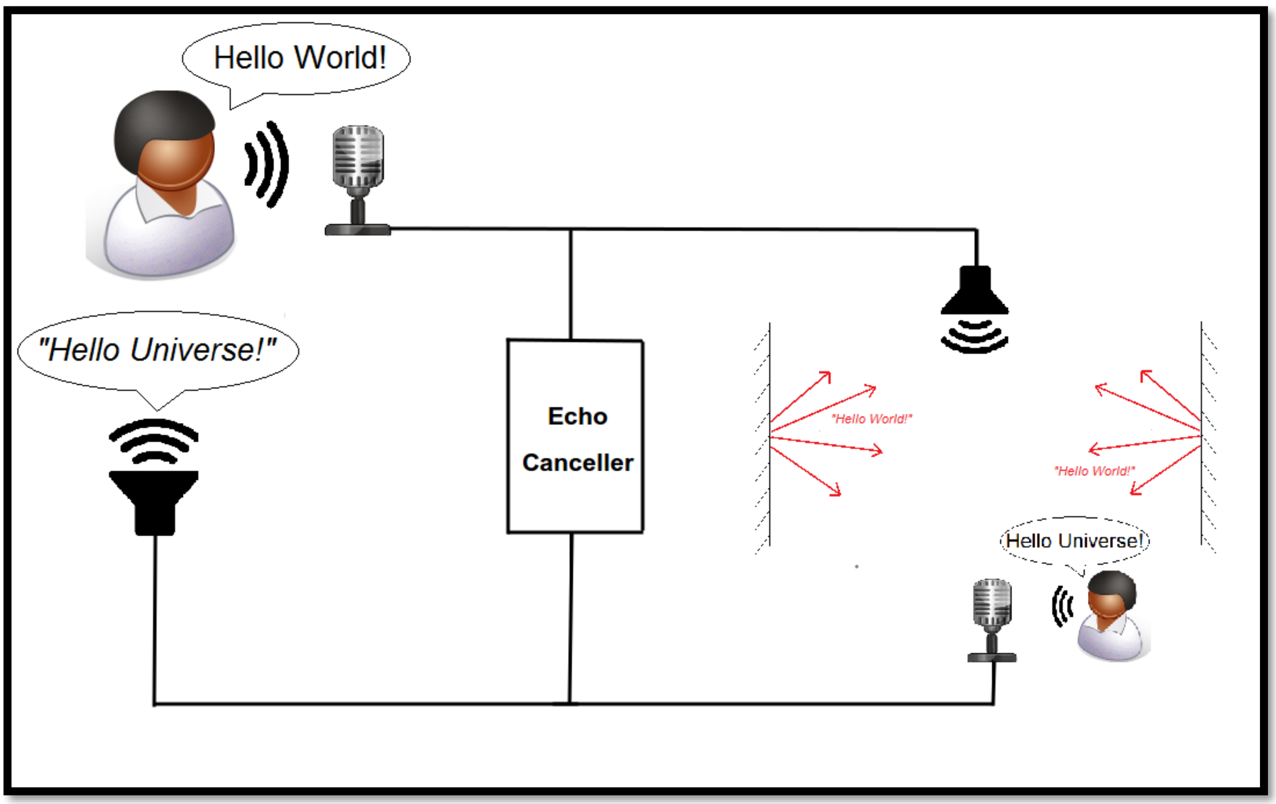
Imperial College London, EEE Department, MSc Communications & Signal Processing

Adaptive Signal Processing Assignment

Adaptive Algorithms for Echo Cancellation



Supervisor of Experiment: Prof. D. P. Mandic

Demonstrator: Takashi Nakamura

Student: Ilias Chrysovergis, ic517@ic.ac.uk, 01449042

1: Least Mean Square (LMS) adaptation algorithm implementation

The matlab routine for the LMS algorithm is given below:

```
function [e, wmat] = LMS(x, d, mu, L)
N = length(x);
wmat = zeros (N, L);
e = zeros(N,1);
for i = L:N
    e(i) = d(i) - wmat(i-1,:)*x(i:-1:(i-L+1));
    wmat(i,:) = wmat(i-1,:) + 2*mu*x(i:-1:(i-L+1))'*e(i);
end
end
```

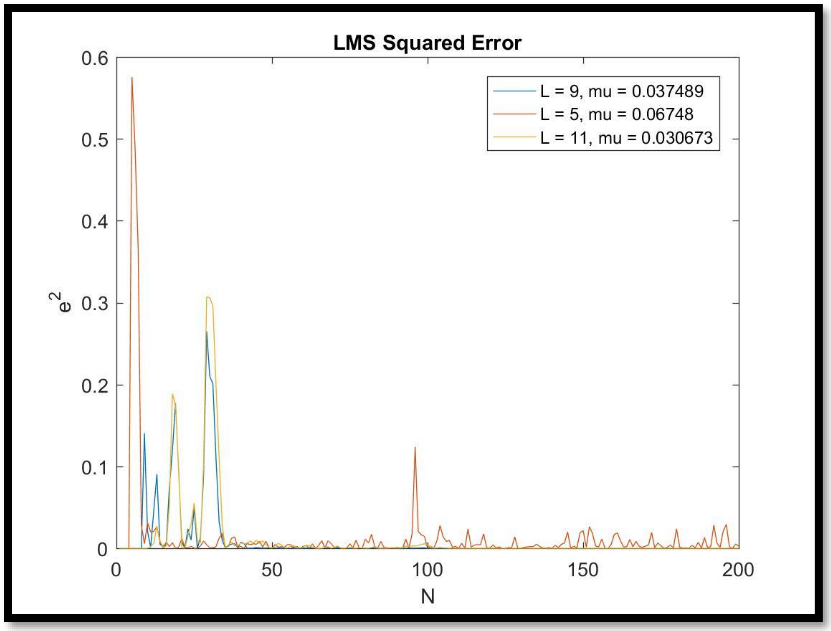
The matlab routine for the Normalized LMS algorithm is given below:

```
function [e, wmat] = NLMS(x, d, mu, L )
N = length(x);
wmat = zeros (N, L);
e = zeros(N,1);
for i = L:N
    e(i) = d(i) - wmat(i-1,:)*x(i:-1:(i-L+1));
    dnm = 1 + norm(x(i:-1:(i-L+1)));
    wmat(i,:) = wmat(i-1,:) + 2*mu/dnm*x(i:-1:(i-L+1))'*e(i);
end
end
```

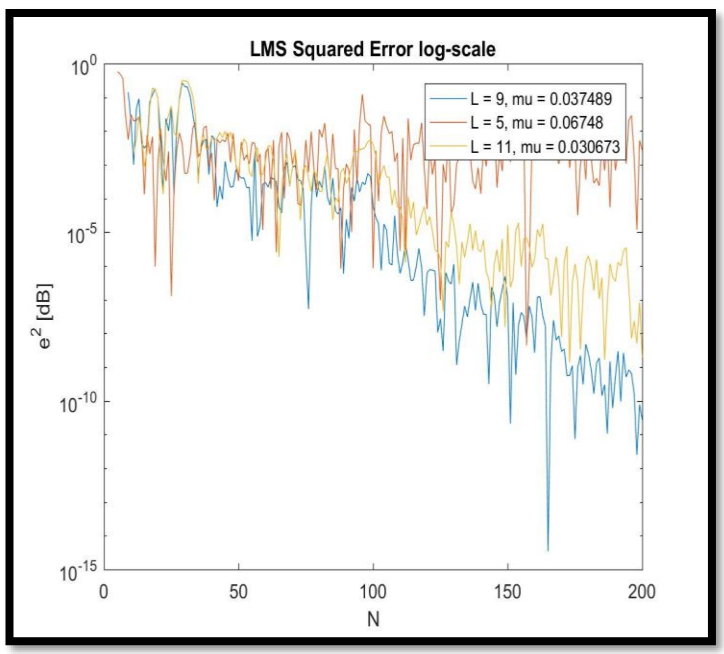
By comparing these two routines one can easily realize that the NLMS algorithm requires more operations than the LMS algorithm.

2: Least Mean Square (LMS) adaptation algorithm performance

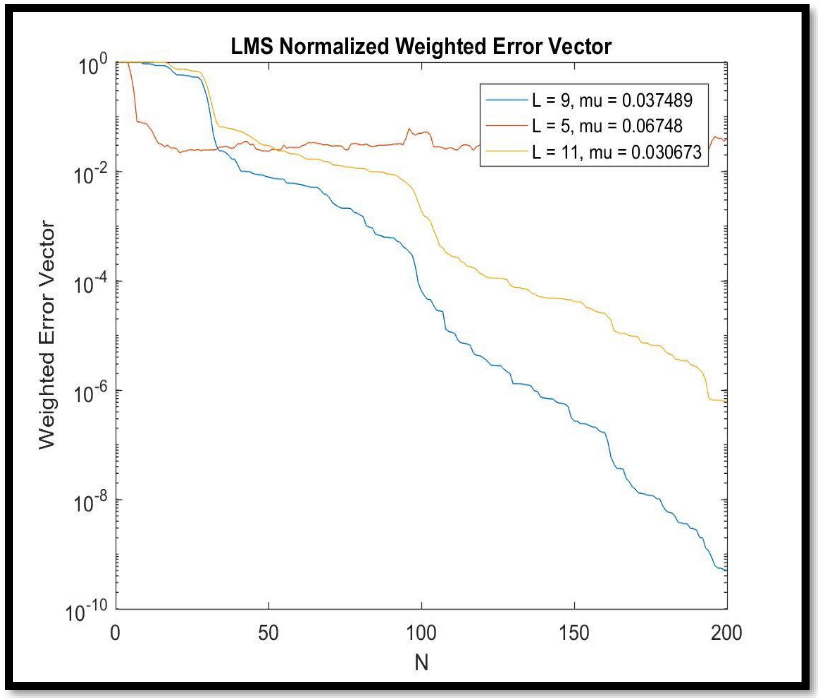
i. The squared error for 3 different lengths of the adaptive filter is depicted. It is easy for someone to realize that as L is increasing, the adaptation gain is decreasing since those two variables are inversely proportional. In this first figure, the plot() function is used and it is obvious that the information for the squared error is not neatly presented.



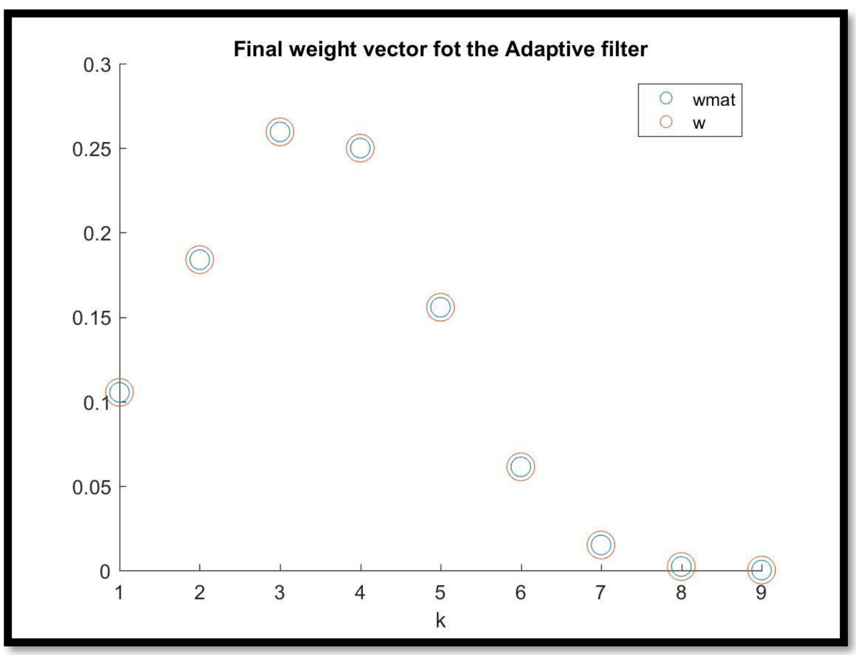
For that reason, the semilogy() function, that plots the data with logarithmic scale for y-axis, is used. The diagram below presents the squared error in logarithmic scale. Now, the difference between the three adaptive filters is obvious as the information is better presented. It can be realized that the best results are achieved when the length of the adaptive filter is equal to the length of the unknown FIR filter. Furthermore, it seems that it is better to overestimate the length of the filter than to underestimate it since the squared error is bigger in the second case. Finally, it is clear that the squared error is descending according to the sample number for L = 9 and L = 11, while the underestimated adaptive filter remains stationary. This descent is the meaning of the adaptive filters and is so important for estimating the desired signals by eliminating the effects of the channel and the noise. In the following figure the squared error in log-scale vs the sample number is depicted.



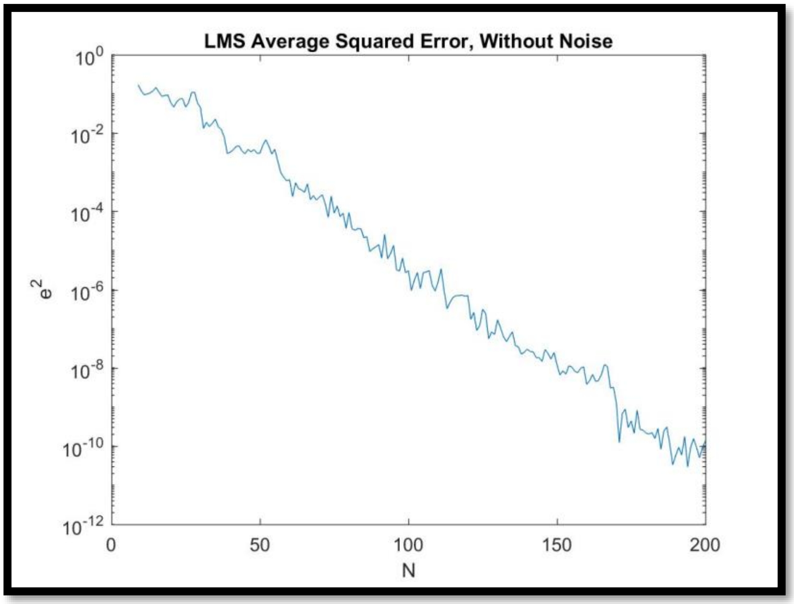
This figure presents the normalized weight error vector norm vs the sample number. This metric gives the difference between the coefficient vector of the adaptive filter and the impulse response sequence of the unknown system. Since there are $N = 200$ sample numbers there are also 200 different estimations of the unknown system. This metric is decreasing too since each estimation of the adaptive filter is better from its' previous one and gives a better approximation of the impulse response of the unknown system. The comments on the performance according to the different lengths is equivalent to the one of the previous figure. The smallest weighted error is achieved when the length of the adaptive filter is equal to the length of the unknown system. Additionally, overestimation of the order of the filter is better than underestimation. Furthermore, the fact that the error for $L = 5$ is not decreasing and remains stationary can be seen here too.

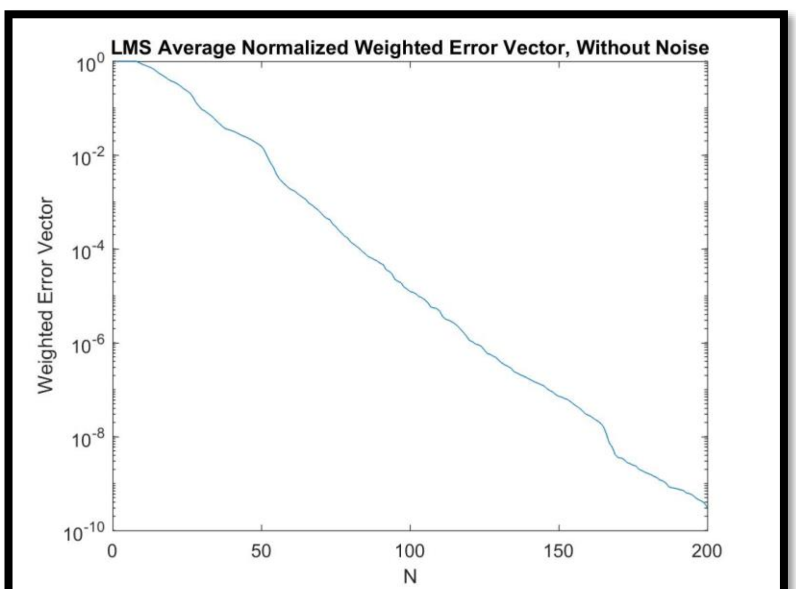


The final weight vector is depicted here. It is clear that the LMS algorithm estimates very accurately the unknown system's coefficients.

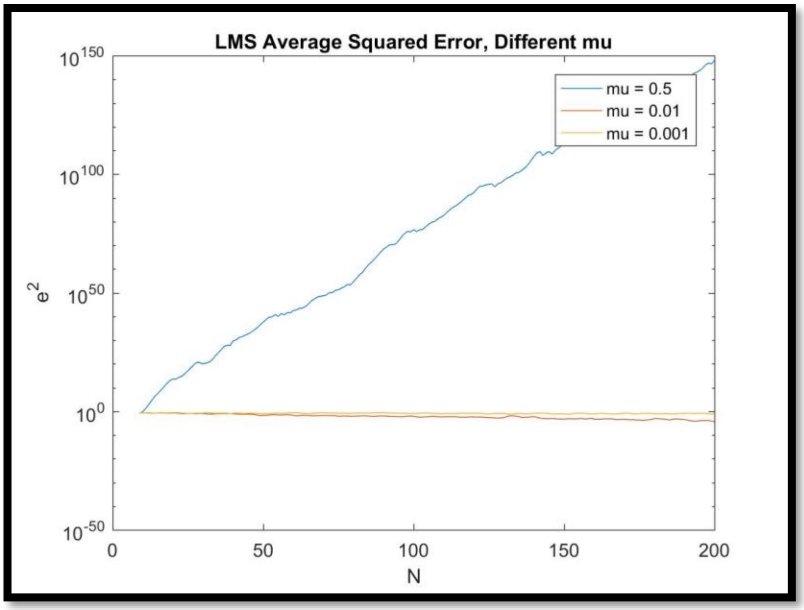


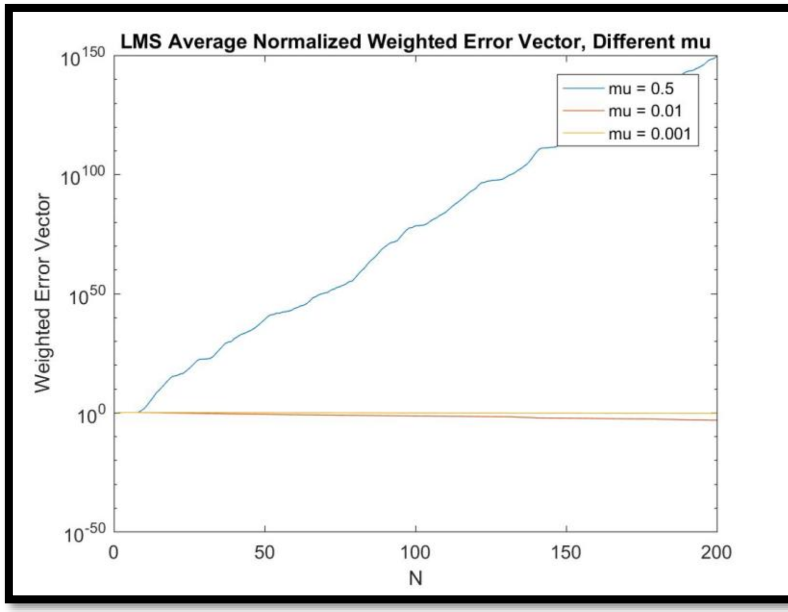
ii. The average squared error and the average weight error vector norms for 20 independent realizations of the input sequences are presented in the figures below. The realization of 20 independent input sequences corresponds to an approximate ensemble average and gives us a better estimation of how the adaptive filter performs, since it is less depended on the random samples of the input sequence. As happened in the previous diagrams, it is obvious here that the squared error as well as the weight error vector are decreasing. It can be seen that now -after the averaging- the fluctuations of the lines are smaller, and one may consider that if more independent realizations are created, the two diagrams will become straighter lines. It is also important to understand that the error is decreasing linearly with the sample number. Therefore, it is clear now that the averaging gives better results than just taking one input sequence and plotting the errors, as happened in the previous question. Th two diagrams are depicted below:



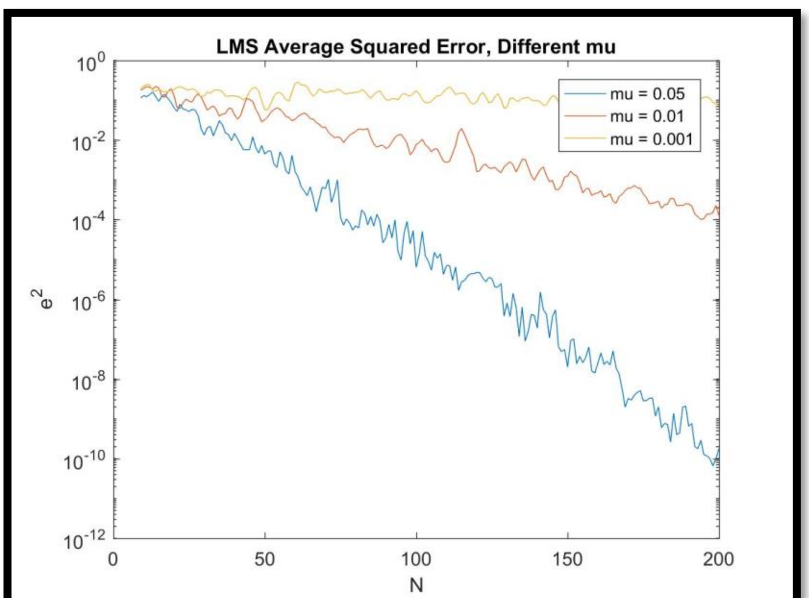


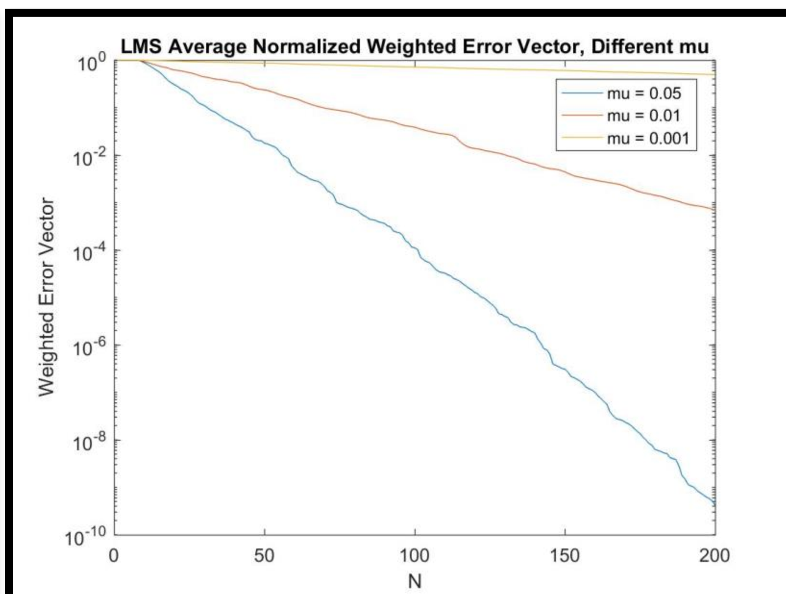
iii. In this question, different values are given for the adaptation gain μ. When $\mu > \frac{1}{L*P_x}$ the adaptive filter does not converge, and the error is always increasing. This fact can be seen in the following two figures, where $\mu = 0.5 > \frac{1}{L*P_x} = 0.11$.



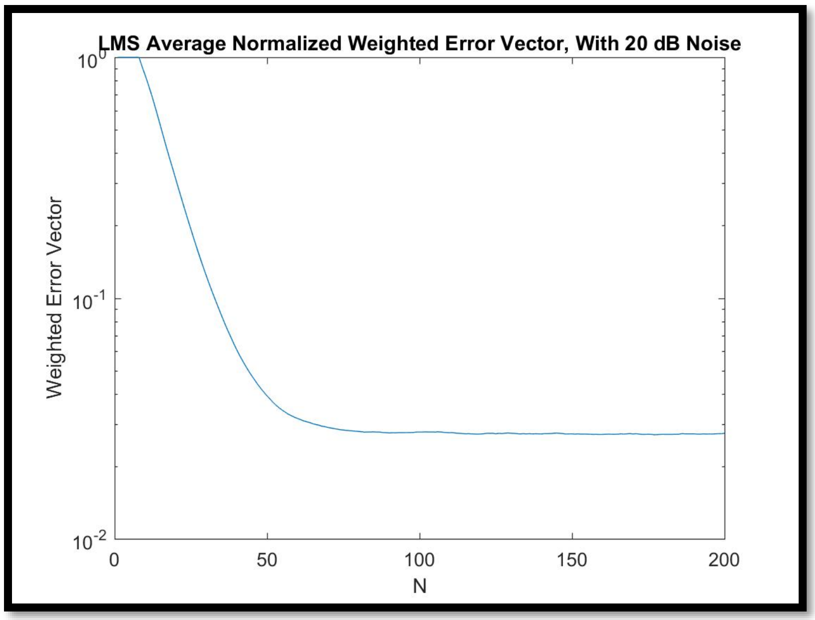


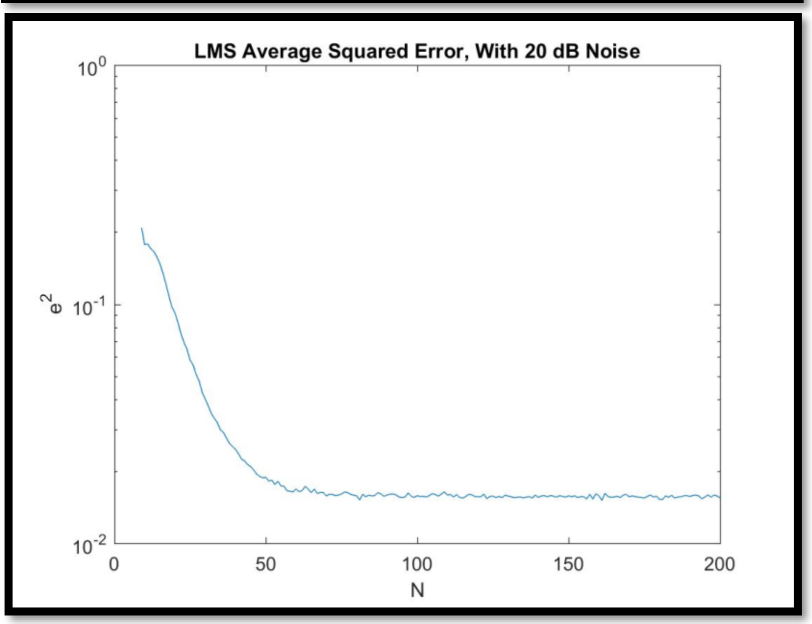
If we choose a different value for the adaptation gain in order to create a stable filter we can have some useful insights with the two diagrams. It can be seen in the two figures below that as we decrease the adaptation gain the error is increasing. Therefore, the choice of the adaptation gain is a very critical task and the performance of our filter depends on it.



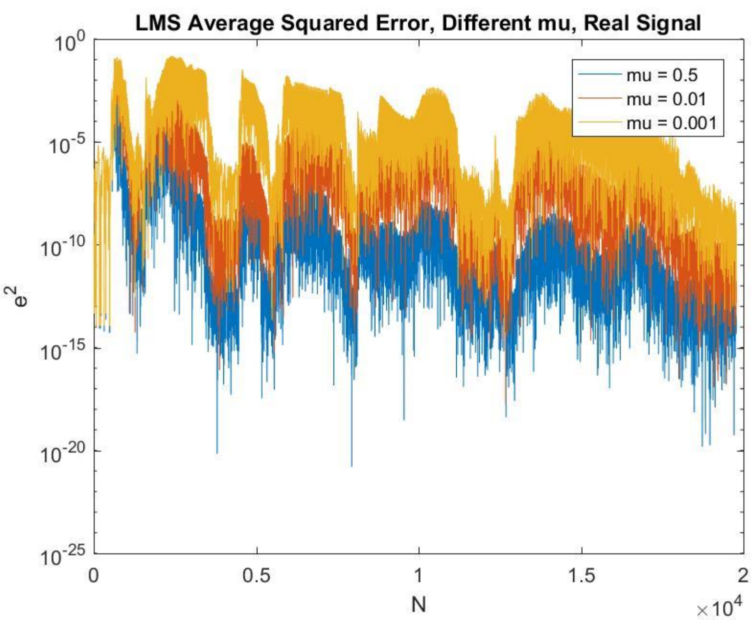


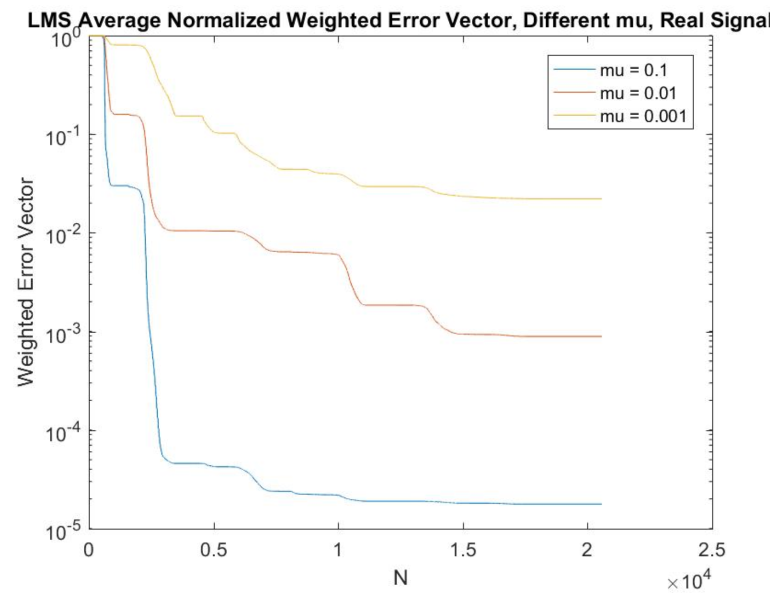
iv. In this question additive white gaussian noise has been added with an SNR equal to 20. That means that the amplitude of the unknown system is 10 times bigger than that of the independent noise. Furthermore, 10000 independent input sequences have been realized in order to have more smooth graphs. It is clear that squared error as well as the normalized weighted error vector are converging to their minimums. Due to the presence of noise, after several iterations the algorithm stops decreasing. This happens because the error, after eliminating the distorted input sequence is equal to the noise of the system. Therefore, in that case the noise is the output of adaptive filter after the echo cancellation and the error cannot be further decreased.

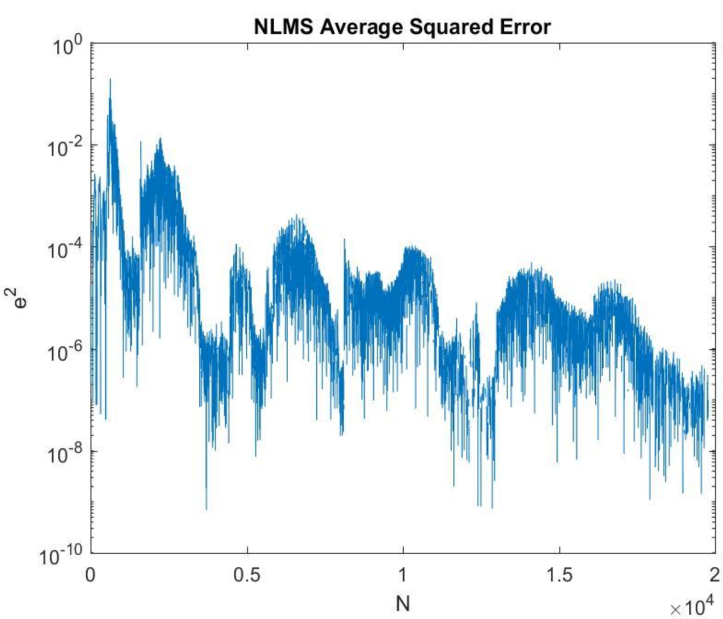


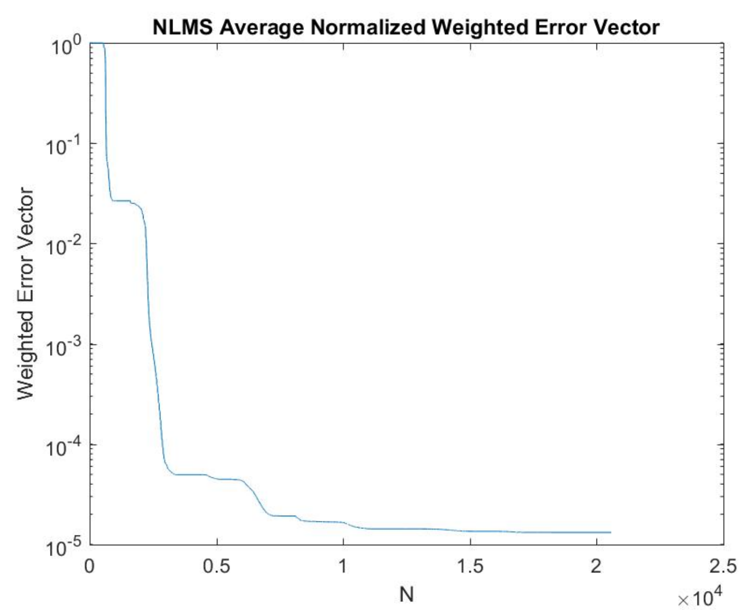


v. In this question a real signal is used as an input to the unknown system. In the following diagrams the squared error and the normalized weight error vector for LMS and NLMS algorithms are presented. For the LMS algorithm three different adaptation gains are used. For the NLMS algorithm the upper bound of the adaptation gain is calculated and then one third of it is chosen as μ to be used in the algorithm. It can be seen that LMS with $\mu = 0.1$ has the same performance with the NLMS algorithm. Furthermore, for LMS the error is increasing while the adaptation gain is decreasing as we have already seen in question iii.





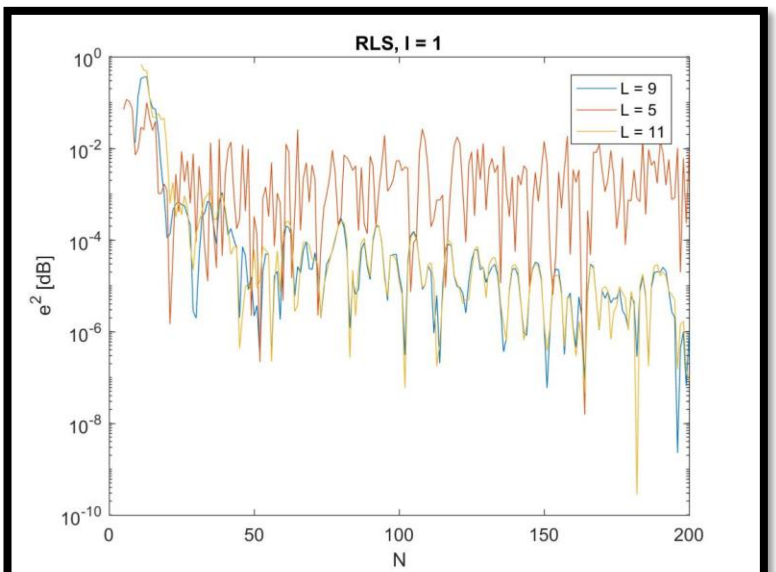


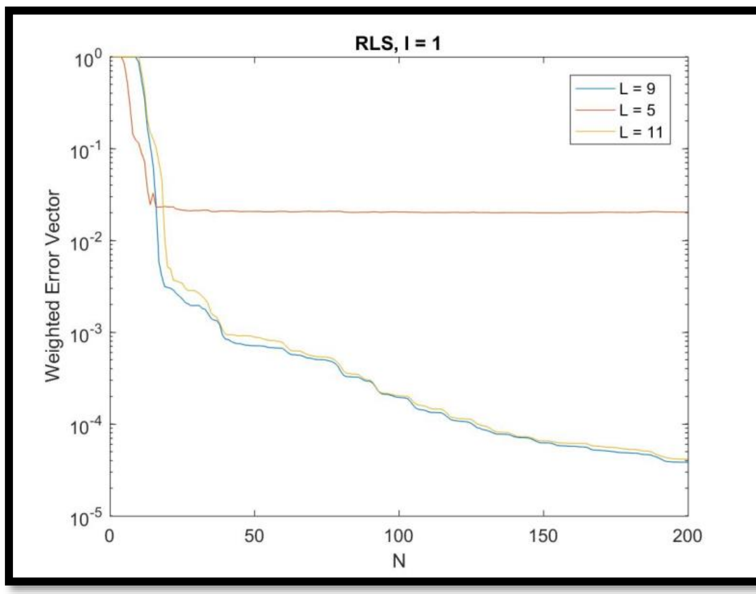


3: Recursive Least Squares (RLS) adaptation algorithm

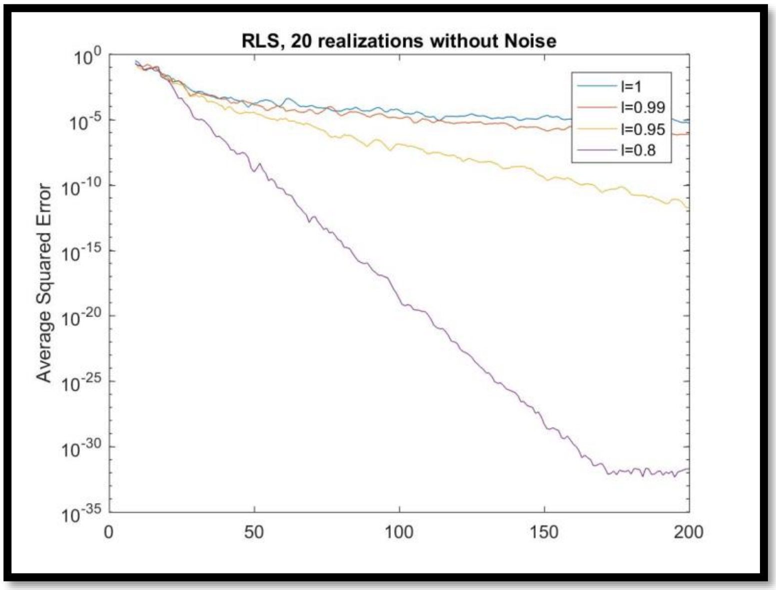
i. The matlab code implements the RLS algorithm in a tricky and clever way. Firstly, the filter coefficients and the error are initialized into zero, while the inverse of the data input matrix R is initialized as the identity matrix. Afterwards, the product of the R inverse matrix and the input sequence are stored into a variable, named z, because it is going to be used in most of the calculations. Then, the transpose of the input vector is multiplied with z in order to produce q, a value that is going to be used for the calculation of v to update the filter coefficient and the R^{-1}. Therefore, z, q, v & e are enough to update the filter coefficients and the inverse of the data input matrix. The complexity of the algorithm, in terms of number of additions and multiplications is $O(L^2)$ per iteration so totally $O(L^2 * N)$.

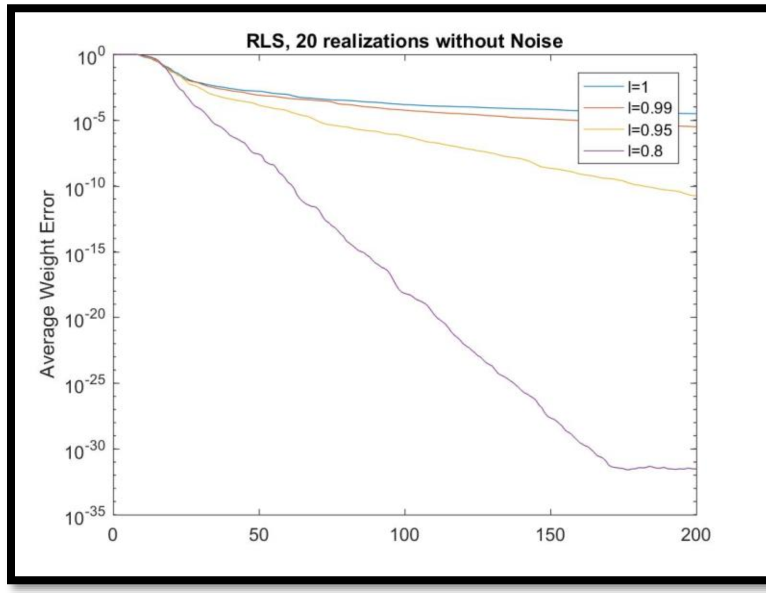
ii. In this question the performance of RLS is examined. Firstly, the squared error and the weighted error vector norms are presented, for different lengths of the adaptive filter. It is obvious that the errors for $L = 9$ & $L = 11$ are almost identical. This result shows that the RLS algorithm performs better than the LMS algorithm when we are not aware of the order of the unknown system that we are trying to approximate with the adaptive filter. Furthermore, it is obvious that this value for the forgetting factor does not give the best results, since we have found in the previous questions that the LMS algorithm can achieve smaller errors.



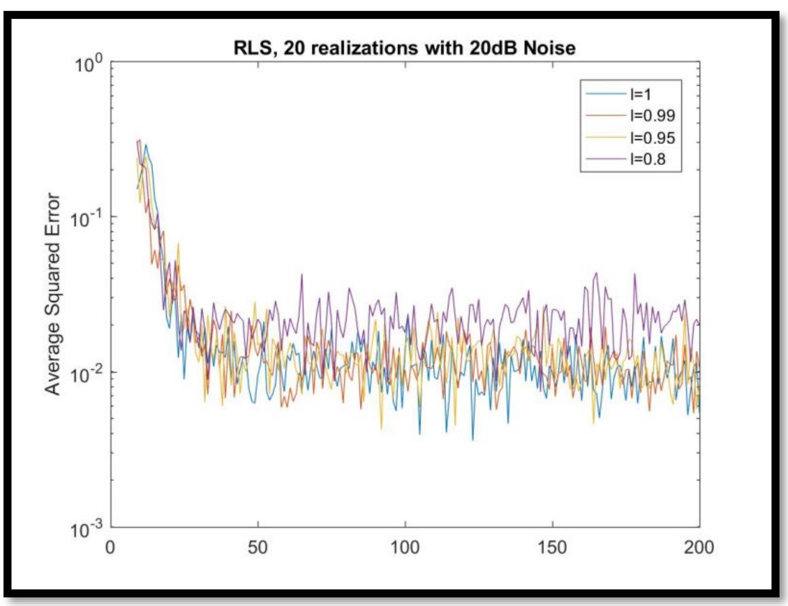


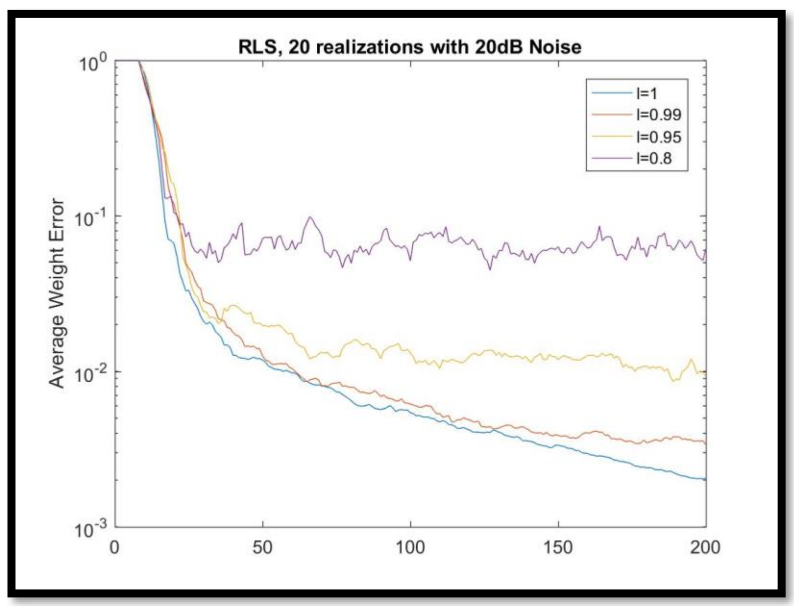
In the following diagrams 20 independent realizations of the input sequence have been implemented and the performance of the RLS algorithm is depicted. Two different sets of figures are shown, that correspond to a noiseless and a noisy system. In the second system, when white gaussian noise is added the SNR is equal to twenty. From the first set of figures, where there is no presence of noise, it is clear that while the forgetting factor is decreasing the error is decreasing too. It is obvious that when $\lambda = 0.8$ both the average squared error and the average weight error vector are almost zero. Thus, the adaptive filter is identical to the unknown system.



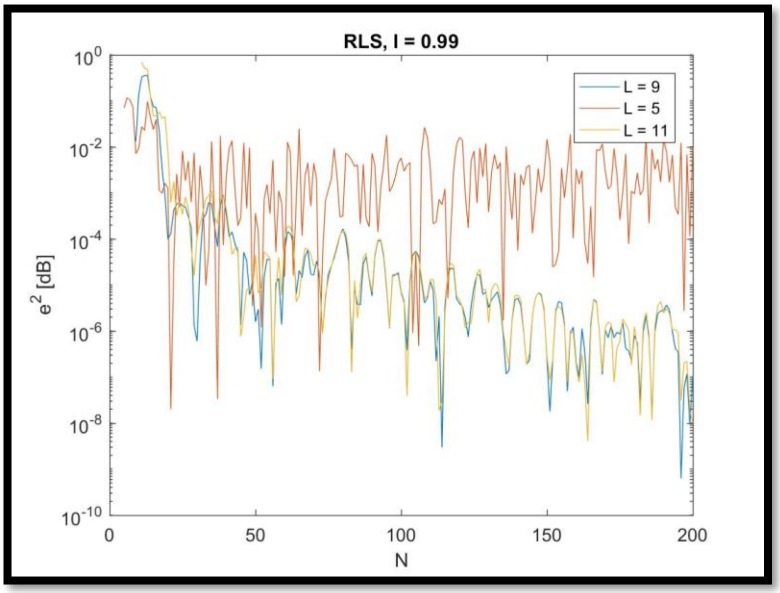


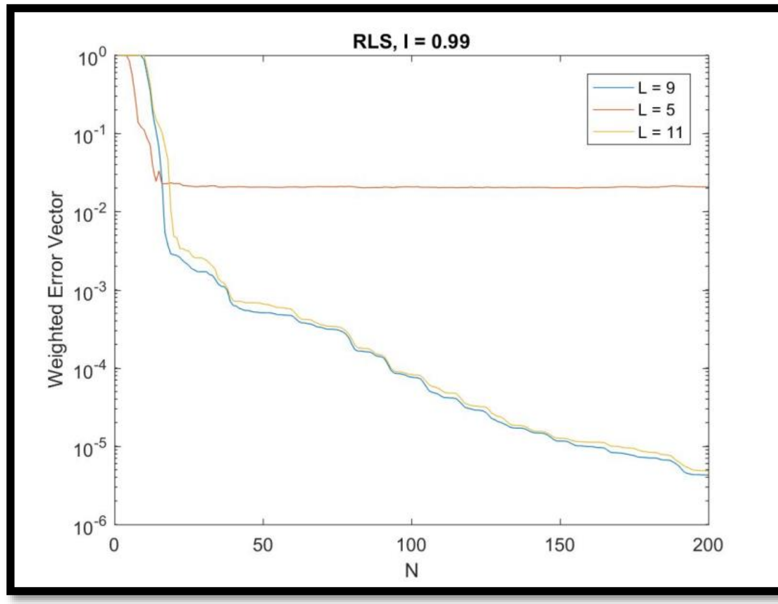
In the presence of noise, the results are worse than before. From the average squared error diagram, it is difficult to extract information about the different forgetting factors, but it seems that the performance is better than the LMS algorithm. From the average weight error vector diagram, it is clear that increase in the forgetting factor causes decrease in the error. Finally, comparing to the corresponding figures from LMS, the RLS algorithm is better.

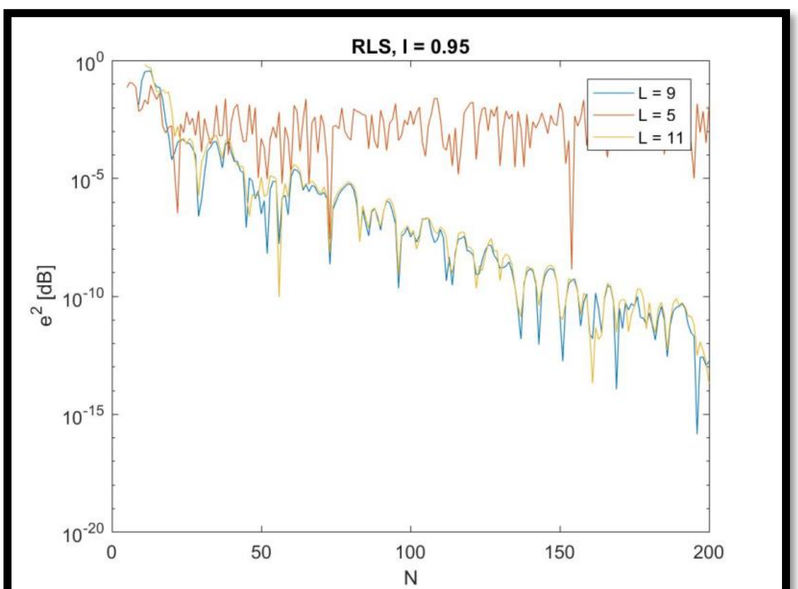


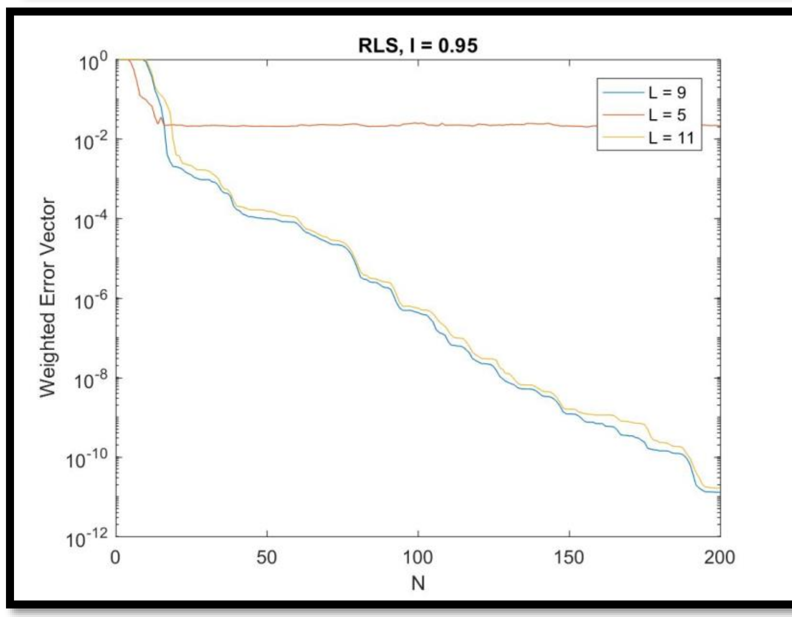


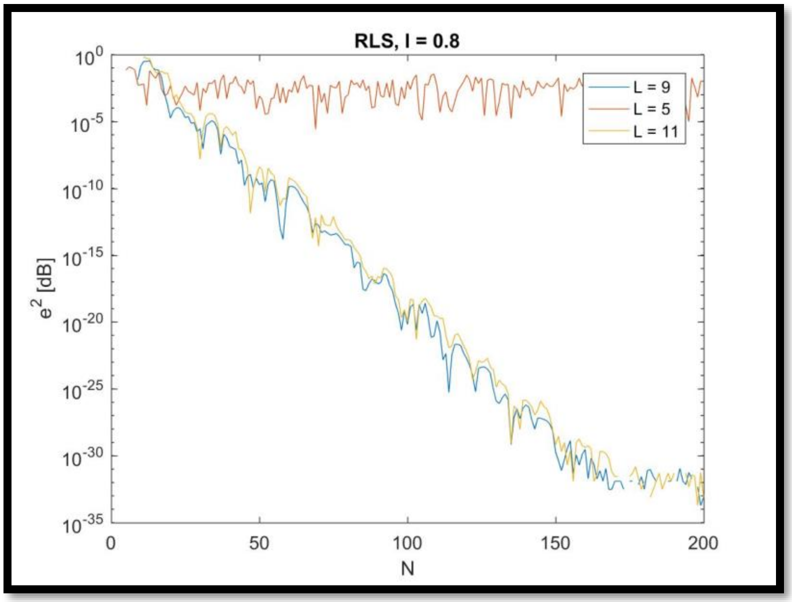
iii. In this question the performance of the RLS algorithm, for different forgetting factors and filter's length, is presented. Firstly, one can realize that overestimation of the unknown system's length is not a major problem, a fact that makes RLS algorithm more useful than the LMS one. Secondly, as the forgetting factor is decreasing, the squared error and the weighted error vector norms are decreasing too. This is a very positive result when there is no presence of noise in the system. Finally, it can be seen again that when $\lambda = 0.8$ the error is almost zero.

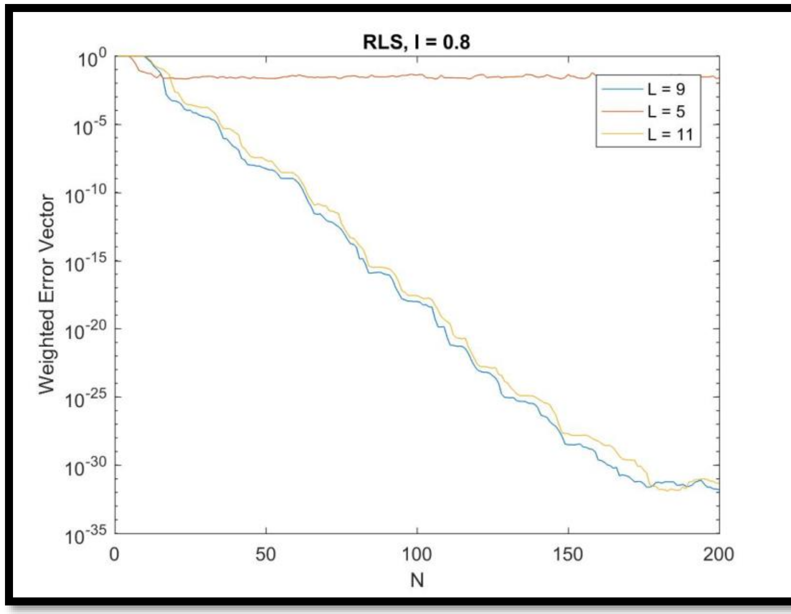












iv. If the coefficients of the unknown system are varied with sample number, the adaptive algorithms have a difficulty in estimating the new parameters of it. Thus, the algorithm needs some time to recalculate the new parameters and minimize the error. Below, the performance of the three algorithms for a time-varying channel is depicted.