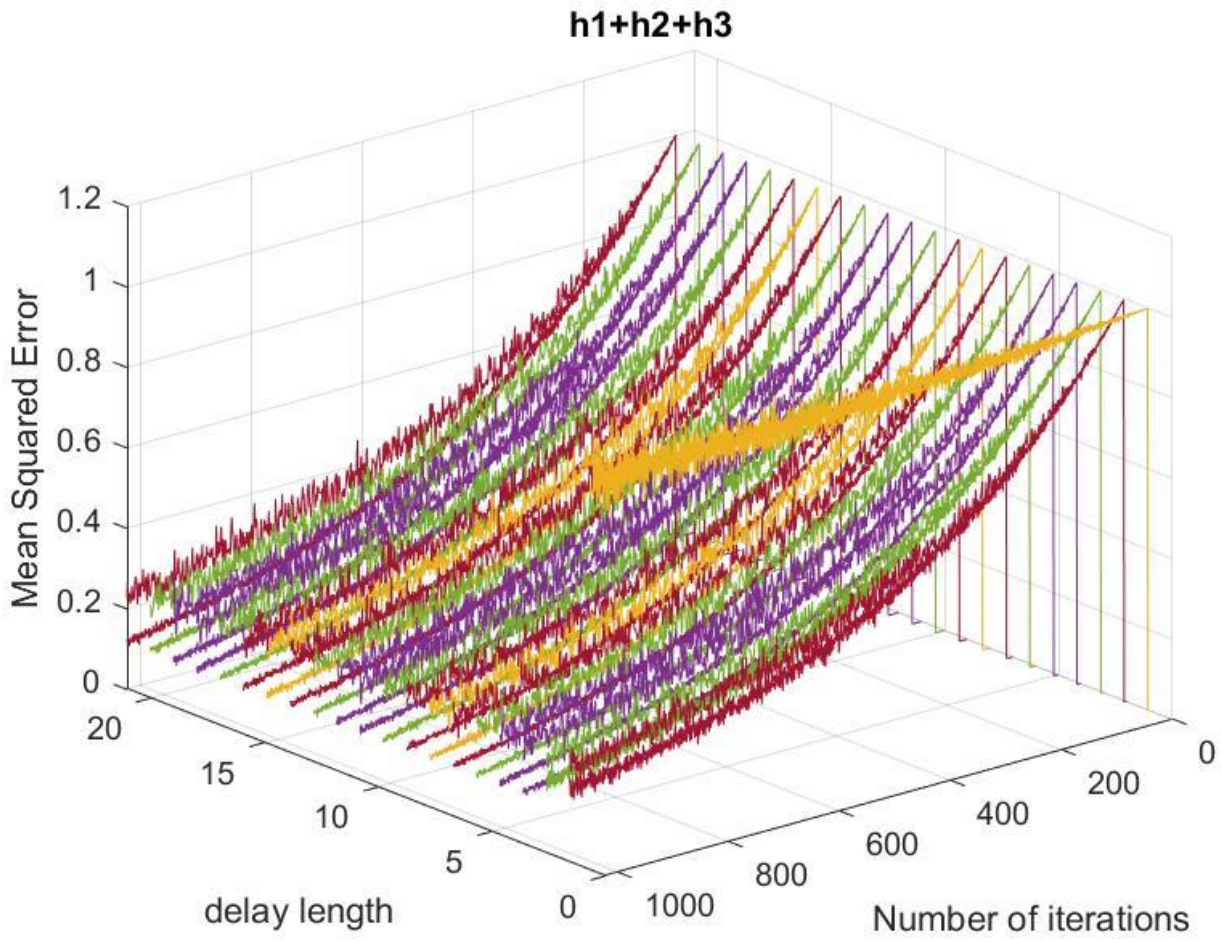
6.18

(a)



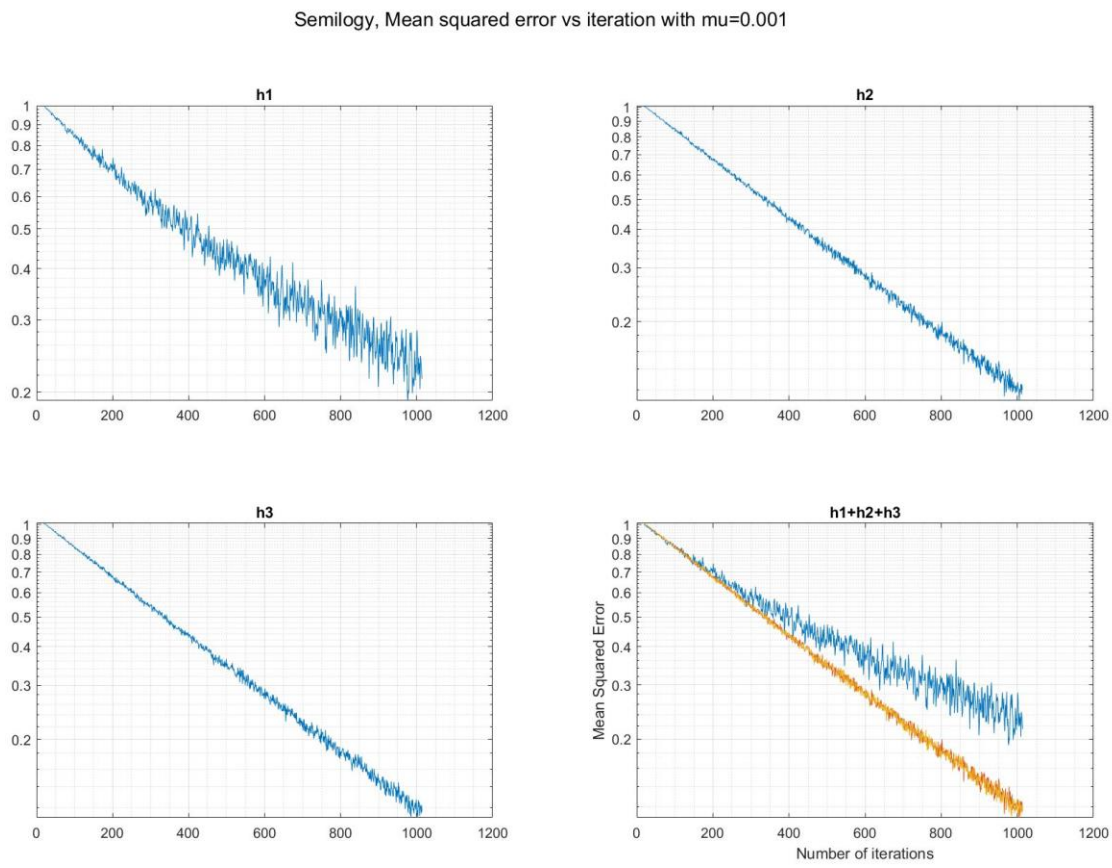
In the picture it does not have a significant final value of the mean squared error from different delay. So, I try choosing delta = (21+1)/2 = 11 as delay in the (b) part maybe works well.

```
function function_618_adaptive_filter_theory
% 6880
% Zeyu Liu
% 2/11/2020
% Adaptive filter theory 5 edition
% problem 6.18
num_data = 1000; % the process literature
num_runs = 100; % Monte Cario 100 times,increase this will get smoothed
image
mu = 0.001; % step-size parameter
NoiseVariance = 0.01; % the variance of noise
Noise_sde = sqrt(NoiseVariance); % stand err of noise
```

```
stream = RandStream('mt19937ar','Seed',1); % Monte Cario seed random
RandStream.setGlobalStream(stream); % generator for reproducable

% a.Determing the delay.
h1=[0.25,1,0.25];
h2=[0.25,1,-0.25];
h3=[-0.25,1,0.25];
subplot(1,1,1)
for m = 1:3
    if m == 1
        h = h1;
    elseif m == 2
        h = h2;
    else
        h = h3;
    end
    for taps = 1:21
        x = zeros(num_data+3+taps,1); % input data stream
        u = zeros(num_data+3+taps,1); % recieved data, delay data
        e = zeros(num_data+3+taps,1); % error between prediction and delay
        g = zeros(num_data+3+taps,1); % squared averaged error
        for k=1:num_runs % Loop for performing the appropriate number of
Monte Carlo simulations
            x=zeros(num_data,1); % Initialize the weights back to zero as
well
            W=zeros(taps,1); % initial weights to 0
        for n=3+taps:num_data+3+taps % The loop that does the data runs
and %filter updates
            x(n)=binornd(1,0.5)*2-1; % binary -1 or 1
            u(n)=x(n)*h(3)+x(n-1)*h(2)+x(n-2)*h(1)+randn(1)*Noise_sde;
            e(n)=x(n-floor(taps/2))-W'*u(n:-1:n-taps+1); % calculate the
error in estimation
            W=W+mu*u(n:-1:n-taps+1)*e(n); % update the weights in the manner
associated
        % LMS based on the recient error in prediction of the last input
        end
        g=g+e.^2; % accumulate squared error of estimation
        end
        g=g/num_runs; % normalize the accumulated error to reflect
        plot3(taps*ones(num_data+3+taps),[1:num_data+3+taps],g);
        title('h1+h2+h3');
        xlabel('delay length')
        ylabel('Number of iterations')
        zlabel('Mean Squared Error')
        hold on
        grid on
    end
end
pause;
hold off
```

(b)



The picture shows the transfer function parameter will affect the stability of convergence.

The more stable the convergence, the better the local optimal solution can be achieved.

```
% b.plot learning curve
x = zeros(num_data+14,1); % input data stream
u = zeros(num_data+14,1); % recieved data, delay data
e = zeros(num_data+14,1); % error between prediction and delay
g = zeros(num_data+14,1); % squared averaged error
h1=[0.25,1,0.25];
h2=[0.25,1,-0.25];
h3=[-0.25,1,0.25];
for m = 1:3
    if m == 1
        h = h1;
    elseif m == 2
        h = h2;
    else
        h = h3;
```

```
    end
    for k=1:num_runs % Loop for performing the appropriate number of Monte
Carlo simulations
        x=zeros(num_data,1); % Initialize the weights back to zero as well
        W=zeros(11,1); % initial weights to 0
    for n=3+11:num_data+3+11 % The loop that does the data runs and %filter
updates
        x(n)=binornd(1,0.5)*2-1; % binary -1 or 1
        u(n)=x(n)*h(3)+x(n-1)*h(2)+x(n-2)*h(1)+randn(1)*Noise_sde;
        e(n)=x(n-5)-W'*u(n:-1:n-10); % calculate the error in estimation
        W=W+mu*u(n:-1:n-10)*e(n); % update the weights in the manner
associated
        % LMS based on the recient error in prediction of the last input
    end
    g=g+e.^2; % accumulate squared error of estimation
    end
    g=g/num_runs; % normalize the accumulated error to reflect
% the average error based on the number of Monte Carlo simulations completed
    subplot(2,2,m)
    semilogy([1:num_data+14],g) % plot the MSE vs iterations
    grid on
    grid minor
    title(['h',num2str(m)]);
    subplot(2,2,4)
    semilogy([1:num_data+14],g)
    title('h1+h2+h3')
    xlabel('Number of iterations') % x-axis label
    %xlim([0 1000])
    %ylim([1e-1 1.1])
    ylabel('Mean Squared Error') % y-axis label
    grid on
    grid minor
    hold on
end
suptitle('Semilogy, Mean squared error vs iteration with mu=0.001');
```