Table of Contents

.... 1
definition 1
part a 1
part b 2
part c 6
RLS algorithms 7

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

Mohammad Javad Amin 401211193 Problem 1 , exercise 3

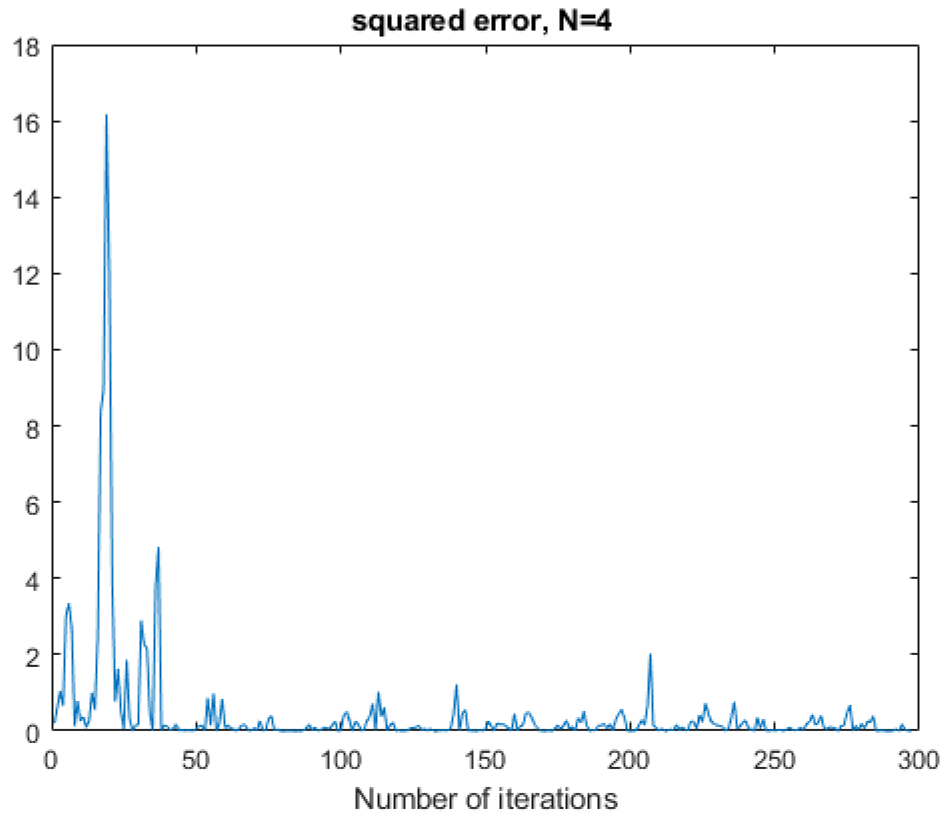
definition

d : desired signal N :length of filter M : length of input signal e : errors w : weights of filter 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-N+1);

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

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


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

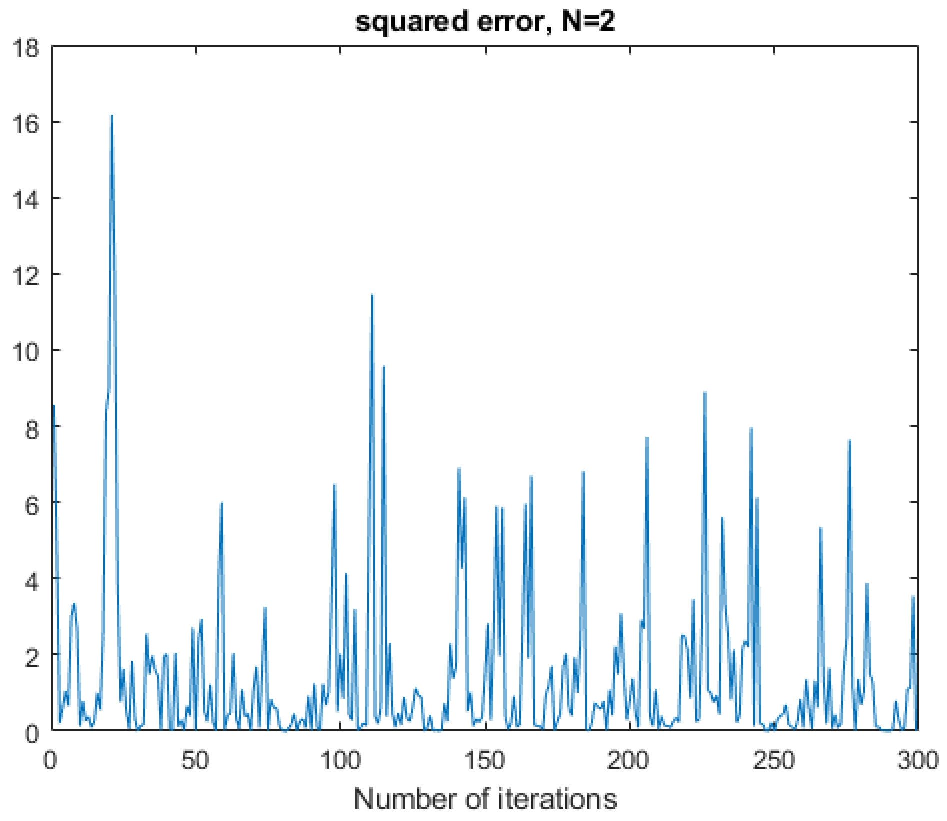


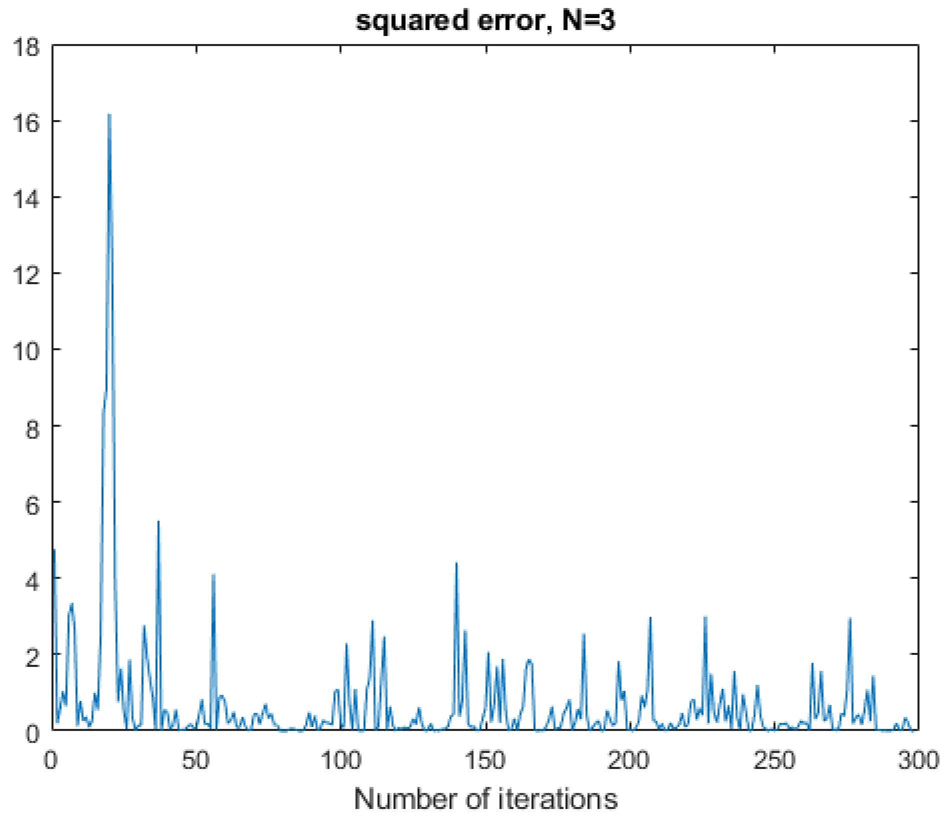
part b

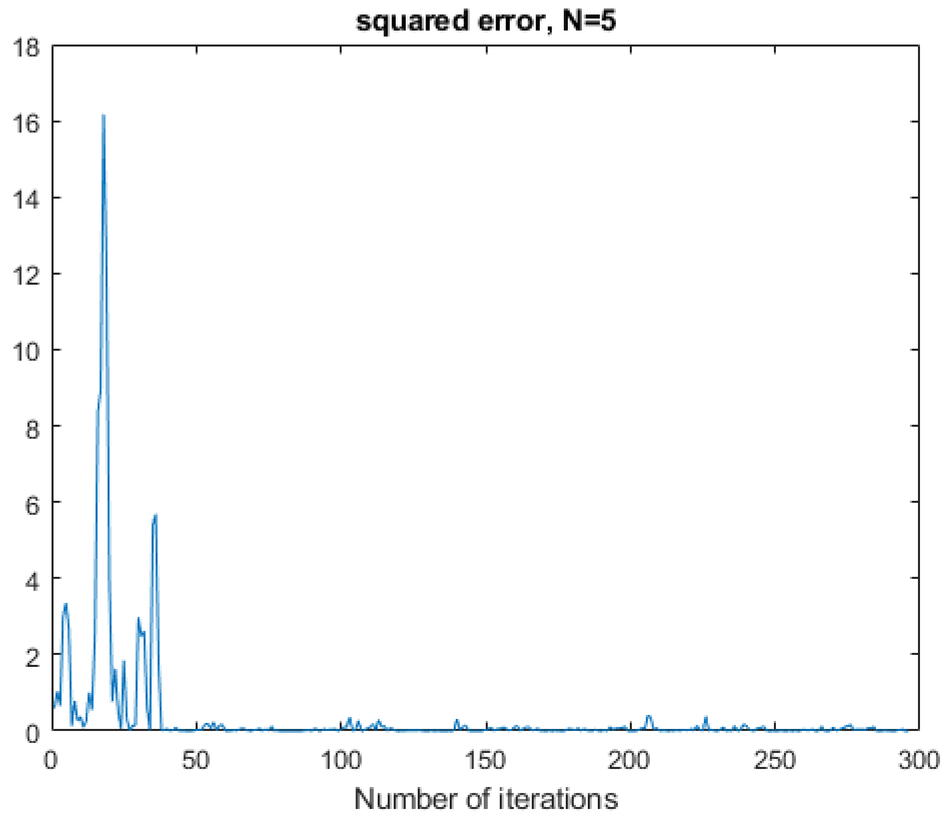
```
N = [2,3,5,7,10];
for i=N

    m_error=zeros(1,M-i+1);

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

    disp(['weights forand N=',num2str(i),':']);
    disp(w');

    figure
    plot(m_error);
    title(['squared error, N=',num2str(i)]);
    xlabel('Number of iterations');
end
```

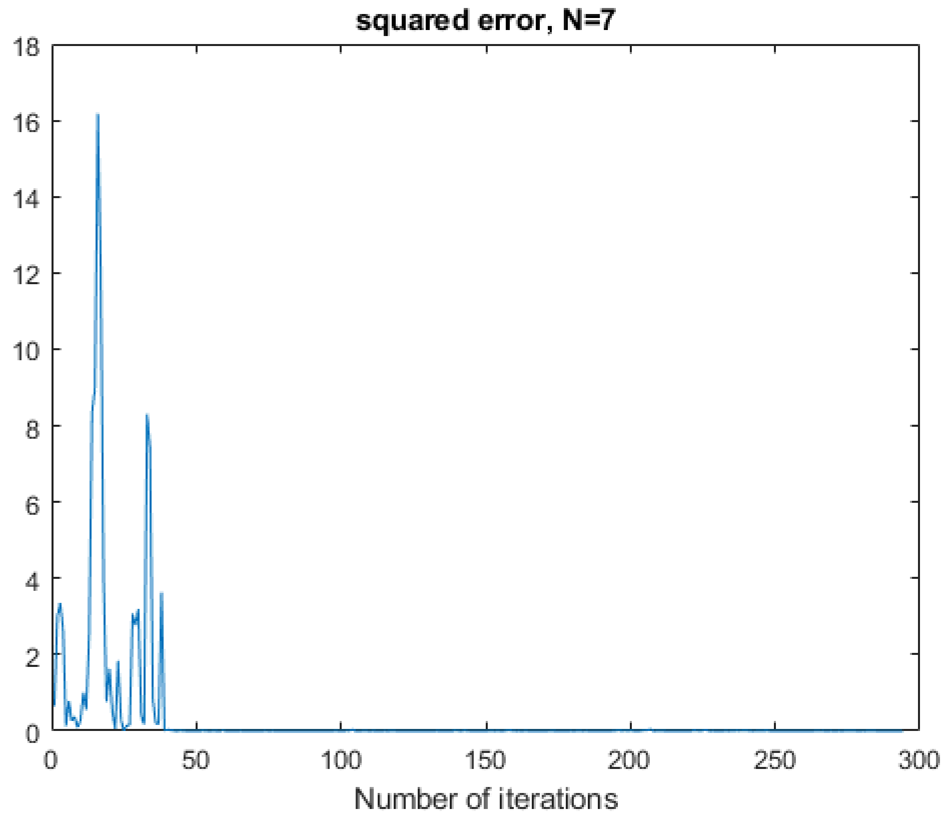
```
weights forand N=2:
    0.9487
   -1.6071
```

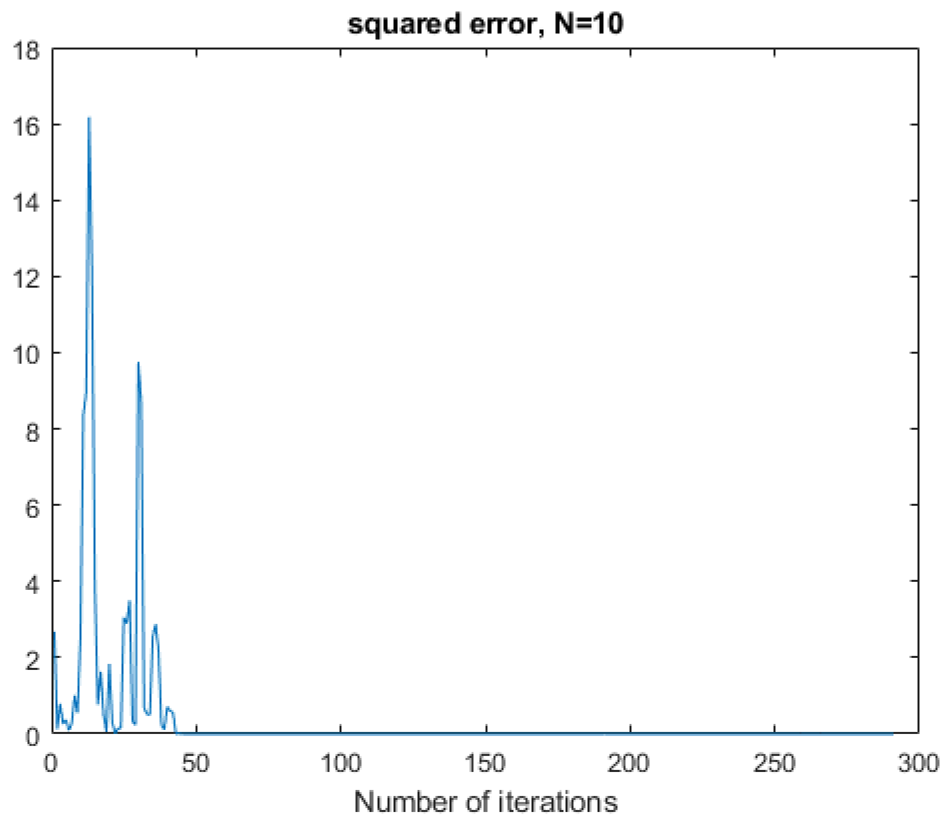
```
weights forand N=3:
    0.8786
   -1.4961
    0.8506

weights forand N=5:
    0.9654
   -1.3759
    0.7130
   -0.3568
    0.1248

weights forand N=7:
    0.9960
   -1.3895
    0.6967
   -0.3447
    0.1667
   -0.0718
    0.0233

weights forand N=10:
    1.0015
   -1.4005
    0.6999
   -0.3497
    0.1777
   -0.0895
    0.0459
   -0.0233
    0.0127
   -0.0058
```


squared error, N=2
Number of iterations


squared error, N=3
Number of iterations


squared error, N=5
Number of iterations
0
50
100
150
200
250
300
0
2
4
6
8
10
12
14
16
18


squared error, N=7
Number of iterations
0
50
100
150
200
250
300
0
2
4
6
8
10
12
14
16
18



part c

```
l = [0.1,0.3,1];
N = 4;
v = randn(1,300);
k=5;
m_error=zeros(1,M-N+1);


for g=l
    d_t=d+g*v;

    for i=1:k
        [w,cost]=RLS(inputs,d_t,N,M);
        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(['squared error, N=4 and l=', num2str(g),' : ']);
    xlabel('Number of iterations');
```

```
end
disp(" The LMS is more quicker than RLS algorithm but the error in RLS is much
 better than LMS and" + ...
    " the number of iterations the algotithm need  to converge in the RLS is
 less than LMS ");
```

RLS algorithms

```
function[w,cost,J_min,J_inf]=RLS(inputs,d,N,M)
% z : error
% N :length of filter
% M : length of input signal

    z=zeros(1,M-N+1);
    w=zeros(1,N);
    lambda=0.5;
    delta= 1e-10;

    p=delta*eye(N);

    for i=N:M-1
        u=inputs(i:-1:i-N+1);
        y=dot(w,u);
        z(i-N+1)=d(i)-y;
        k=(p*u')/(lambda+u*p*u');
        w=w+k'*conj(z(i-N+1));
        p=(p -k*conj(u)*p)/lambda;

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

end
```

```
weights for N=4 :
    0.9320
   -1.3797
    0.7734
   -0.3235
```

Published with MATLAB® R2022b