English Vowel Phonemes Identification

Submitted to: Dr. Qadri Mayyala

Department of Electrical and Computer Engineering

Birzeit University

Group members: Yafa Naji 1200708 Nour Rahib , 1202853 Dunia Hamada 1201001

The project abstract on system identification methods and techniques from traditional to modern and offline to online rapid analysis during the previous years to the present days. It takes into account part of AI technologies (soft computing technologies), primarily artificial neural network methods, for off-line system data analysis and real-time system identification, including linear and nonlinear issues. This study will provide a very brief survey of the researchers on the development of the definition of the system, its applications, and the techniques that were used.

INTRODUCTION

The process of identifying the mathematical model that explains the behavior of a physical system based on a collection of input-output data is known as system identification. The goal of this project is to create a system identification algorithm capable of properly identifying the underlying dynamics of a physical system from a collection of measurable inputs and outputs.

EVALUATION CRITERIA

Project evaluation mainly depends on the accuracy of the result. Accuracy of our project results is developed by working on the project. The results were fairly accurate, after working more on the project and studying more MATLAB and DSP techniques, the results were revised to give better accuracy. Results are developed and filter performance is evaluated by comparing the filtered output with the desired output, using metrics such as mean squared error, signal-to-noise ratio, or other performance criteria.

By working on the second part, we used a filter, and it was noticed that there were not many differences.

.PROBLEM SPECIFICATION

The problem addressed in this project is to develop a system identification algorithm that can accurately identify the underlying dynamics of a physical system from a set of measured inputs and outputs. The algorithm should be able to handle nonlinear, time-varying systems with varying levels of noise and uncertainty.

DATA

This project's data will be a mix of real-world signals and synthetic signals created via a physics-based simulation tool. The synthetic signals will be created using a physics-based simulation tool, while the real-world signals will be gathered from physical systems such as mechanical, electrical, and fluid systems.

APPROACH

The model was built in two different ways, but the results are similar and differ in complexity and speed.

An adaptive filter generally performs a spatial operation to identify pixels affected by pulse noise. The first method is applied "LMSFilter"() in Matlab. Using the lms() function. The second method is applied "RLSFilter" using Matlab's rls() function. We used desired And we got the value of the error in each of them

RESULTS AND ANALYSIS

In general, the results from using LMS or RLS filters can include improved signal quality, reduced noise and interference, enhanced system performance, and increased accuracy of the estimated parameters. However, these results can also be affected by factors such as the convergence rate, the initialization of the filter parameters, and the presence of nonlinearities in the system.

It is important to carefully design and evaluate the performance of the LMS or RLS filter in the specific context of each application, taking into account the trade-offs between computational complexity, convergence speed, and accuracy of the estimated parameters.

DEVELOPMENT

Enormous progress has been accomplished It is still the major task in any advanced control project Major challenge today: reduce the cost of identification

- Make it more user friendly and data-driven
- Optimize the experiment: less time, less energy
- Tune the experiment towards the application: don't waste

Many research challenges remain, new ones have appeared:

- Structured systems
- Nonlinear systems
- Large distributed and network controlled systems

CONCLUSION

The main operations of this project (System Identification) have included many concepts and basics of the DSP. The project has included dealing with filters main concept. And for sure many other concepts of DSP. Also, many MATLAB skills were gained after working on such a project. After finish working on the project, very satisfying results were obtained. But still, many more developments and features can be added to the project.

REFERENCES

https://ch.mathworks.com/help/ident/gs/about-system-identification.html

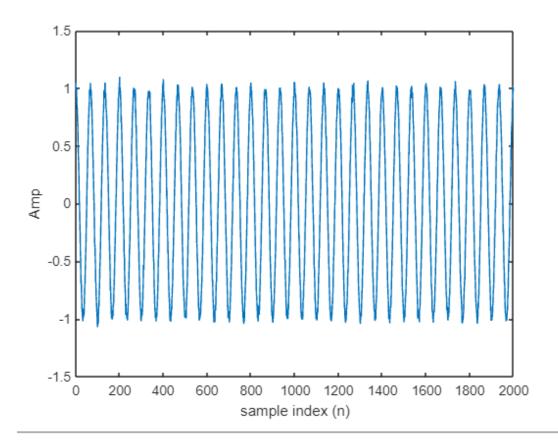
https://ch.mathworks.com/help/dsp/ref/dsp.lmsfilter-system-object.html

https://ch.mathworks.com/help/dsp/ref/dsp.rlsfilter-system-object.html

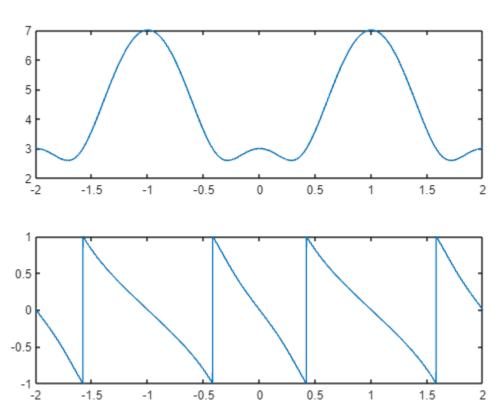
Part1

A.

```
N=2000;
n=0:N-1;
x = cos(0.03*pi*n);
xn=awgn(x,30)
plot(n,xn)
xlabel("sample index (n)");
ylabel("Amp");
```

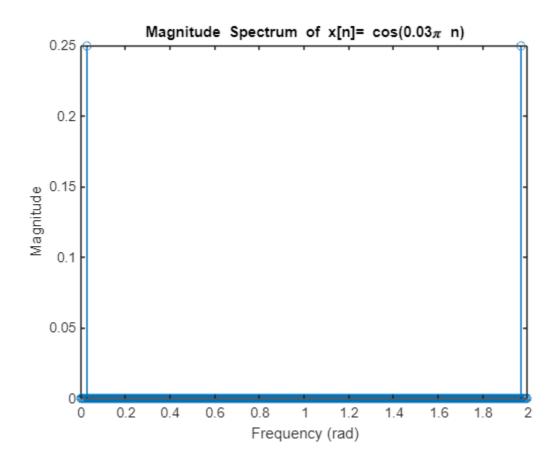


```
B.
w=-2:0.001:2;
w=w*pi;
H=1-2*exp(-j*w) +4*exp(-j*2*w);
figure
subplot(211),plot(w/pi,abs(H))
subplot(212),plot(w/pi,angle(H)/pi)
```



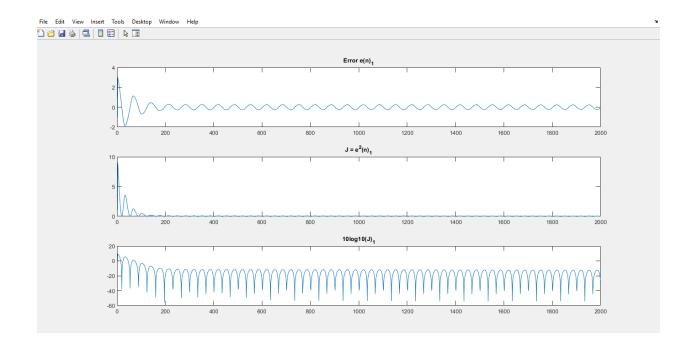
C.

```
N = 2000;
n = 0:N-1;
x = cos(0.03*pi*n);
stem(0:2/N:2-1/N, abs(fft(x)/N).^2);
xlabel('Frequency (rad)')
ylabel('Magnitude')
title('Magnitude Spectrum of x[n]= cos(0.03\pi n)')
```



```
R2020b ▶ bin ▶
```

```
Editor - C:\Users\Asus\OneDrive\Desktop\DSP_Project\d_1.m
  [ E.m × d_1.m × d_2.m × f_1.m × e_2.m × f_2.m × + ]
  1
  2 -
         N=2000;
         M = 3; % number of filter coefficients
  3 -
         x = cos(0.03*pi*(0:N-1)'); % input signal
  5 -
         w=[0,0,0,0];
  6 -
         s=[1 -2 4];
  7 -
         mu=0.01;
  8 -
         a=1;
  9 -
         d=filter(s,a,x);
  10
         % Create LMS filter object
  11 -
         lms= dsp.LMSFilter('Length', M, 'StepSize', mu);
  12
  13
         % Filter the input signal
  14 -
         [y,err,wts] = lms(x,d);
  15
         N = length(x);
  16 -
  17 -
       \bigcirc for n = 4:N
  18 -
            x_{in} = cos(0.03*pi*n); % get the input vector
  19 -
             y_est = w(n)' * x_in; % estimate the output
  20 -
             e(n) = d(n) - y_est; % calculate the error
  21 -
             w(n+1) = w(n) + 2*mu*e(n)*x_in; update the filter coefficients
  22 -
  23
         % Plot the learning curves
  24
  25 -
         figure;
 26 -
         subplot(3,1,1);
24
        % Plot the learning curves
25 -
        figure;
26 -
        subplot (3,1,1);
27 -
        plot(err);
        title('Error e(n)_1');
28 -
29
30 -
        subplot (3,1,2);
31 -
        J = err.^2;
32 -
        plot(J);
33 -
        title('J = e^2(n) 1');
34
35 -
        subplot (3,1,3);
36 -
        logJ = 10*log10(J);
37 -
        plot(logJ);
38 -
        title('10log10(J) 1');
```

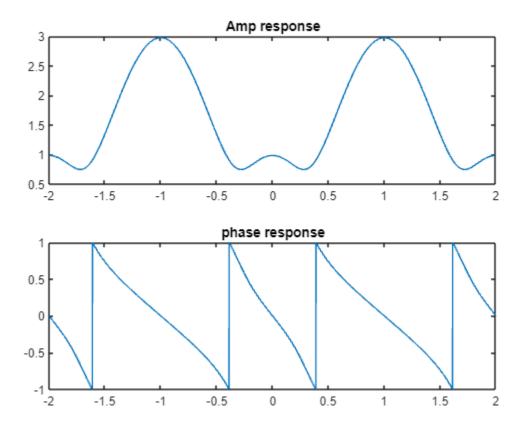


The estimate of W

```
2.8504
                     2.8498
                                2.8508
                                          2.8533
                                                                         2.8686
                                                                                                        2.8915
                                                                                                                  2.8997
Columns 1, YYY through 1, YAA
          2.9152
                      2.9220
                                2.9280
                                           2.9328
                                                     2.9363
                                                               2.9383
                                                                         2.9389
                                                                                    2.9378
                                                                                              2.9352
                                                                                                        2.9311
                                                                                                                  2.9256
Columns 1, YA4 through 1, A..
 2.9190
            2.9114
                      2.9032
                                2.8947
                                           2.8861
                                                     2.8778
                                                               2.8702
                                                                         2.8635
                                                                                    2.8579
                                                                                              2.8537
                                                                                                        2.8510
                                                                                                                  2.8498
Columns 1, A.1 through 1, A1Y
  2.8503
            2.8523
                      2.8557
                                2.8605
                                           2.8664
                                                     2.8733
                                                               2.8808
                                                                         2.8888
                                                                                              2.9050
                                                                                                        2.9127
                                                                                                                  2.9198
                                                                                    2.8970
Columns 1, A17 through 1, AYE
  2.9261
            2.9313
                      2.9353
                                2.9378
                                                                                              2.9213
                                           2.9389
                                                     2.9383
                                                               2.9362
                                                                         2.9326
                                                                                   2.9276
                                                                                                        2.9140
                                                                                                                  2.9060
Columns 1, AYo through 1, AY1
            2 0000
```

\mathbf{E}

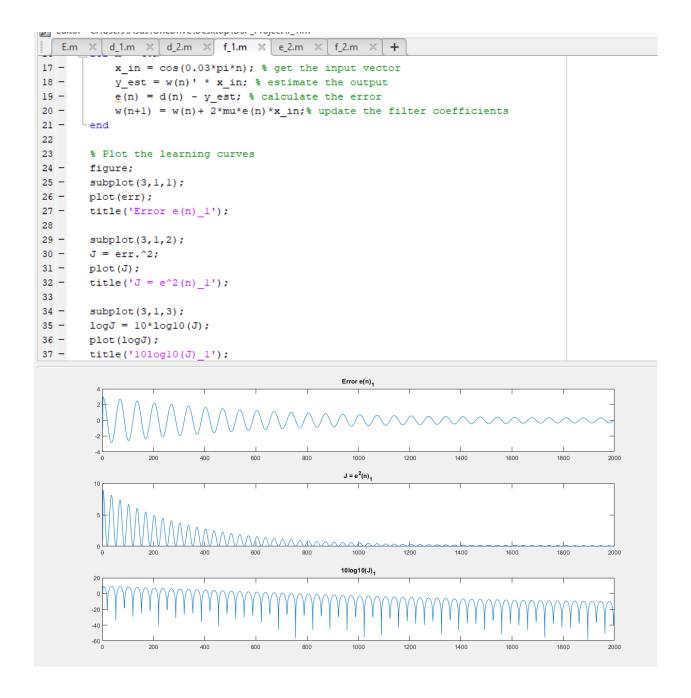
```
w=-2:0.001:2;
w=w*pi;
H1=0.5357-0.9933*exp(-j*w) +1.4421*exp(-j*2*w);
figure
subplot(211);|
plot(w/pi,abs(H1));
title("Amp response");
subplot(212);
plot(w/pi,angle(H1)/pi);
title("phase response");
```



The difference between it and branch B is the change the amplitude due to the change in the coefficients

```
F
```

```
Editor - C:\Users\Asus\OneDrive\Desktop\DSP_Project\t_1.m
       f_2.m
 1 -
       N=2000;
 2 -
       M = 3; % number of filter coefficients
       x = cos(0.03*pi*(0:N-1)'); % input signal
 3 -
       w=[0,0,0,0];
 5 -
       s=[1 -2 4];
 6 -
       mu=0.001; %Decrease the value of μ.
 7 -
       a=1;
       d=filter(s,a,x);
 8 -
 9
        % Create LMS filter object
10 -
       lms= dsp.LMSFilter('Length', M, 'StepSize', mu);
11
12
       % Filter the input signal
13 -
        [y,err,wts] = lms(x,d);
15 -
       N = length(x);
16 -
     \neg for n = 4:N
17 -
           x in = cos(0.03*pi*n); % get the input vector
```

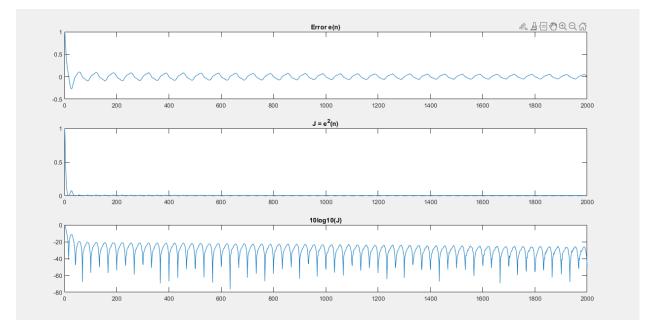


Decreasing the value of μ will decrease the speed of the learning process. This means that the controller will respond more slowly to changes in the system, which can be beneficial for avoiding overshoot and oscillation. However, it can also result in slower convergence to the desired setpoint.

Steady-state error: Decreasing the value of μ will increase the steady-state error. This means that the final error between the actual output and the desired setpoint will be larger. However, it can also result in a more stable system, as the controller will be less sensitive to noise and disturbances in the system.

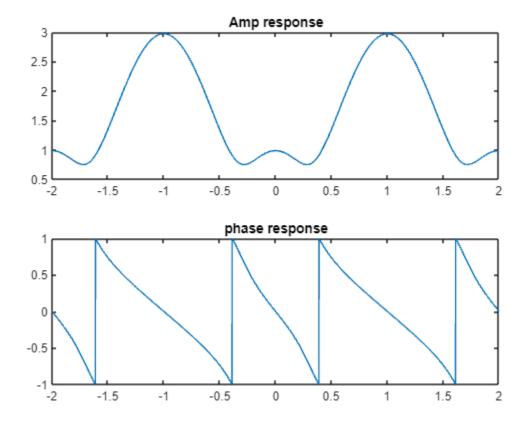
G_as_D

```
N = 2000; % number of samples
 x = cos(0.03*pi*(0:N-1)'); % input signal
 d = x; % desired signal
 mu = 0.01; % step size
 M = 3; % number of filter coefficients
 x = x + awgn(x, 40, 'measured');
 % Create LMS filter object
 lms = dsp.LMSFilter('Length', M, 'StepSize', mu);
  [y,err,wts] = lms(x,d);
 N = length(x);
\Box for n = 4:N
     x_{in} = cos(0.03*pi*n); % get the input vector
     y_est = w(n)' * x_in; % estimate the output
     e(n) = d(n) - y_est; % calculate the error
     w(n+1) = w(n) + 2*mu*e(n)*x in;% update the filter coefficients
 end
 % Plot the learning curves
 figure;
 subplot(3,1,1);
 plot(err);
 title('Error e(n)');
 subplot(3,1,2);
 J = err.^2;
 plot(J);
 title('J = e^2(n)');
 subplot(3,1,3);
 logJ = 10*log10(J);
 plot(logJ);
 title('10log10(J)');
```



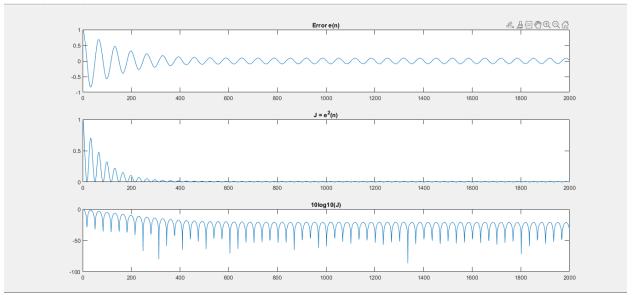
G_as_E

```
w=-2:0.001:2;
w=w*pi;
H1=0.5357-0.9933*exp(-j*w) +1.4421*exp(-j*2*w);
x=H1;
noisy_x=awgn(x,40);
figure
subplot(2,1,1);
plot(w/pi,abs(H1));
title("Amp response");
subplot(2,1,2);
plot(w/pi,angle(H1)/pi);
title("phase response");
```



G_as_F

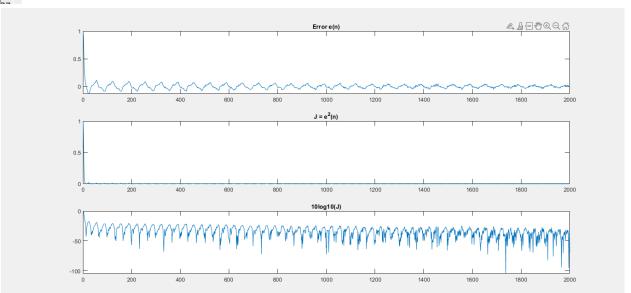
```
1 -
       N = 2000;
2 -
       x = cos(0.03*pi*(0:N-1)'); % input signal
3 -
       d = x;
 4 -
       mu = 0.001;
       M = 3;
 5 -
 6 -
       x = x + awgn(x, 40, 'measured');
 7
 8 -
       lms = dsp.LMSFilter('Length', M, 'StepSize', mu);
9
        [y,err,wts] = lms(x,d);
10 -
11
12
13 -
       N = length(x);
14 -
      \neg for n = 4:N
15 -
            x_{in} = cos(0.03*pi*n); % get the input vector
16 -
           y_{est} = w(n)' * x_{in}; % estimate the output
17 -
           e(n) = d(n) - y_est; % calculate the error
           w(n+1) = w(n) + 2*mu*e(n)*x_in;% update the filter coefficients
18 -
19 -
20
21
       % Plot the learning curves
22 -
       figure;
23 -
       subplot(3,1,1);
24 -
       plot(err);
25 -
        title('Error e(n)');
26
27 -
       subplot(3,1,2);
28 -
       J = err.^2;
29 -
       plot(J);
30 -
        title('J = e^2(n)');
31
32 -
       subplot(3,1,3);
33 -
       logJ = 10*log10(J);
34 -
       plot(logJ);
35 -
        title('10log10(J)');
36
```



After adding 40dB of zero mean white Gaussian noise to the signal x[n], the results of parts (D)-(F) change. This is because the added noise may cause the filter to perform less accurately

H as D

```
%30dB step size =0.02
1
2
3 -
       N = 2000; % number of samples
4 -
       x = cos(0.03*pi*(0:N-1)'); % input signal
5 -
       d = x; % desired signal
6 -
       mu = 0.02; % step size
7 -
       M = 3; % number of filter coefficients
8 -
       x = x + awgn(x, 30, 'measured');
       % Create LMS filter object
9
10 -
       lms = dsp.LMSFilter('Length', M, 'StepSize', mu);
11 -
       [y,err,wts] =lms(x,d);
12
13
14 -
15 -
       N = length(x);
     \neg for n = 4:N
16 -
            x_in = cos(0.03*pi*n); % get the input vector
17 -
            y_est = w(n)' * x_in; % estimate the output
18 -
            e(n) = d(n) - y_{est}; % calculate the error
19 -
            w(n+1) = w(n) + 2*mu*e(n)*x_in;% update the filter coefficients
20 -
       ^{\mathsf{L}} end
21
22
       % Plot the learning curves
23 -
       figure;
24 -
       subplot(3,1,1);
25 -
       plot(err);
26 -
       title('Error e(n)');
27
28 -
       subplot(3,1,2);
29 -
       J = err.^2;
30 -
       plot(J);
31 -
       title('J = e^2(n)');
32
33 -
       subplot(3,1,3);
34 -
        logJ = 10*log10(J);
35 -
       plot(logJ);
36 -
       title('10log10(J)');
```

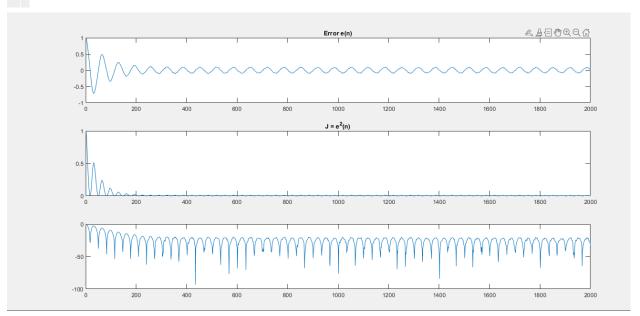


```
H_as_E
```

```
1 -
       w=-2:0.001:2;
 2 -
       w=w*pi;
 3
       H=1-2*exp(-i*w) +4*exp(-i*2*w);
 4 -
       H1=0.5357-0.9933*exp(-j*w) +1.4421*exp(-j*2*w);
 5 -
6
       % Add noise
7
       x = H1:
8 -
       noisy_x = awgn(x, 30);
9 -
10
       figure
11 -
       subplot(211),plot(w/pi,abs(noisy_x))
12 -
       subplot(212),plot(w/pi,angle(noisy_x)/pi)
13 -
```

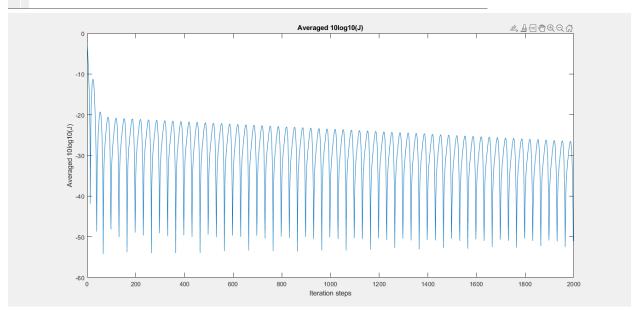
H as F

```
1 -
       N = 2000; % number of samples
2 -
       x = cos(0.03*pi*(0:N-1)'); % input signal
3 -
       d = x; % desired signal
4 -
       mu = 0.002; % step size
5 -
       M = 3; % number of filter coefficients
       x = x + awgn(x, 30, 'measured');
6 -
7
       % Create LMS filter object
8 -
       lms = dsp.LMSFilter('Length', M, 'StepSize', mu);
9
10 -
       [y,err,wts] = lms(x,d);
11
12
13 -
       N = length(x);
14 -
     \neg for n = 4:N
15 -
           x_in = cos(0.03*pi*n); % get the input vector
16 -
           y_{est} = w(n)' * x_{in}; % estimate the output
17 -
           e(n) = d(n) - y_est; % calculate the error
18 -
           w(n+1) = w(n) + 2*mu*e(n)*x_in;% update the filter coefficients
19 -
20
       % Plot the learning curves
21
       figure;
22 -
23 -
       subplot(3,1,1);
24 -
       plot(err);
25 -
       title('Error e(n)');
26
27 -
       subplot(3,1,2);
28 -
       J = err.^2;
29 -
       plot(J);
       title('J = e^2(n)');
30 -
31
32 -
       subplot(3,1,3);
33 -
       logJ = 10*log10(J);
34 -
       plot(logJ);
```



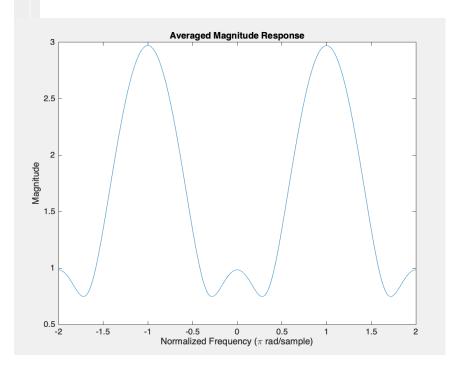
I as D

```
1 -
       num_trials = 1000;
2 -
       J_avg = zeros(N-3,1);
3
4 -
5 -
     \neg for i = 1:num_trials
           x = cos(0.03*pi*(0:N-1)');
6 -
           d = x;
7 -
           mu = 0.01;
8 -
           M = 3;
9 -
           x = x + awgn(x, 40, 'measured');
           % Create LMS filter object
10
           lms = dsp.LMSFilter('Length', M, 'StepSize', mu);
11 -
12
13 -
           [y,err,wts] =lms(x,d);
14
           J = zeros(N-3,1); % initialize J with zeros before each trial
15 -
16 -
           for n = 4:N
17 -
                x_in = cos(0.03*pi*n); % get the input vector
18 -
                y_{est} = w(n)' * x_{in};
19 -
                e(n) = d(n) - y_est;
20 -
                w(n+1) = w(n) + 2*mu*e(n)*x_in;% update the filter coefficients
21 -
                J(n-3) = err(n)^2;
22 -
           end
23 -
           J_avg = J_avg + J;
24 -
       end
25
26 -
       J_avg = J_avg / num_trials;
27 -
       logJ_avg = 10*log10(J_avg);
28
29
       % Plot the averaged learning curves
       figure;
30 -
31 -
       plot(logJ_avg);
32 -
       title('Averaged 10log10(J)');
33 -
       xlabel('Iteration steps');
34 -
       ylabel('Averaged 10log10(J)');
```



I_as_E

```
1 -
       num_trials = 1000;
2 -
       w = -2:0.001:2;
3 -
       w = w * pi;
4 -
       N = length(w);
5 -
       H_{avg} = zeros(1, N);
 6
7 -
8 -
     □ for i = 1:num_trials
           H1=0.5357-0.9933*exp(-1i*w) +1.4421*exp(-1i*2*w);
9 -
           noisy_x = awgn(H1, 40);
10 -
           H_avg = H_avg + abs(noisy_x);
11 -
       end
12
13 -
       H_avg = H_avg / num_trials;
14
       % Plot the averaged magnitude response
15
16 -
       figure;
17 -
       plot(w/pi, H_avg);
18 -
       title('Averaged Magnitude Response');
19 -
       xlabel('Normalized Frequency (\pi rad/sample)');
20 -
       ylabel('Magnitude');
```

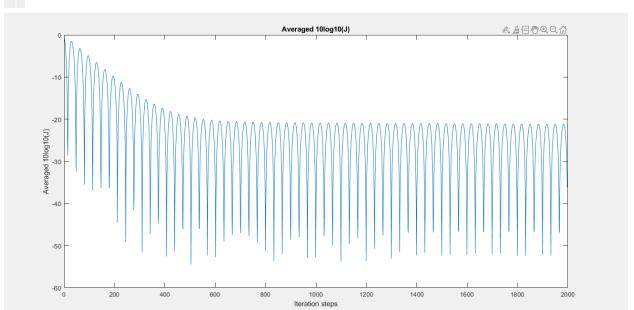


I as F

```
1 -
       num_trials = 1000;
2 -
       J_avg = zeros(N-3,1);
3
4 -

¬ for i = 1:num_trials

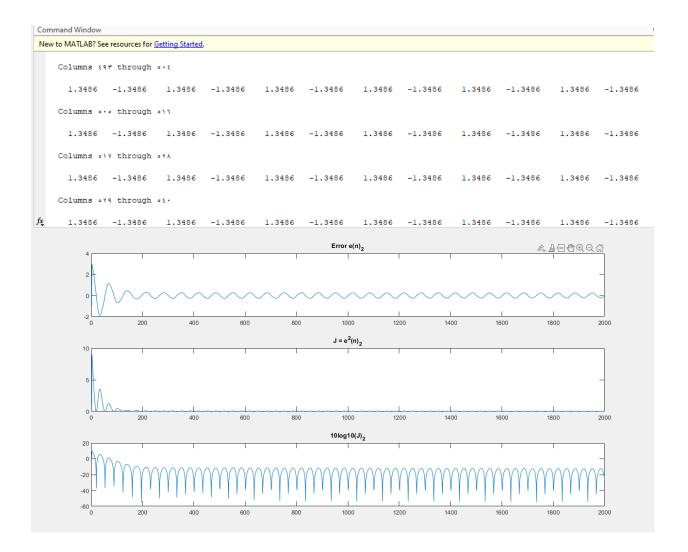
5 -
           x = cos(0.03*pi*(0:N-1)'); % input signal
6 -
            d = x; % desired signal
7 -
            mu = 0.001; % step size
8 -
           M = 3; % number of filter coefficients
9 -
           x = x + awgn(x, 40, 'measured');
10
            % Create LMS filter object
11 -
            lms = dsp.LMSFilter('Length', M, 'StepSize', mu);
12
13 -
            [y,err,wts] =lms(x,d);
14
15 -
            J = zeros(N-3,1); % initialize J with zeros before each trial
16 -
            for n = 4:N
17 -
18 -
                x_in = cos(0.03*pi*n); % get the input vector
                y_{est} = w(n)' * x_{in}; % estimate the output
19 -
                e(n) = d(n) - y_{est}; % calculate the error
20 -
                w(n+1) = w(n) + 2*mu*e(n)*x_in;% update the filter coefficients
21 -
                J(n-3) = err(n)^2;
22 -
            end
23 -
            J_avg = J_avg + J;
24 -
       end
25
26 -
       J_avg = J_avg / num_trials;
27 -
       logJ_avg = 10*log10(J_avg);
28
       % Plot the averaged learning curves
29
30 -
       figure;
31 -
       plot(logJ_avg);
32 -
       title('Averaged 10log10(J)');
33 -
       xlabel('Iteration steps');
34 -
       ylabel('Averaged 10log10(J)');
```



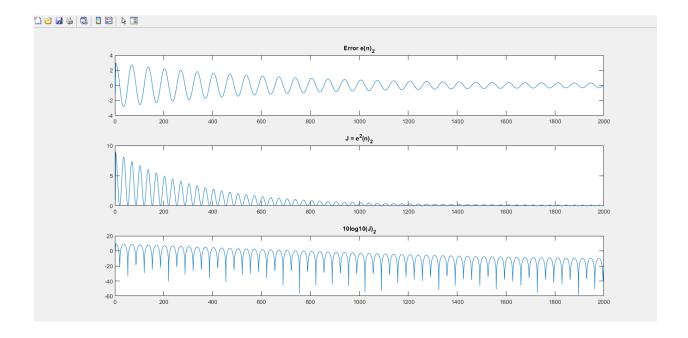
Part 2

D

```
Editor - C:\Users\Asus\OneDrive\Desktop\DSP_Project\d_2.m
    E.m \times d_1.m \times d_2.m \times f_1.m \times e_2.m \times f_2.m \times
 1 -
         N=2000:
 2 -
        M = 3; % number of filter coefficients
 3 -
        x = cos(0.03*pi*(0:N-1)'); % input signal
 4 -
        w=[0,0,0,0];
 5 -
        s=[1 -2 4];
 6 -
        mu=0.01;
 7 -
         a=1;
 8 -
        d=filter(s,a,x);
 9
10
        % Create RLS filter object
        rls = dsp.RLSFilter(11, 'ForgettingFactor', 0.98);
12
13
         % Filter the input signal
14 -
        [y,e] = rls(x,d);
15
16 - for n = M:N
17 -
            x_{in} = cos(0.03*pi*n); % get the input vector
18 -
             y = st = w(n) \cdot * x in; % estimate the output
             e(n) = d(n) - y_est;
19 -
20 -
            mu = 1 / (x in' * x in);
21 -
             w(n+1) = w(n) + 2*mu*e(n)*x in;
P KZUZUD P DIN P
  Editor - C:\Users\Asus\OneDrive\Desktop\DSP_Project\d_2.m
      E.m \times d_1.m \times d_2.m \times f_1.m \times e_2.m \times f_2.m \times +
              y_{est} = w(n)' * x in; % estimate the output
  18 -
  19 -
              e(n) = d(n) - y_est;
  20 -
              mu = 1 / (x_in' * x_in);
  21 -
              w(n+1) = w(n) + 2*mu*e(n)*x in;
  22 -
              J(n) = e(n)^2;
  23 -
        L end
  24
          % Plot the learning curves
  25 -
         figure;
  26 -
         subplot(3,1,1);
  27 -
         plot(err);
  28 -
         title('Error e(n) 2');
  29
  30 -
         subplot(3,1,2);
  31 -
         J = err.^2;
  32 -
         plot(J);
  33 -
         title('J = e^2(n) 2');
  34
  35 -
         subplot(3,1,3);
  36 -
          logJ = 10*log10(J);
  37 -
         plot(logJ);
  38 -
          title('10log10(J) 2');
```

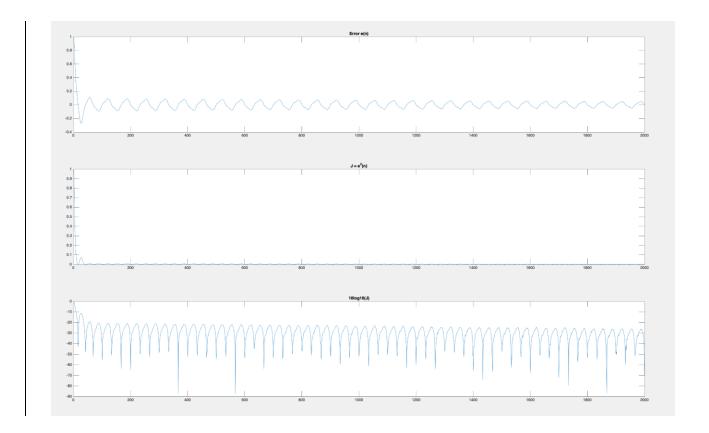


```
Z Editor - C:\Users\Asus\OneDrive\Desktop\DSP_Project\f_2.m
  E.m × d_1.m × d_2.m × f_1.m × e_2.m × f_2.m × +
 1 -
       N=2000;
 2 -
       M = 3; % number of filter coefficients
 3 -
       x = cos(0.03*pi*(0:N-1)'); % input signal
 4 -
       w=[0,0,0,0];
 5 -
       s=[1 -2 4];
 6 -
       mu=0.001; %Decrease the value of u.
 7 -
        a=1;
 8 -
       d=filter(s,a,x);
 9 -
       rls = dsp.RLSFilter(11, 'ForgettingFactor', 0.98);
 10 -
       [y,e] = rls(x,d);
 11
 12 -
      for n = M:N
 13 -
           x_{in} = cos(0.03*pi*n); % get the input vector
 14 -
           y_{est} = w(n) \cdot * x_{in}; % estimate the output
           e(n) = d(n) - y_est;
mu = 1 / (x_in' * x_in);
 15 -
 16 -
           w(n+1) = w(n) + 2*mu*e(n)*x_in;
 17 -
 18 -
           J(n) = e(n)^2;
 19 -
      end
 20
       % Plot the learning curves
 21 -
       figure;
 22 -
       subplot(3,1,1);
 23 -
       plot(err);
 24 -
       title('Error e(n)_2');
Editor - C:\Users\Asus\OneDrive\Desktop\DSP_Project\f_2.m
    E.m \times d_1.m \times d_2.m \times f_1.m \times e_2.m \times f_2.m \times +
10 -
         [y,e] = rls(x,d);
12 -
      - for n = M:N
13 -
              x_{in} = cos(0.03*pi*n); % get the input vector
14 -
              y_{est} = w(n) \cdot * x_{in}; % estimate the output
              e(n) = d(n) - y_est;
15 -
16 -
              mu = 1 / (x_in' * x_in);
17 -
              w(n+1) = w(n) + 2*mu*e(n)*x in;
18 -
              J(n) = e(n)^2;
19 -
        ∟end
20
        % Plot the learning curves
21 -
         figure;
22 -
         subplot (3,1,1);
23 -
         plot(err);
24 -
         title('Error e(n) 2');
25
26 -
         subplot(3,1,2);
27 -
         J = err.^2;
28 -
         plot(J);
29 -
         title('J = e^2(n) 2');
30
31 -
         subplot(3,1,3);
32 -
         logJ = 10*log10(J);
33 -
         plot(logJ);
34 -
         title('10log10(J)_2');
```



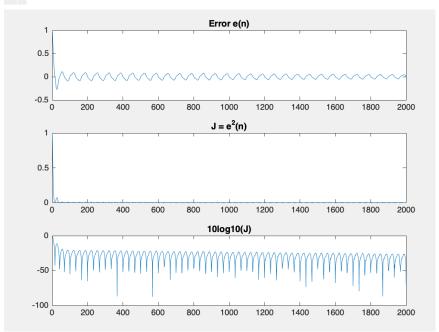
G2 as D2

```
1 -
2 -
       M = 3; % number of filter coefficients
       x = cos(0.03*pi*(0:N-1)'); % input signal
3 -
4 -
       x = awgn(x, 40, 'measured');
 5 -
       W=[0,0,0,0];
6 -
        s=[1 -2 4];
7 -
       mu=0.01;
8
        a=1;
9 -
        d=filter(s,a,x);
        rls = dsp.RLSFilter(11, 'ForgettingFactor', 0.98);
10 -
11 -
        [y,e] = rls(x,d);
12
      \Box for n = M:N
13 -
            x_{in} = cos(0.03*pi*n); % get the input vector
14 -
15 -
            y_est = w(n)' * x_in; % estimate the output
16 -
            e(n) = d(n) - y_est;
            mu = 1 / (x in' * x in);
17 -
18 -
            w(n+1) = w(n) + 2*mu*e(n)*x_in;
19 -
            J(n) = e(n)^2;
20 -
       end
21
       % Plot the learning curves
22 -
       figure;
23 -
        subplot(3,1,1);
24 -
        plot(err);
25 -
       title('Error e(n)');
26
27 -
        subplot(3,1,2);
28 -
        J = err.^2;
        plot(J);
29 -
30 -
       title('J = e^2(n)');
31
32 -
        subplot(3,1,3);
33 -
        logJ = 10*log10(J);
34 -
        plot(logJ);
35 -
        title('10log10(J)');
```



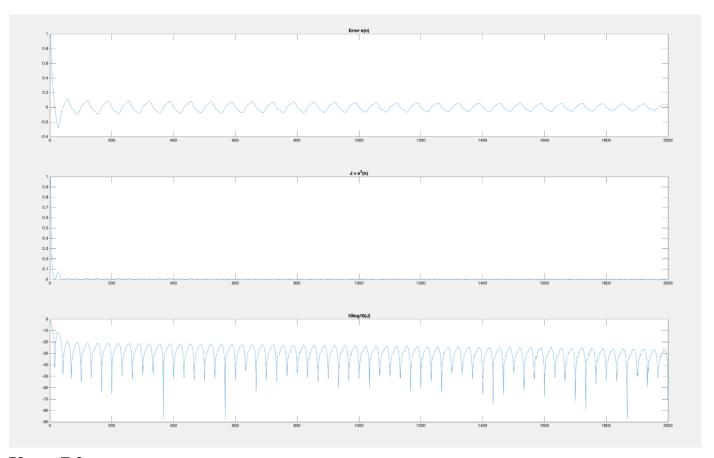
G2 as F2

```
1 -
2 -
       M = 3; % number of filter coefficients
3 -
       x = cos(0.03*pi*(0:N-1)'); % input signal
       x = awgn(x, 40, 'measured');
 4 -
 5 -
       W=[0,0,0,0];
 6 -
       s=[1 -2 4];
 7 -
       mu=0.001;
 8 -
       a=1;
 9 -
       d=filter(s,a,x);
       rls = dsp.RLSFilter(11, 'ForgettingFactor', 0.98);
10 -
11 -
       [y,e] = rls(x,d);
12
13 -
     \neg for n = M:N
14 -
           x_{in} = cos(0.03*pi*n); % get the input vector
15 -
            y_{est} = w(n)' * x_{in}; % estimate the output
           e(n) = d(n) - y_est;
16 -
17 -
            mu = 1 / (x_in' * x_in);
            w(n+1) = w(n) + 2*mu*e(n)*x_in;
18 -
19 -
            J(n) = e(n)^2;
20 -
21
       % Plot the learning curves
22 -
       figure;
       subplot(3,1,1);
23 -
24 -
       plot(err);
25 -
       title('Error e(n)');
26
27 -
       subplot(3,1,2);
28 -
       J = err.^2;
29 -
       plot(J);
30 -
       title('J = e^2(n)');
31
32 -
       subplot(3,1,3);
33 -
       logJ = 10*log10(J);
34 -
       plot(logJ);
35 -
       title('10log10(J)');
```



```
1 -
       N=2000;
2 -
       M = 3; % number of filter coefficients
       x = cos(0.03*pi*(0:N-1)'); % input signal
3 -
       x = awgn(x, 30, 'measured');
4 -
       W=[0,0,0,0];
5 -
6 -
       s=[1 -2 4];
7 -
       mu=0.01;
       a=1;
8 -
9 -
       d=filter(s,a,x);
       rls = dsp.RLSFilter(11, 'ForgettingFactor', 0.98);
10 -
11
       %[y,e] = rls(x,d);
12
     \neg for n = M:N
13 -
14 -
            x_{in} = cos(0.03*pi*n); % get the input vector
15 -
            y_est = w(n)' * x_in; % estimate the output
16 -
            e(n) = d(n) - y_{est};
            mu = 1 / (x_in' * x_in);
17 -
            w(n+1) = w(n) + 2*mu*e(n)*x_in;
18 -
            J(n) = e(n)^2;
19 -
20 -
       end
       % Plot the learning curves
21
22 -
       figure;
23 -
       subplot(3,1,1);
       plot(err);
24 -
25 -
       title('Error e(n)');
26
27 -
       subplot(3,1,2);
28 -
       J = err.^2;
29 -
       plot(J);
       title('J = e^2(n)');
30 -
31
32 -
       subplot(3,1,3);
33 -
       logJ = 10*log10(J);
34 -
       plot(logJ);
       title('10log10(J)');
35 -
```

```
1 -
        N=2000;
2 -
3 -
4 -
        M = 3; % number of filter coefficients
        x = cos(0.03*pi*(0:N-1)'); % input signal
x = awgn(x, 30, 'measured');
5 -
        w=[0,0,0,0];
6 -
7 -
         s=[1 -2 4];
        mu=0.001;
8 -
         a=1;
9 -
        d=filter(s,a,x);
10 -
         rls = dsp.RLSFilter(11, 'ForgettingFactor', 0.98);
11
        [y,e] = rls(x,d);
12
13 -
      \neg for n = M:N
             x_in = cos(0.03*pi*n); % get the input vector
y_est = w(n)' * x_in; % estimate the output
14 -
15 -
             e(n) = d(n) - y_est;
mu = 1 / (x_in' * x_in);
16 -
17 -
18 -
             w(n+1) = w(n) + 2*mu*e(n)*x_in;
19 -
             J(n) = e(n)^2;
20 -
       ^{\mathsf{L}} end
21
        % Plot the learning curves
22 -
        figure;
        subplot(3,1,1);
23 -
24 -
        plot(err);
25 -
         title('Error e(n)');
26
27 -
        subplot(3,1,2);
        J = err.^2;
28 -
29 -
        plot(J);
30 -
         title('J = e^2(n)');
31
         subplot(3,1,3);
32 -
33 -
34 -
        logJ = 10*log10(J);
         plot(logJ);
35 -
         title('10log10(J)');
```



```
1 -
       trials = 1000;
2 -
       J_total = zeros(1, N);
3
4 -

□ for i = 1:trials

5 -
            x = cos(0.03*pi*(0:N-1)');
6 -
            x = awgn(x, 40, 'measured');
7 -
            W=[0,0,0,0];
8 -
            s=[1 -2 4];
9 -
            mu=0.01;
10 -
            a=1:
11 -
            d=filter(s,a,x);
            rls = dsp.RLSFilter(11, 'ForgettingFactor', 0.98);
12 -
13 -
            [y,e] = rls(x,d);
14
15 -
            for n = M:N
16 -
                x_{in} = cos(0.03*pi*n);
17 -
                y_{est} = w(n)' * x_{in};
                e(n) = d(n) - y_est;
18 -
                mu = 1 / (x_i^* * x_i^*);
19 -
                w(n+1) = w(n) + 2*mu*e(n)*x_in;
20 -
21 -
                J(n) = e(n)^2;
22 -
            end
23 -
             if i == 1
24 -
                J_total = J;
25 -
            else
26 -
                J_total = J_total + J;
27 -
             end
28 -
       end
29
       % Average J over number of trials
30
31 -
       J_avg = J_total / trials;
32
33
       % Plot the averaged J
34 -
        figure;
35 -
        logJ = 10*log10(J_avg);
36 -
        plot(logJ);
37 -
        xlabel('Iteration Steps');
38 -
        ylabel('10log10(J)');
39 -
        title('Averaged J (10log10(J) vs iteration steps)');
                       Averaged J (10log10(J) vs iteration steps)
   350
   300
   250
   200
 10log10(J)
    150
   100
    50
     0
```

-50

200

400

600

1000

Iteration Steps

800

1200

1400

1600

1800

2000

```
1 -
       N=2000;
2 - 3 -
       M = 3;
       num_trials = 1000;
4 -
       J_avg = zeros(N-M+1, 1);
5 -

¬ for trial = 1:num_trials

6 -
            x = cos(0.03*pi*(0:N-1)');
 7 -
            x = awgn(x, 40, 'measured');
 8 -
            W=[0,0,0,0];
 9 -
            s=[1 -2 4];
10 -
            mu=0.001;
11 -
            a=1;
12 -
            d=filter(s,a,x);
13 -
            rls = dsp.RLSFilter(11, 'ForgettingFactor', 0.98);
14 -
           [y,e] = rls(x,d);
15 -
            J = zeros(N-M+1, 1); % initialize J for this trial
16 -
            for n = M:N
17 -
                x_{in} = cos(0.03*pi*n);
18 -
                y_{est} = w(n)' * x_{in};
19 -
                e(n) = d(n) - y_est;
20 -
                mu = 1 / (x_in' * x_in);
21 -
                w(n+1) = w(n) + 2*mu*e(n)*x_in;
                J(n-M+1) = e(n)^2;
22 -
23 -
            end
24 -
            J_avg = J_avg + J;
25 -
26 -
       J_avg = J_avg / num_trials;
27
       % Plot the averaged J
28
29 -
       figure;
30 -
       logJ = 10*log10(J_avg);
31 -
       plot(logJ);
32 -
       title('10log10(J) vs iteration steps');
33 -
       xlabel('Iteration steps');
34 -
       ylabel('10log10(J)');
```

