Table of Contents

```
clear;
clc;
close all;
```

Mohammad Javad Amin 401211193 Problem 1 , exercise 3

definition

d : desired signal N :length of filter M : length of input signal alpha : learning rate e : errors w : weights of filter p : power of input signal l : noise amplitude d_t : corrupted desired signal

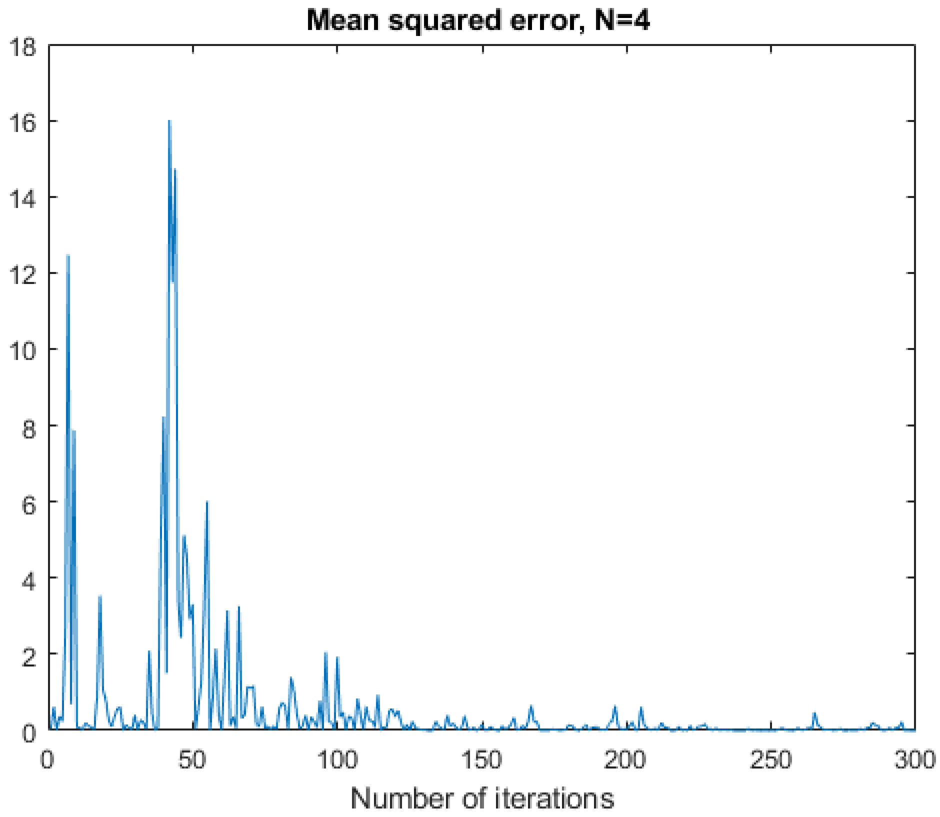
```
a=[1,0.5];
b=[1,-0.9];         % impulse response
inputs=randn(1,300);
d=filter(b,a,inputs);
M=length(inputs);
```

part a

```
N = 4;
k=5;
m_error=zeros(1,M);

% calulate mu max for N=4
p= inputs*inputs'/M;
 alpha_max=2/(2*N*p);
 disp('mu max for N=4 and is :');
 disp(alpha_max);
 alpha_int = alpha_max*ones(1,N);

for i=1:k
    [w,cost]=VSLMS(inputs,d,N,alpha_int,M,alpha_max);
    m_error=m_error+cost;
end
m_error=m_error/5;

disp("weights for N=4 :");
disp(w');


figure
plot(m_error);
```

```
title('Mean squared error, N=4 ');
xlabel('Number of iterations');
```

```
mu max for N=4 and is :
    0.2569
```

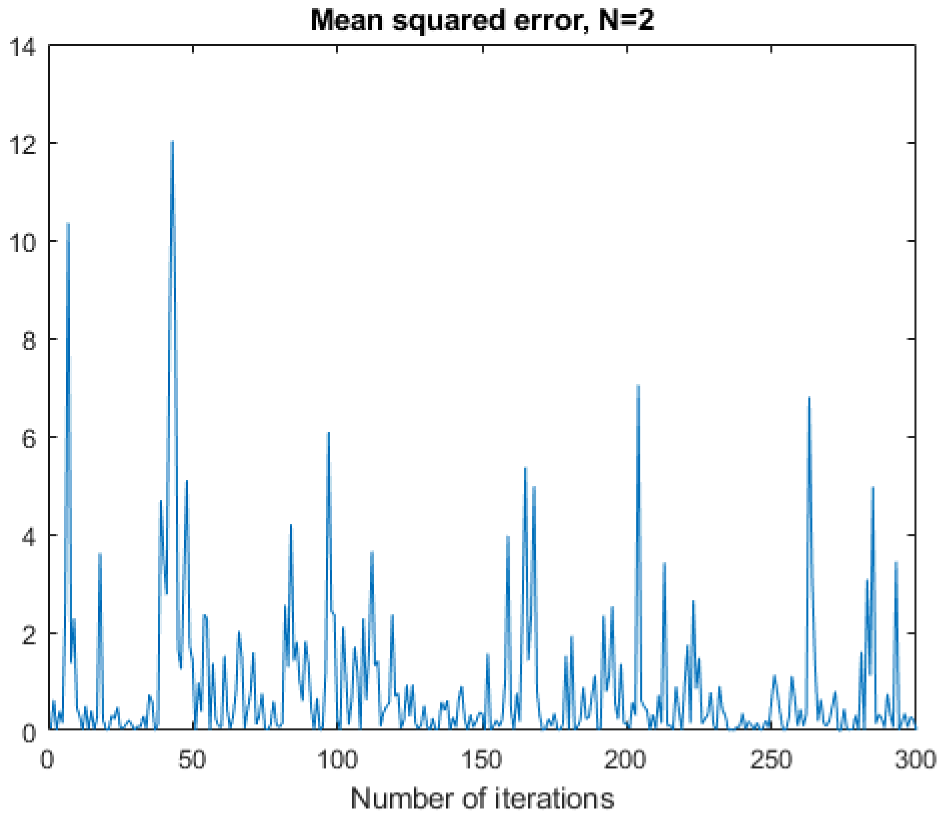


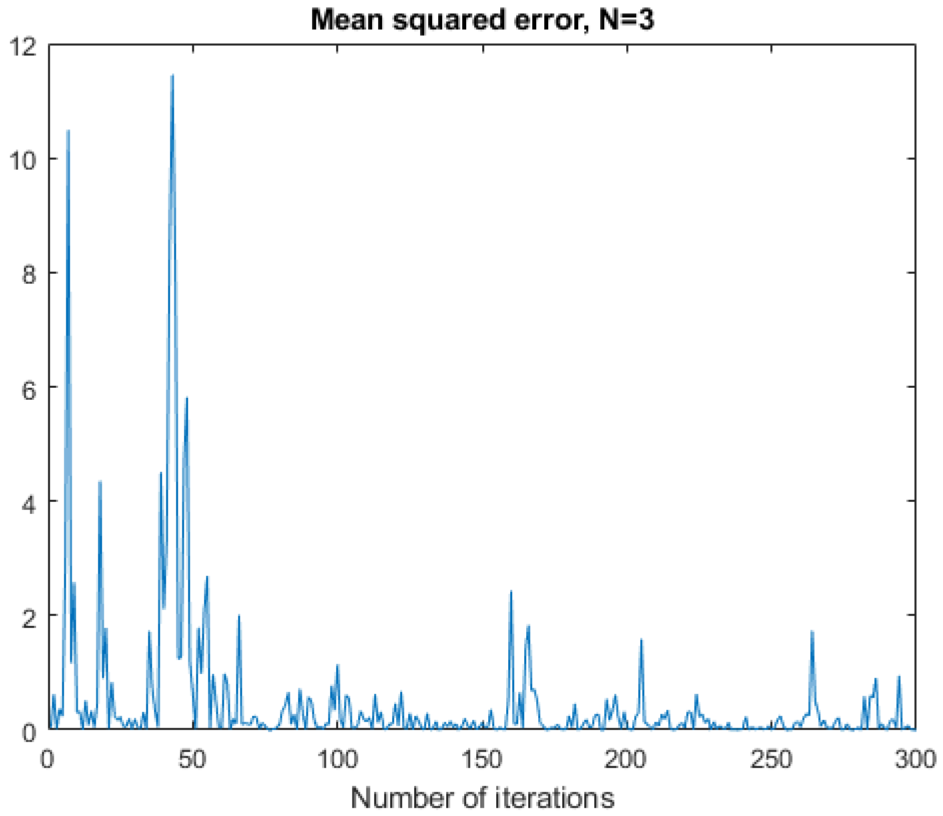
part b

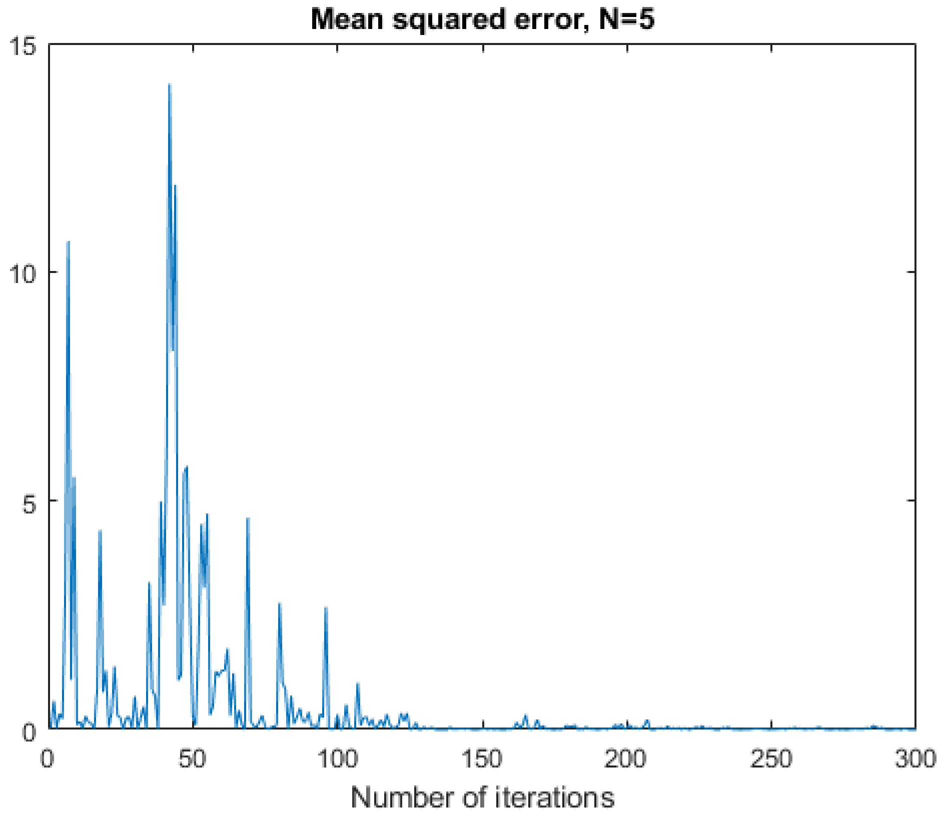
```
N = [2,3,5,7,10];
for i=N
    alpha_max=2/(3*i*p);
    disp(['mu max for N=',num2str(i)]);
    disp(alpha_max);

    m_error=zeros(1,M);
    alpha_int = alpha_max*ones(1,i);

    for g=1:k
        [w,cost]=VSLMS(inputs,d,i,alpha_int,M,alpha_max);
        m_error=m_error+cost;
    end
    m_error=m_error/5;

    disp(['weights forand N=',num2str(i),':']);
    disp(w');
    figure
    plot(m_error);
    title(['Mean squared error, N=',num2str(i)]);
    xlabel('Number of iterations');
end
```

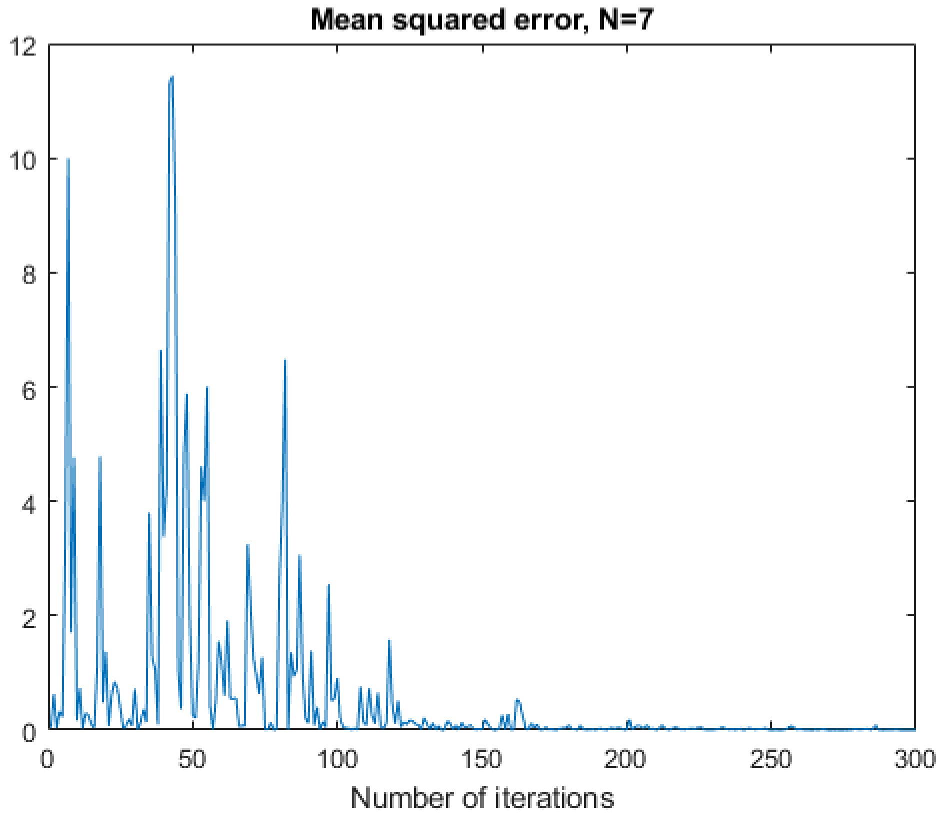
```
mu max for N=2
    0.3426

weights forand N=2:
    1.1586
   -1.4484

mu max for N=3
    0.2284

weights forand N=3:
    0.9770
   -1.4907
    0.7232

mu max for N=5
    0.1370

weights forand N=5:
    1.0528
   -1.3784
    0.7723
   -0.3360
    0.2132

mu max for N=7
    0.0979

weights forand N=7:
    0.9747
   -1.4083
    0.6421
   -0.3809
    0.1769
   -0.1393
    0.0343

mu max for N=10
    0.0685

weights forand N=10:
    0.9938
   -1.3925
    0.7112
   -0.3387
    0.1860
   -0.0982
```

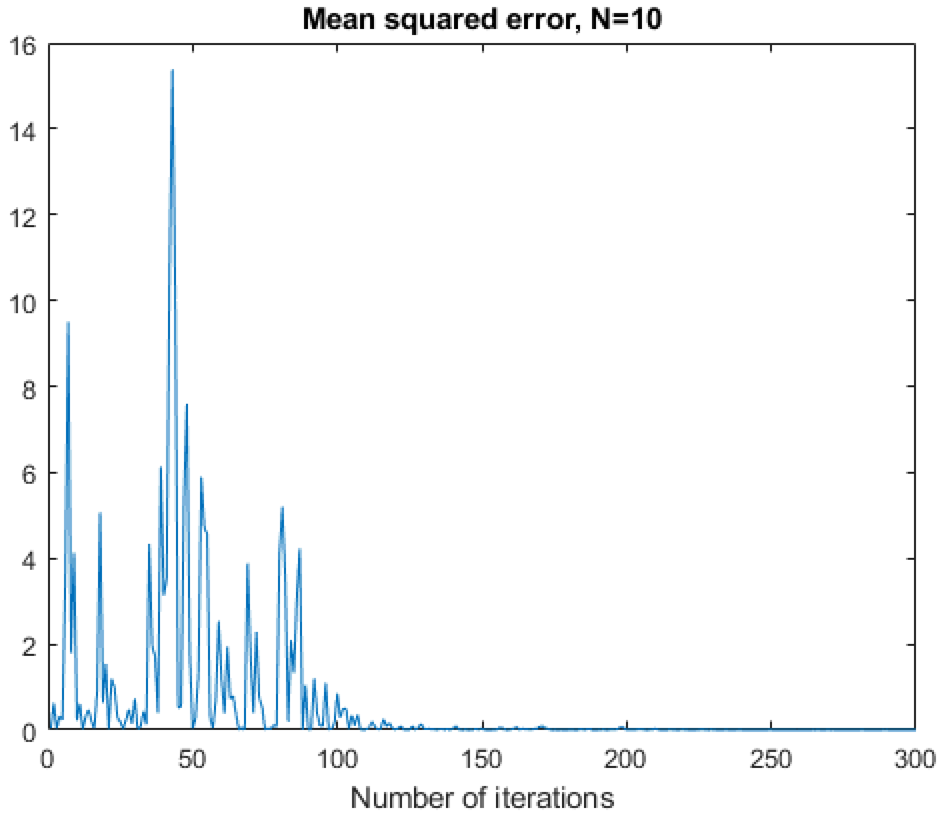
```
 0.0571
-0.0157
 0.0401
-0.0091
```




Mean squared error, N=3
12
10
8
6
4
2
0
0
50
100
150
200
250
300
Number of iterations


Mean squared error, N=5
15
10
5
0
0
50
100
150
200
250
300
Number of iterations


Mean squared error, N=7
12
10
8
6
4
2
0
0
50
100
150
200
250
300
Number of iterations


Mean squared error, N=10
16
14
12
10
8
6
4
2
0
0
50
100
150
200
250
300
Number of iterations

part c

```
l = [0.1,0.3,1];
N = 4;
v = randn(1,300);
k=5;

% calulate mu max for N=4
p= inputs*inputs'/M;
alpha_max=2/(3*N*p);
disp('mu max for N=4 and is :');
disp(alpha_max);

for g=l
    m_error=zeros(1,M);
    d_t=d+g*v;


    for i=1:k
        [w,cost]=VSLMS(inputs,d_t,N,alpha_int,M,alpha_max);
        m_error=m_error+cost;
    end
    m_error=m_error/5;

    disp(['weights for N=4 and l=',num2str(g), ': ']);
    disp(w')

    figure
    plot(m_error);
    title(['Mean squared error, N=4 and l=', num2str(g),' : ']);
    xlabel('Number of iterations');
end
disp(" The VSLMS is more quicker than LMS algorithm ")
```

```
mu max for N=4 and is :
    0.1713
```

VSLMS algorithms

```
function[w,cost,J_min,J_inf]=VSLMS(inputs,d,N,alpha,M,mu_max)
% e : error
% u_temp : because LMS run when the first sample arrive, we put M-1 zeros in
 beging of inputs, if whe don't put this zeros we must wait to m sample arrive
    u_temp=[zeros(1,N-1),inputs];
    e=zeros(1,M);
    w=zeros(1,N);
    g = ones(1,N);
    g_past = ones(1,N);
    mu_min=1e-6;
    p=5;
    alpha_past=alpha;
    for i=N:M
        u=u_temp(i:-1:i-N+1);
        y=dot(w,u);
        e(i-N+1)=d(i-N+1)-y;

        for j=1:N
            g(j)=e(i-N+1)*u(j);

            if sign(g(j))==sign(g_past(j))
                alpha(j)=p*alpha_past(j);
            else
                alpha(j)=alpha_past(j)/p;

            end

            if alpha(j)>mu_max
                alpha(j)= mu_max;
            end

            if alpha(j)<mu_min
                alpha(j)= mu_min;
            end

            w(j) =  w(j) + alpha(j)*g(j);

        end

        g_past=g;
        alpha_past=alpha;

    end
    cost=e.^2;
    J_min=min(cost);
    J_inf=sum(cost(M-19:M))/20;

end
```

```
weights for N=4 :
    0.9031
   -1.4056
    0.7621
   -0.3698
```

Published with MATLAB® R2022b