(n)$. As the error signal is converging to zero, the adaptive filter coefficients $w(n)$ are converging to the inverse of the unknown system $u(n)$.

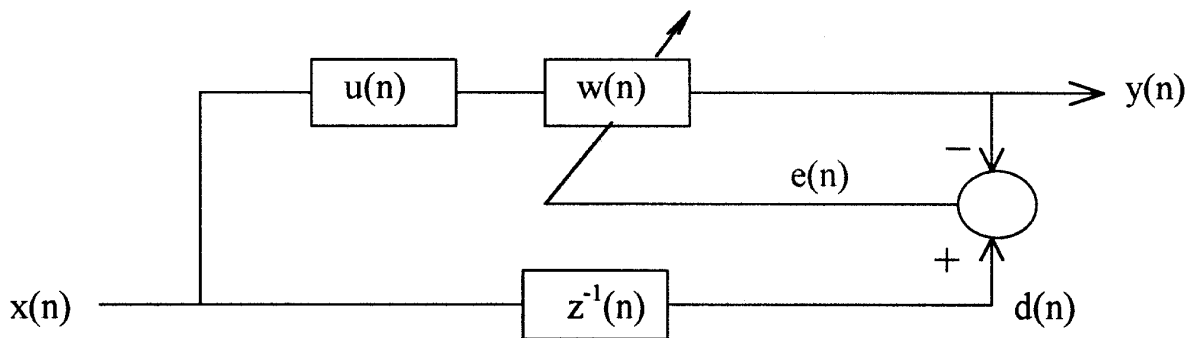


Figure 4. Adaptive Inverse Configuration.

For this configuration, as for the system identification configuration, the error can theoretically go to zero. This will only be true, however, if the unknown system consists only of a finite number of poles or the adaptive filter is an IIR filter. If neither of these conditions are true, the system will converge only to a constant due to the limited number of zeroes available in an FIR system.

Performance Measures in Adaptive Systems

Six performance measures will be discussed in the following sections; convergence rate, minimum mean square error, computational complexity, stability, robustness, and filter length.

Convergence Rate

The convergence rate determines the rate at which the filter converges to its resultant state.

Usually a faster convergence rate is a desired characteristic of an adaptive system. Convergence rate is not, however, independent of all of the other performance characteristics. There will be a tradeoff, in other performance criteria, for an improved convergence rate and there will be a decreased convergence performance for an increase in other performance. For example, if the convergence rate is increased, the stability characteristics will decrease, making the system more likely to diverge instead of converge to the proper solution. Likewise, a decrease in convergence rate can cause the system to become more stable. This shows that the convergence rate can only be considered in relation to the other performance metrics, not by itself with no regards to the rest of the system.

Minimum Mean Square Error

The minimum mean square error (MSE) is a metric indicating how well a system can adapt to a given solution. A small minimum MSE is an indication that the adaptive system has accurately modeled, predicted, adapted and/or converged to a solution for the system. A very large MSE usually indicates that the adaptive filter cannot accurately model the given system or the initial state of the adaptive filter is an inadequate starting point to cause the adaptive filter to converge.

There are a number of factors which will help to determine the minimum MSE including, but not

limited to; quantization noise, order of the adaptive system, measurement noise, and error of the gradient due to the finite step size.

Computational Complexity

Computational complexity is particularly important in real time adaptive filter applications.

When a real time system is being implemented, there are hardware limitations that may affect the performance of the system. A highly complex algorithm will require much greater hardware resources than a simplistic algorithm.

Stability

Stability is probably the most important performance measure for the adaptive system. By the nature of the adaptive system, there are very few completely asymptotically stable systems that can be realized. In most cases the systems that are implemented are marginally stable, with the stability determined by the initial conditions, transfer function of the system and the step size of the input.

Robustness

The robustness of a system is directly related to the stability of a system. Robustness is a measure of how well the system can resist both input and quantization noise.

Filter Length

The filter length of the adaptive system is inherently tied to many of the other performance measures. The length of the filter specifies how accurately a given system can be modeled by the adaptive filter. In addition, the filter length affects the convergence rate, by increasing or decreasing computation time, it can affect the stability of the system, at certain step sizes, and it

affects the minimum MSE. If the filter length of the system is increased, the number of computations will increase, decreasing the maximum convergence rate. Conversely, if the filter length is decreased, the number of computations will decrease, increasing the maximum convergence rate. For stability, due to an increase in length of the filter for a given system, you may add additional poles or zeroes that may be smaller than those that already exist. In this case the maximum step size, or maximum convergence rate, will have to be decreased to maintain stability. Finally, if the system is under specified, meaning there are not enough poles and/or zeroes to model the system, the mean square error will converge to a nonzero constant. If the system is over specified, meaning it has too many poles and/or zeroes for the system model, it will have the potential to converge to zero, but increased calculations will affect the maximum convergence rate possible.

Filter Algorithms

A number of filter algorithms will be discussed in this section; the finite impulse response (FIR) least mean squares (LMS) gradient approximation method will be discussed in detail, characteristics of infinite impulse response (IIR) adaptive filters will be briefly discussed, the transform domain adaptive filter (TDAF) and numerous other algorithms will be mentioned for completeness.

Finite Impulse Response (FIR) Algorithms

Least Mean Squares Gradient Approximation Method

Given an adaptive filter with an input $\underline{x}(n)$, an impulse response $w(n)$ and an output $y(n)$ you will get a mathematical relation for the transfer function of the system

$$y(n) = \underline{\mathbf{w}}^T(n)\underline{\mathbf{x}}(n)$$

and

$$\underline{\mathbf{x}}(n) = [x(n), x(n-1), x(n-2), \dots, x(n-(N-1))]$$

where $\underline{\mathbf{w}}^T(n) = [w_0(n), w_1(n), w_2(n) \dots w_{N-1}(n)]$ are the time domain coefficients for an N^{th} order FIR filter.

Note in the above equation and throughout a boldface letter represents a vector and the superscript T represents the transpose of a real valued vector or matrix.

Using an estimate of the ideal cost function the following equation can be derived.

$$\underline{\mathbf{w}}(n+1) = \underline{\mathbf{w}}(n) - \mu \Delta_{E[e^2]}(n).$$

In the above equation $\underline{\mathbf{w}}(n+1)$ represents the new coefficient values for the next time interval, μ is a scaling factor, and $\Delta_{E[e^2]}(n)$ is the ideal cost function with respect to the vector $\underline{\mathbf{w}}(n)$. From the above formula one can derive the estimate for the ideal cost function

$$\underline{\mathbf{w}}(n+1) = \underline{\mathbf{w}}(n) - \mu e(n)\underline{\mathbf{x}}(n)$$

where

$$e(n) = d(n) - y(n)$$

and

$$y(n) = \mathbf{x}^T(n)\mathbf{w}(n).$$

In the above equation μ is sometimes multiplied by 2, but here we will assume it is absorbed by the μ factor.

In summary, in the Least Mean Squares Gradient Approximation Method, often referred to as the Method of Steepest Descent, a guess based on the current filter coefficients is made, and the gradient vector, the derivative of the MSE with respect to the filter coefficients, is calculated from the guess. Then a second guess is made at the tap-weight vector by making a change in the present guess in a direction opposite to the gradient vector. This process is repeated until the derivative of the MSE is zero.

Convergence of the LMS Adaptive Filter

The convergence characteristics of the LMS adaptive filter is related to the autocorrelation of the input process as defined by

$$\mathbf{R}_x = E[\mathbf{x}(n)\mathbf{x}^T(n)]$$

There are a two conditions that must be satisfied in order for the system to converge. These conditions include:

- The autocorrelation matrix, \mathbf{R}_x , must be positive definite.
- $0 < \mu < 1/\lambda_{\max}$, where λ_{\max} is the largest eigenvalue of \mathbf{R}_x .

In addition, the rate of convergence is related to the eigenvalue spread. This is defined using the condition number of \mathbf{R}_x , defined as $\kappa = \lambda_{\max}/\lambda_{\min}$, where λ_{\min} is the minimum eigenvalue of \mathbf{R}_x . The fastest convergence of this system occurs when $\kappa = 1$, corresponding to white noise. This states that the fastest way to train a LMS adaptive system is to use white noise as the training input. As the noise becomes more and more colored, the speed of the training will decrease.

Transform Domain Adaptive Filter (TDAF)

The TDAF refers to the application of an adaptive filter after a transform of the input system to an orthogonal basis, such as the discrete Fourier Transform (DFT), the discrete cosine transform (DCT), and the Walsh Hadamard transform (WHT). The primary motivation for the use of the TDAF is that for colored noise, the TDAF can exhibit faster convergence rates. This will be briefly explained in the next section.

The TDAF, in simple terms, is a transform of the system, using one of the above techniques, followed by an application of an adaptive filter algorithm, such as the LMS adaptive filter.

Convergence of the TDAF

It can be shown that the optimum MSE surface is a hypersphere. When white noise is used in training, the error surface will approach a hypersphere. If, however, colored noise is being used as the training input, the shape of the MSE surface will be a hyperellipse. This causes an increase in the convergence time.

When the TDAF is applied to the system with a colored noise input, it causes a rotation and a scaling of the axis, causing the points of the MSE surface to intersect the axes at equal distances. This will cause an increase in the convergence time of the adaptive filter.

Quasi-Newton Adaptive Algorithms

The quasi-Newton adaptive algorithm uses second order statistics to reduce the convergence rate of an adaptive filter, via the Gauss-Newton method. Probably the best known quasi-Newton algorithm is the recursive least squares (RLS) algorithm. It is important to note that even with the increase in convergence rate, the RLS algorithm requires great amounts of processing power, which can make it difficult to implement on real-time systems.

There are a number of other quasi-Newton algorithms that have fast convergence rates, and that are also feasible alternatives for real time processing. See [1] and [3] for more information.

Adaptive Lattice Algorithms

The primary reason for the use of a lattice structure is to reduce the quantization noise introduced by filter coefficients in finite word length systems. As the name suggests, adaptive lattice algorithms are design with the goal of reducing coefficient quantization error, and thus, possibly decreasing the word length of the system with comparable performance. See [1] and [3] for more information.

Infinite Impulse Response (IIR) Adaptive Filters

The primary advantage of IIR filters is that to produce an equivalent frequency response to an FIR filter, they can have a fewer number of coefficients. This in theory should reduce the number of adds, multiplies and shifts to perform a filtering operation.

This theory of using IIR filters to reduce the computational burden is the primary motivation for the use of IIR adaptive filters. There are, however, a number of problems that are introduced with the use of IIR adaptive filters.

- The fundamental concern with IIR adaptive filters is the potential for instability due to poles moving outside the unit circle during the training process. Even if the system is initially stable and the final system is stable, there is still the possibility of the system going unstable during the convergence process. Some suggestion has been made to limit the poles to within the unit circle, however, this method requires that the step sizes be small, which considerably reduces the convergence rate.
- Due to the interplay between the movement of the poles and zeros, the convergence of IIR systems tends to be slow [3]. The result is that even though IIR filters have fewer coefficients, therefore few calculations per iteration, the number of iterations may increase cause a net loss in processing time to convergence. This, however, is not a problem with all pole filters.
- In an IIR system, the MSE surface may contain local minimum that can cause a convergence of that system to the local minimum instead of the absolute minimum. More care need to be taken in the initial conditions in IIR adaptive filters than in FIR adaptive filters.
- IIR filters are more susceptible to coefficient quantization error than FIR, due to the feedback.

There have been a number of studies done on the use of IIR adaptive filters, but due to the problems stated above, they are still not widely used in industry today.

Software Simulations

A simulation of each of the four FIR systems was performed using Matlab. The results are outlined in the 4 sections below.

Adaptive System Identification

A fourth order system was created using the unknown system Matlab program, see figure 1.

Using the adaptive system identification program a 7th order system was initialized to zero and the algorithm was performed. In under 200 iterations, see figure 2, the coefficients were determined with an overall error of 4.7269×10^{-008} . The coefficients of the system were estimated as shown below.

Estimated coefficients = [1.0000, -1.4160, 5.0000, -1.4160, 1.0000, 0.0000, 0.0000, 0.0000]

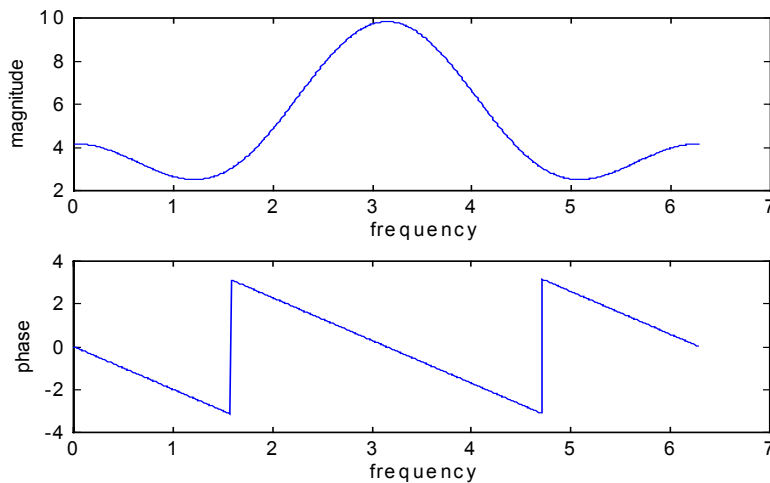


Figure 1. Magnitude and Phase Response of the Unknown System.

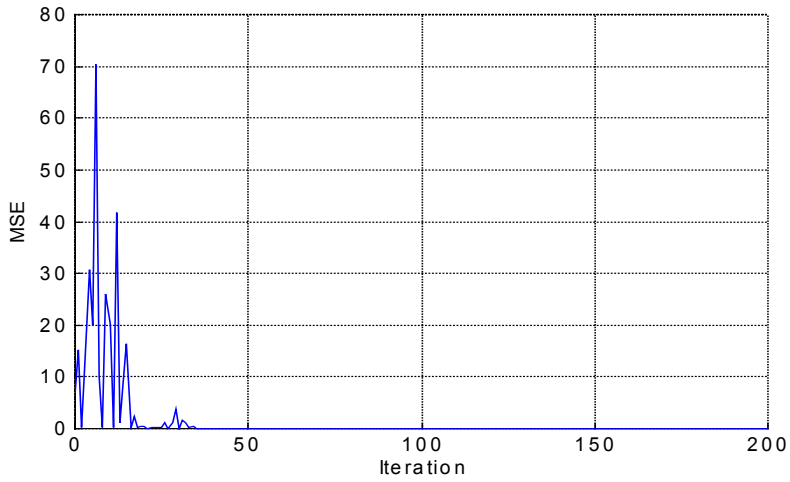


Figure 2. MSE verses Number of Iterations.

Adaptive Noise Cancellation

The adaptive noise cancellation system was given an input of a noisy, frequency varying sinewave, shown in figure 3, with a frequency response shown in figure 4. After 1000 iterations of the filter, figure 7, the noise is considerably reduced, see figures 5 and 6.

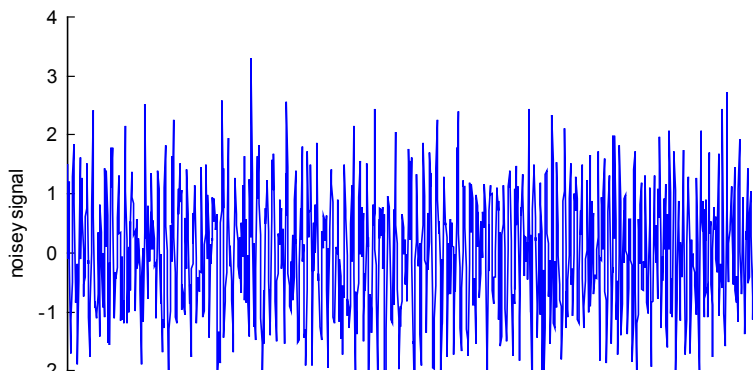


Figure 3. Time Response of a Noisy Signal

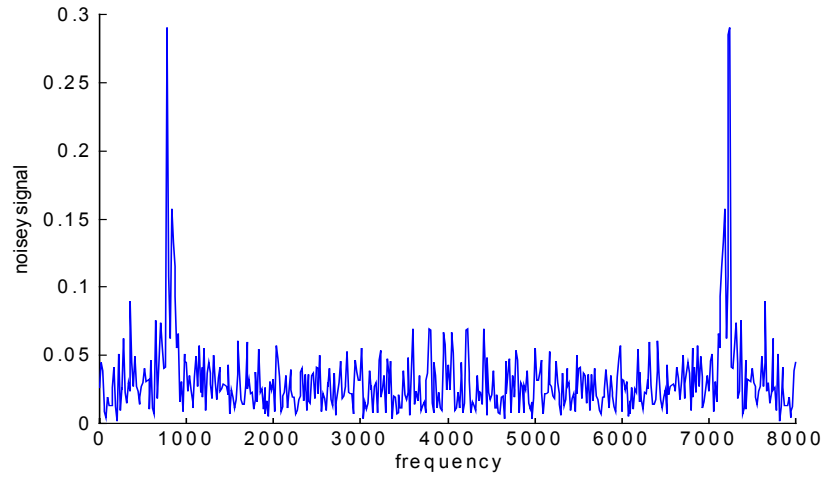


Figure 4. Frequency Response of a Noisy Signal.

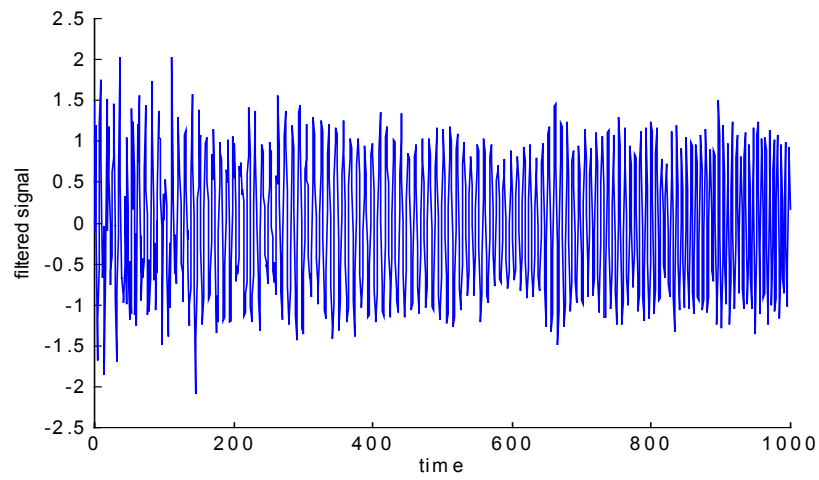


Figure 5. Time Response of the Filtered Signal.

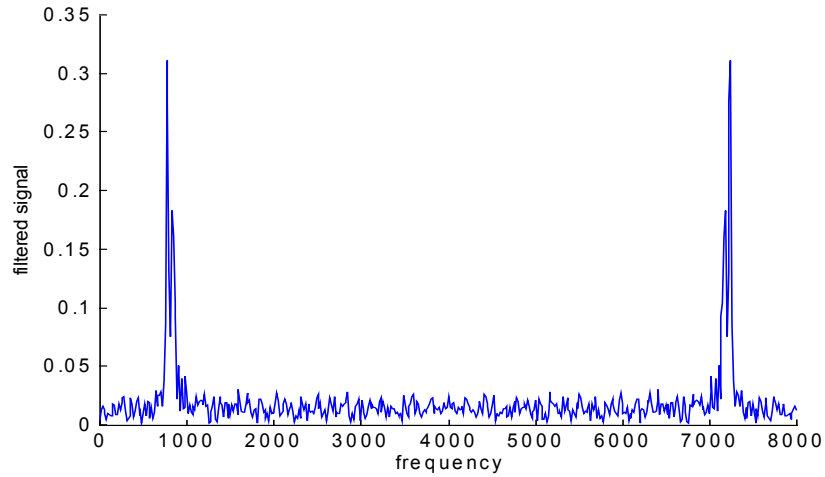


Figure 6. Frequency Response of the Filtered Signal.

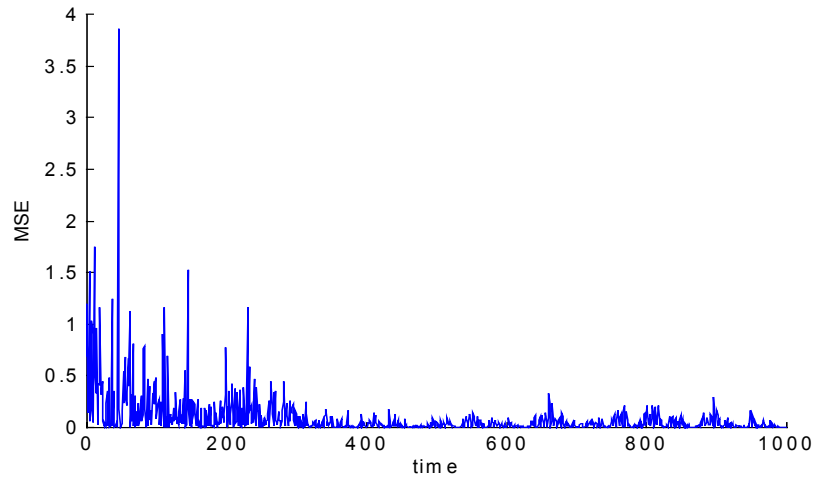


Figure 7. MSE versus the Number of Iterations.

Adaptive Linear Predictive Coding

Figure 8 shows the input to the adaptive linear predictive coding algorithm. Figure 9 shows the output, $y(t)$, of the adaptive LPC system representing the noise cancellation output, where it can be seen that the output is converging to a noiseless system. Figure 10 shows the error signal of the LPC system.

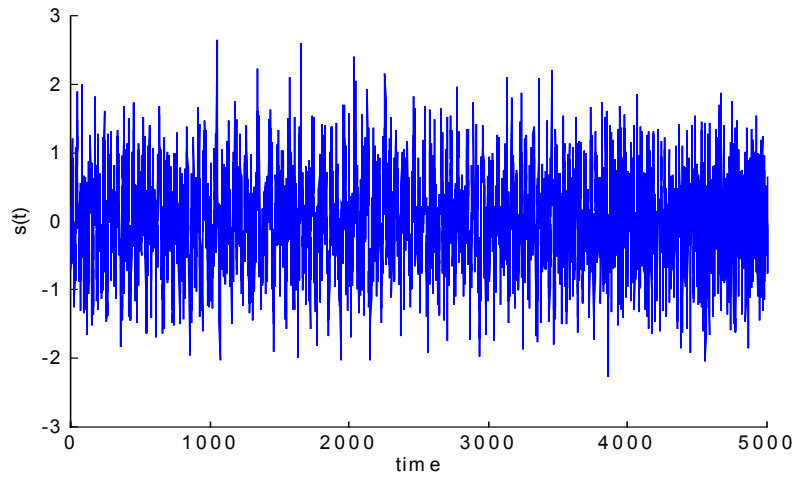


Figure 8. Time Response of a Noisy Signal.

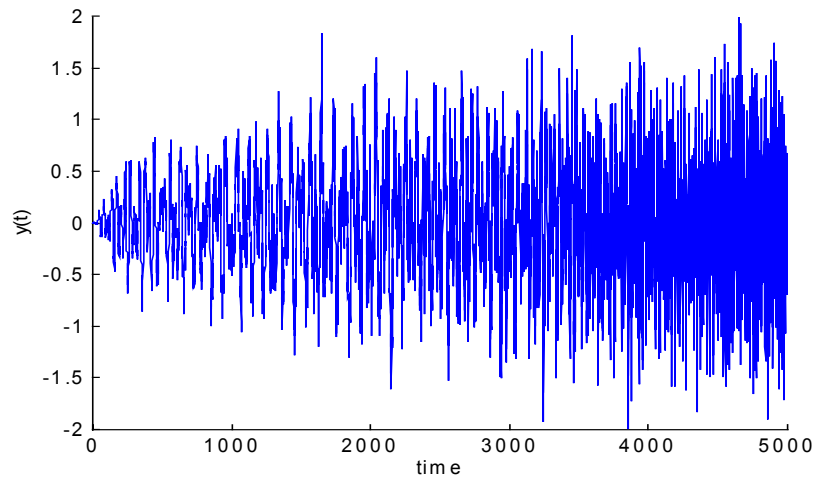


Figure 9. Time Response of the Filtered Signal.

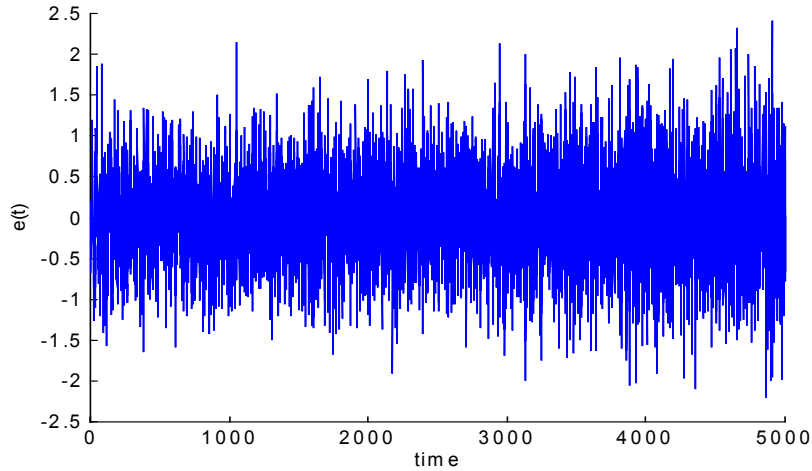


Figure 10. Time Response of the Error Signal.

The frequency responses of each of the above inputs and outputs are given in figures 11-13.

Figure 12 shows that the LPC is filtering the signal. Figure 13 shows that the error signal from an LPC system has a much smaller dynamic range in the frequency domain, than does the original signal.



Figure 11. Frequency Response of a Noisy Signal.

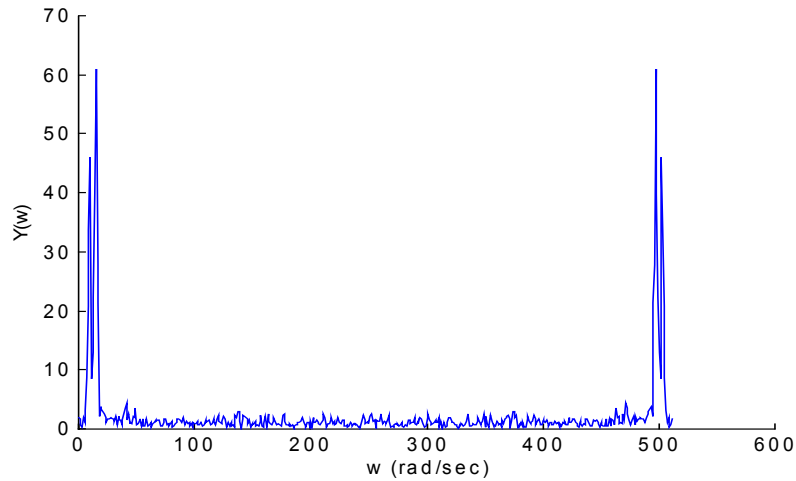


Figure 12. Frequency Response of the Filtered Signal.

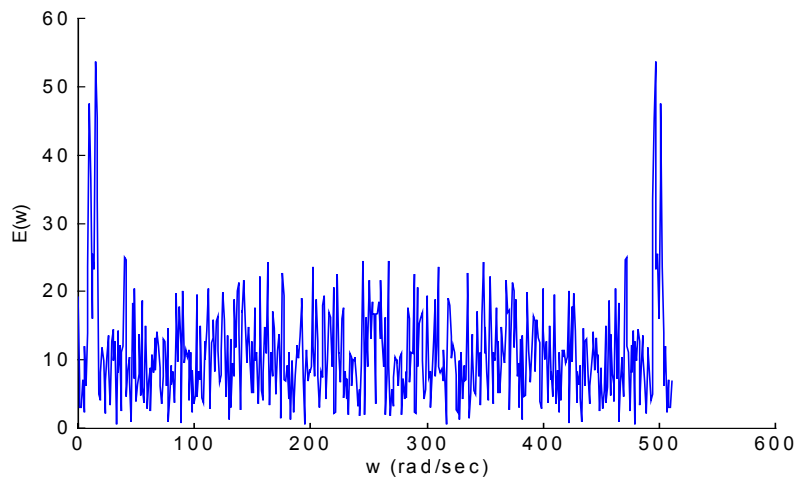


Figure 13. Frequency Response of the Error Signal.

Adaptive Inverse System

Figure 14 shows the frequency response of an unknown system. The adaptive system algorithm was performed on this system and the result is shown in figure 15. Figure 16 shows the combination of the 2 responses in an attempt to equalize the frequency response of the original unknown system.

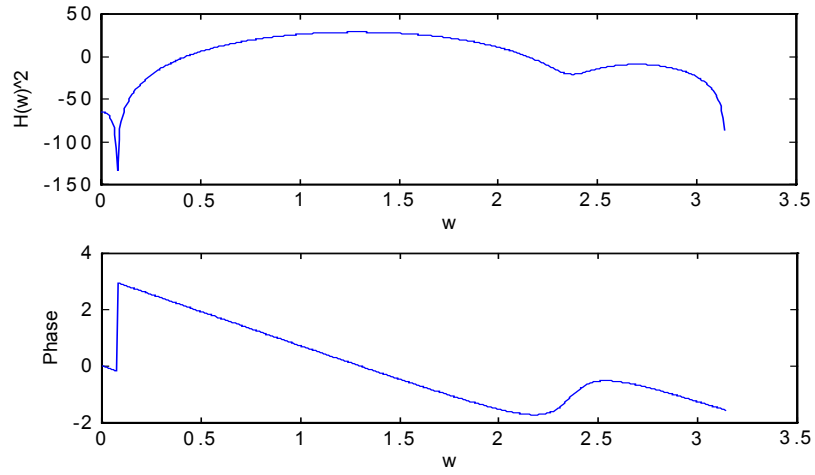


Figure 14. Frequency Response of the Unknown System.

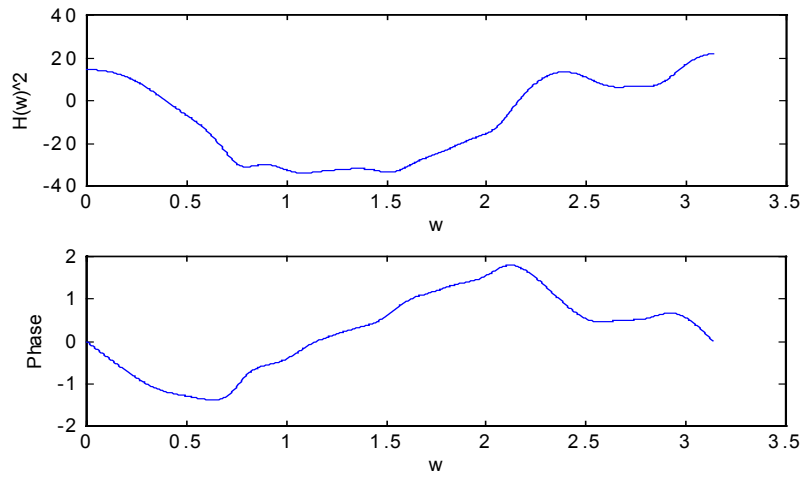


Figure 15. Frequency Response of the Adaptive System.

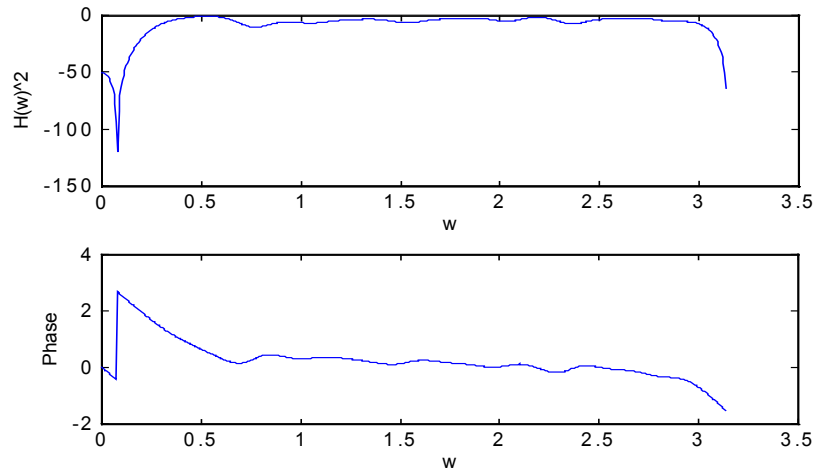


Figure 16. Frequency Response of the Corrected System.

Matlab Code

Below is the Matlab Code that was used in the simulations above.

```

% Adaptive System Identification
% This M file recursively calculated the coefficients of
% an unknown system.
% mu - specifies the rate of convergence of the adaptive
% algorithm. mu must be less than the smallest eigenvalue
% of the unknown system for the adaptive filter to be
% properly conditioned.
% len - the order of the adaptive filter.
% len must be greater than or equal to the order of the
% unknown filter to model the unknown filter properly.
% lin - the number of iterations to take place.
% xin - Gaussian white noise input signal to the system.
%
% This M file uses unsys.m
%
% By Thomas Drumright
% 12/12/97

clear
% Initializations
INPUTVEC = 0;           % Initialize a global variable
mu = .1;                % Convergence factor

```

```

len = 8;                % Filter length
lin = 200;             % Number of iterations
xin = randn(lin,1);   % Input white noise
mse = 0;
x = 0*(1:len)';       % Initial input
w = 0*(1:len);        % Initial filter coeffs.

% Calculate first iteration
d = unsys(x(1));      % Send x(1) to the unknown sys.
y = w*x;              % Calculate output of adapt. filt.
e = d-y;              % Calculate error signal
mse(1) = e^2;         % Calculate mean square error
w = w + mu*e*x';     % Calculate next set of coefficients

% Calculate Nth iteration
for j = 1:lin
    for n = len:-1:2
        x(n) = x(n-1);    % Delay the Nth sample
    end
    x(1) = xin(j);        % Get a new sample
    d = unsys(x(1));      % Send the new sample to unk. sys.
    y = w*x;
    e = d-y;
    mse(j+1) = e^2;       % Calculate Nth MSE value
    w = w + mu*e*x';
end

% Calculate and display the frequency spectrum of the final
% unknown system.
figure(1)
hold on
[H,f] = freqz(w,1,512,'whole');
subplot(2,1,1),plot(f,abs(H),'b')
xlabel('frequency')
ylabel('magnitude')
subplot(2,1,2),plot(f,angle(H),'b')
xlabel('frequency')
ylabel('phase')
% Calculate and plot the mean squared error for
% each iteration.
num = 0:length(mse)-1;
figure (2)
hold on
grid on
%title ('Plot of Mean Square Error vs. Iteration Number')
ylabel('MSE')

```

```
xlabel('Iteration')
plot(num,mse,'b')
% Calculate coefficient and final MSE values.
disp('Estimated coefficients w =')
disp(w')
disp('Final error e =')
disp(abs(e))
```

```

% Adaptive noise cancellation
% This M file recursively calculated the coefficients of
% an unknown system. The value mu specifies the rate
% of convergence of the adaptive algorithm. mu must
% be less than the smallest eigenvalue of the unknown
% transfer function for the adaptive filter to be properly
% conditioned. len is the order of the filter.
% lin is the number of iterations to take place.
% x is the input signal to the system.
% n0, n1 are two independent noise sources.
% Author: Thomas Drumright
% Date: 6/11/98

```

```

clear
mu = .01;           % Convergence factor
len = 40;          % Filter length
lin = 1000;        % Number of iterations

```

```

n1 = 0.7*randn(lin,1); % Input white noise
n0 = n1;
mse = 0;

```

```

for n=1:lin
    s(n) = sin(2*pi*(n)/abs(10*sin(pi*(1000+n)/3000)));
end

```

```

% Apply the adaptive algorithm
sn = s'+n1;
x = 0*(1:len)';           % Initial input
w = 0*(1:len);           % Initial filter coeffs.
% Calculate first iteration
y = w*x;
e = n0(1)-y;
mse(1) = e;
w = w + mu*e*x';
% Calculate Nth iteration
for j = 1:lin
    for n = len:-1:2
        x(n) = x(n-1);
    end
    x(1) = n0(j);
    y = w*x;
    e = sn(j)-y;
    mse(j) = e;
    w = w + mu*e*x';
end

```

```

end
% Display the time domain of the input and
% the error.
f = 0:length(sn)-1;
W = fft(w,512);
figure(1)
hold on
xlabel('time')
ylabel('noisy signal')
plot(f,sn,'b')
% Calculate and plot the mean squared error for
% each iteration.
num = 0:length(mse)-1;
figure (2)
hold on
xlabel('time')
ylabel('filtered signal')
plot(num,mse,'b')
% Calculate and plot the frequency spectrums of the
% filtered and unfiltered signals.
figure (4)
hold on
t = 0:8000/(length(W)-1):8000;
plot(t,abs(W))
figure (5)
hold on
xlabel('frequency')
ylabel('noisy signal')
plot(t,abs(fft(mse,512))./512,'b')
figure(6)
hold on
xlabel('frequency')
ylabel('filtered signal')
plot(t,abs(fft(sn,512))./512,'b')
figure(7)
hold on
xlabel('time')
ylabel('MSE')
plot(f,(s-mse).^2,'b')
% Display final coefficients and MSE
disp('Estimated coefficients w =')
disp(w')
disp('Final error e =')
disp(abs(e))
snr = 10*log10(sum(abs(fft(s)).^2)/sum(abs(fft(sn-s')).^2))
snrf = 10*log10(sum(abs(fft(s)).^2)/sum(abs(fft(mse-s)).^2))

```

```

% Adaptive Linear Predictive Coding
% This M file recursively calculated the coefficients of
% an unknown system. The value mu specifies the rate
% of convergence of the adaptive algorithm. mu must
% be less than the smallest eigenvalue of the unknown
% transfer function for the adaptive filter to be properly
% conditioned. len is the order of the filter.
% lin is the number of iterations to take place.
% x is the input signal to the system.
% n0, n1 are two independent noise sources.
% Author: Thomas Drumright
% Date: 6/11/98

mu = .0007;           % Convergence factor
len = 100;           % Filter length
lin = 5000;          % Number of iterations
n0 = .5*randn(lin,1); % Input white noise
a = 0;
mse = 0;
% Create input of the modeled system.
s = 0;
for n=1:(lin);
    s1(n) = 1*sin(2*pi.*n/abs(50*sin(pi*(2500.+n)/10000))).*sin(2*pi*n/200);
end
s = s1+n0';
% Apply the adaptive algorithm
x = 0*(1:len)';      % Initial input
w = 0*(1:len);       % Initial filter coeffs.
% Calculate Nth iteration
for j = 1:lin
    y = w*x;
    a(j) = y;
    for n = len:-1:2
        x(n) = x(n-1);
    end
    x(1) = s(j);
    e = x(1)-y;
    mse(j+1) = e;
    w = w + mu*e*x';
end
% Display the time domain of the filtered and
% unfiltered system.
f = 0:length(a)-1;
figure(1)
hold on
title('y(t)')

```

```

plot(f,a)
l = 0:len-1;
figure (2)
hold on
title('w(t)')
plot (l,w)
% Calculate and plot the mean squared error for
% each iteration.
num = 0:length(mse)-1;
figure (3)
hold on
title('e(t)')
plot(num,mse)
% Calculate and plot the frequency spectrums of the
% filtered and unfiltered signals.
W = fft(s,512);
figure (4)
hold on
title('S(w)')
t = 0:length(W)-1;
plot(t,abs(W))
figure (5)
hold on
title('s(t)')
t = 0:length(s)-1;
plot(t,s)
W = fft(a,512);
figure (6)
hold on
title('Y(w)')
t = 0:length(W)-1;
plot(t,abs(W))
W = fft(mse,512);
figure (7)
hold on
title('E(w)')
t = 0:length(W)-1;
plot(t,abs(W))
% Display final coefficients and MSE
%disp('Estimated coefficients w =')
%disp(w')
%disp('Final error e =')
%disp(abs(e))

```

```

% Adaptive Inverse System
% This M file recursively calculated the coefficients of
% an unknown system. The value mu specifies the rate
% of convergence of the adaptive algorithm. mu must
% be less that the smallest eigenvalue of the unknown
% transfer function for the adaptive filter to be properly
% conditioned. len is the order of the estimating filter.
% len must be greater than or equal to the order of the
% unknown filter to model the unknown filter properly.
% lin is the number of iterations to take place.
% xin is the input signal to the system.
% By Thomas Drumright
% 11/20/97

% Initialize variables
INPUTVEC = 0;           % Initialize a global variable
mu = .005;             % Convergence factor
len = 20;              % Filter length
lin = 5000;           % Number of iterations
mse = 0;
x = 0*(1:len)';       % Initial delayed input vector
w = 0*(1:len);        % Initial filter coeffs.
x1 = 0*(1:len)';      % Initial input vector
n0 = .1*randn(lin,1); % Unity Gaussian white noise

% Calculate input signal
n = 0:lin-1;
xin = n0'+sin(2*pi*n./(sin(2*pi*(n+1)/11000)));

% Calculate Nth iteration
for j = 1:lin
    for n = len:-1:2
        x1(n) = x1(n-1);
    end
    x1(1) = unsys2(x(1));
    b(j) = x1(1);
    y = w*x1;
    a(j) = y;
    e = x(1)-y;
    mse(j+1) = e^2;
    w = w + mu*e*x1';
    for n = len:-1:2
        x(n) = x(n-1);
    end
    x(1) = xin(j);
end

```

```

end
w'
[H,f] = freqz(w,1,512);
figure(1)
hold on
subplot(2,1,1), plot(f,20*log(abs(H)), 'b')
ylabel('H(w)^2')
xlabel('w')
subplot(2,1,2), plot(f,angle(H), 'b')
ylabel('Phase')
xlabel('w')
% Calculate and plot the mean squared error for
% each iteration.
num = 0:length(a)-1;
figure (2)
hold on
grid on
title ('y(n)')
plot(num,a, 'b')
figure (3)
hold on
grid on
title ('x(n)')
plot(num,xin, 'b')
figure(4)
w1 = [1 0.2790 -1.4488 -1.0700 0.4678 0.8100];
hold on
[H,f] = freqz(w1,1,512);
subplot(2,1,1), plot(f,20*log(abs(H)), 'b')
ylabel('H(w)^2')
xlabel('w')
subplot(2,1,2), plot(f,angle(H), 'b')
ylabel('Phase')
xlabel('w')
figure(5)
hold on
grid on
title ('UW(k)*W(k)')
w2 = conv(w1,w);
[H,f] = freqz(w2,1,512);
subplot(2,1,1), plot(f,20*log(abs(H)), 'b')
ylabel('H(w)^2')
xlabel('w')
subplot(2,1,2), plot(f,angle(H), 'b')
ylabel('Phase')
xlabel('w')

```

```
disp('Final error e =')  
disp(abs(e))
```

```

% Unknown System Function
% This function calculates the output of a single input,
% single output system. This function takes samples
% from a vector and calculate the output based on past
% samples and the current sample.
% INPUTVEC - a global variable that will store a number
% of samples equal to the length of the unknown system.
% uw - the unknown system coefficients.
% dn - the output of the unknown system.
%
% By Thomas Drumright
% 12/12/97

function dn = unsys(xn)
global INPUTVEC % Global variable
uw = [1 -1.416 5 -1.416 1]; % System Coefficients
len = length(uw); % Length of system
INPUTVEC(1) = xn; % get new sample
if length(INPUTVEC) ~= len
    INPUTVEC(len) = 0;
end
dn = uw*INPUTVEC'; % convolve samples with uw
for i = len:-1:2 % update coefficients
    INPUTVEC(i) = INPUTVEC(i-1);
end

```

```

% Unknown System Function 2
% This function calculates the output of a single input,
% single output system. This function takes samples
% from a vector and calculate the output based on past
% samples and the current sample.
% INPUTVEC - a global variable that will store a number
% of samples equal to the length of the unknown system.
% uw - the unknown system coefficients.
% dn - the ouput of the unknown system.
%
% By Thomas Drumright
% 12/12/97

function dn = unsys(xn)
global INPUTVEC % Global variable
uw = [1 0.2790 -1.4488 -1.0700 0.4678 0.8100]; % System Coefficients
len = length(uw); % Length of system
INPUTVEC(1) = xn; % get new sample
if length(INPUTVEC) ~= len
    INPUTVEC(len) = 0;
end
dn = uw*INPUTVEC'; % convolve samples with uw
for i = len:-1:2 % update coefficients
    INPUTVEC(i) = INPUTVEC(i-1);
end

```

List of References

1. Haykin, Simon, *Adaptive Filter Theory*, Prentice Hall, Upper Saddle River, New Jersey, 1996.
2. Honig, Michael L., Messerschmitt, David G., *Adaptive Filters, Structures, Algorithms and Applications*, Kluwer Academic Publishers, Boston, 1984.
3. Jenkins, W. Kenneth, Hull, Andrew W., Strait, Jeffrey C., Schnauffer, Bernard A., Li, Xiaohui, *Advanced Concepts in Adaptive Signal Processing*, Kluwer Academic Publishers, Boston, 1996.