Normalised LMS Algorithm - Interference Cancelling, Project 1

Madhusudan Govindraju
University of Florida
Email: madhusudangr@ufl.edu

***Abstract*—This papers gives a detailed report on how the problem statement for the project has been handled, with the assumptions and the learnings. The problem statement is an application of Normalized LMS algorithm to remove the noise from an audio signal. The problem also states that the input signal is 21KHz. We are to design and evaluate the performance of an adaptive FIR filter using the NLMS algorithm. This is also considered as an inference cancelling problem.**

I. Introduction

With wiener solutions there are methods which explains, when the reference input is free of signal noise in the primary input can be cancelled without any or very less signal distortion[1]. There are various applications for adaptive noise cancelling in the industry, some of them include cancelling interference in the ECG, speech signal, borad-band interference in an antenna array [1]. The paper [1] also talks about the design of the filter to be used. We can use a fixed filter for this problem of noise cancelling if we have prior knowledge of both the signal and the noise. But adaptive filters will adjust their own parameters automatically, without prior knowledge of the signal or noise. One famous commercial application of adaptive filtering is the MODEM for digital communication, which are widely used in connecting computers through the internet with less interference.

II. Literature Review

The earliest works in noise cancelling for Adaptive filters were "P.Howells, Intermediate frequency side-lobe canceller"[2],"B.Widrow & M.Hoff Adaptive Switching circuits" [3], "N.Nilsson, Learning Machines" [4]. Widrow and Hoff talk about an adaptive algorithm and pattern recognition scheme called the "Adaline" - short for "Adaptive Linear Threshold Logic Element" in [3].

III. Adaptive Filtering

The basic model for adaptive filtering is shown in figure 1. The $X(n)$ is the input signal and we need to predict the $Y(n)$. The desired output signal is denoted as $d(n)$ in the adder. The error between the desired and the obtained output signal will give rise to an error signal, based on this error signal the parameters of the adaptive algorithm are changed to give the $w(n+1)$ from the formula

$$w(n+1) = w(n) + StepSize * Error * X(n)$$

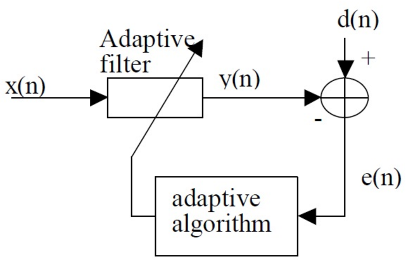

Fig. 1. Basic Block Diagram of Adaptive Filter

where $w(n+1)$ is the new parameter and the $w(n)$ is the old parameter. This special adaptive filter is referred to as the LMS algorithm. To control the deviation caused due to the change in power of signal in real time we can normalize the signal with the power of the input signal to get the NLMS - Normalized LMS algorithm.

The equations to explain the Normalized LMS algorithm is as follows.

$$Y = W^T \times X$$

$$Error = Desired - Y$$

$$w(n+1) = w(n) + [StepSize/X(n) * X(n)'] * Error * X(n)$$

"NLMS is an iterative gradient descent algorithm that uses an estimate of the gradient on the mean square error surface to seek the minimum optimum weight vector at the minimum mean square error point. "[5]

So this system actually corrects itself to learn from the error signal produced. So after a few iterations when the error signal goes to zero the desired output is approximately equal to the obtained output leading to a system which predicts exactly what we need.

When we look a the convergence performance of the Normalized LMS algorithm, we can see that the normalized algorithm will adapt to the system in far fewer iterations.

To actually achieve the proper working of the proposed method we have to consider a few factors they are the step size and the filter order.

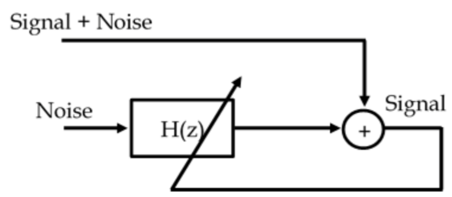

Fig. 2. Block Diagram for our problem

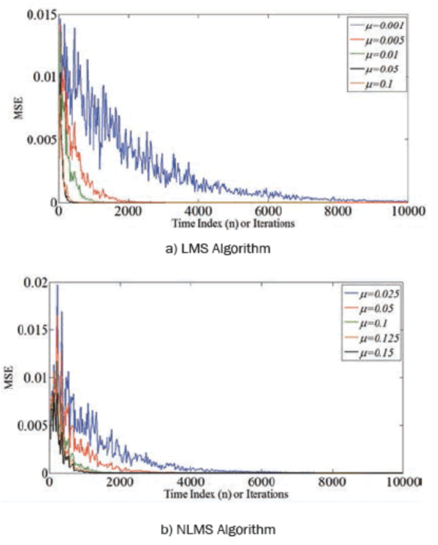

Fig. 3. Learning Rate for different Step Sizes [6]

A. Step Size

The variable step size is actually a constant that must be given when the filter begins its prediction. There are a few conditions to choosing the step size. The step size(mu) should satisfy the following.

$$0 < mu < 1/\lambda_{max}$$

where λ_{max} is the largest eigen value of the auto correlation matrix obtained with the input over the delav elements and

$$mu < 1/trace[R]$$

where R is the auto correlation matrix. From these we understand that the step size is dependent on the input signal, and the input signal in our case is a real time signal and hence it will depend on the number of taps in the filter. The number of taps is essentially the filter order in the filter we have designed.

And it is said that maximum convergence speed is achieved when the

$$mu = 2/(\lambda_{max} + \lambda_{min})$$

Thus filter's learning rate is thus directly controlled with the step size, this can be seen in the figure 3. [6] As we can see in the figure, as the step size increases, the learning time reduces but the final cost at which the system stops is a little higher than that for the optimal step size. Using a very small step size will reach the point of least cost but it will take a very long time to reach that optimal least cost. Thus using a values in between will reach a cost that is sufficient at well as a shorter delay. This relation between the step size and the learning time can be given by the formula

$$\tau = 1/4 \times mu \times \lambda_n$$

where $n = 1, 2, 3, 4, 5......L$ and L is the order of the filter here. So choosing the best step size is important.

B. Filter Order

The filter order or the number of taps depend completely on the system under observation. Based on the number of taps or the filter order the step size can be varied to obtain a good learning curve to suit our requirements perfectly. The higher the filter order we will over fit the training data set and it will be disastrous when we try to predict on a completely different testing dataset. The lesser filter order will under fit the data and will also lead to a lot of errors. So the only way to choose the best filter order is with the cross validating across the Training, Validation and Testing Datasets.

IV. Proposed Method

In our case we are going to slightly modify the signals in the block diagram so that we can understand our problem in case. The modified block diagram can be seen in fig 2

As seen from the figure, the signal with the noise(which is the primary in our case) is given as the desired output to the adder while the only noise signal(which is the reference in our case) is given as the input to the adaptive filter. So this adaptive filter optimizes its parameters to add the speech signal components to the reference to get the primary signal which is the speech with the noise. So the optimized parameters give us the speech components of the signal and hence the error that comes up in the last iteration should be completely free of noise.

Also for cross validation we have to separate the data into Training Data , Validation Data and Testing Data. The ratio for training validation and testing chosen is 5:1:4 respectively.

1) First we run the NLMS algorithm for the training set to learn the weights and then run the same with the validation sets to get the best hyper parameters. Now with the best hyper parameters we run in the testing data set to obtain the voice signal. We can go ahead the test the hyper parameters on the entire data set to obtain the full and complete voice signal.
2) Selecting the step size : In our case the input signal is a realtime online signal and hence calculating the theoretical step size(mu) is not possible. One way to estimate the theoretical step size would be to wait and collect a huge number of samlpes (ie., 100 samples) and use it to calculate the Eigen spread and estimate the step size. The other way would be to use an educated guess to estimate the step size by letting the algorithm run for

a few different step size and filter order combination and choose the combination with which we obtain the least Mean Squared Error, which here is also referred to as the Cost. Once we find the combination which gives us the least cost, we use it to do the other calculations.

3) Selecting the filter order : As explained in the previous subsection, in our case, we let the system run for a couple of different filter order and step size settings and choose the one that gives the least mean squared error.

V. Observations

For Obesrvation Step 1 the various step sizes used for setup are $[0.01 : 0.01 : 0.5]$, the various filter orders used in this setup are $[2, 8, 10, 20, 25, 30, 32, 36, 38, 40, 44, 46, 48, 50, 55, 60, 65]$. For Observation Step 2 the various step sizes used for setup are $[0.01 : 0.01 : 0.3]$, the various filter orders used in this setup are $[10, 20, 50]$. First, we train the system and then validate the system to obtain the best hyper parameters.

A. *Observation Set 1*

1) Two weight case : In this case we specify the filter order as 2. We choose the step size as 0.1 and calculate MSE for the different weights. To plot the MSE vs W1 vs W2 plot. We first run the NLMS algorithm with the above mentioned hyper parameters and then get the different W1 and W2s. Now we check the limit of the the weights. and then discretize the range so that it becomes computationally do-able and then calculate the cost train for the different weights. This gives us a cost train for the different weight starting point. Thus plotting a line plot for each of the cost train we find that the system runs in a way to reduce the cost and chooses a weight train that reduces the cost. This explains the adaptive nature of the filter. The figures 5 &4 are the 2 different views of the plots. The 2 views are provided for better understanding of the plot surface.

   From the figures 4 & 5 we can see that , even when we start at different points in the weight surface we tend to converge at the cluster (where we have a lot of points). Also this would look like a concave surface if the points were replaced by a plane. With the concave facing upwards. The cluster is also referred to as the local minimum or global minimum in this case. That is the point where we obtain the least cost. We can also understand that the weights converge at the local minimum. This has been plotted for 30000 iterations so it also shows that it needs around 30000 iterations for attaining the local minimum. But in the case of higher order which will be explained in the further sections we can see that we will attain the local minimum at around 50 iterations itself. This is the MSE surface for the 2 weight case for our problem.

   The figure 6 gives us the different values of W1 and W2 over iterations.

2) The learning curve for different step size mu has been plot and is available in the figure 7. This figure 7 is a zoom into the learning curve to actually see where we obtain the first convergence . From the learning curve we can observe the following
   a) For higher step size $mu = 0.27$ the learning curve converges in less than 5 iterations but for lower step sizes of $mu = 0.20$ & $mu = -0.02$ the filter converges at around 15 and 50+ iterations respectively. This confirms our proof that learning rate is quicker at higher step sizes and slower at lower step sizes.
   b) Their oscillations around the mean point can be seen in the next set of figure 8, 9 & 10. Also we can clearly see that for higher step size the curve converges sooner.

3) We calculate the ERLE(Echo Return Loss Enhancement) improvement to measure the quality of the voice signal we obtain for the given input and the desired output for different hyper parameters and plot the ERLE vs step size and filter order. This plot is available in figure 11. This figure also shows us that the hyper parameters chosen for the system can also be chosen from this plot. The point of highest dB coincides with point of minimum error of the MSE plot which is available in the figure 12. We get the best performance with filter order 48 and step size 0.25 for our system. We can filter the signal for any local impulse response and remove any jitter or vibrations before calculating the plot.
   The ERLE was calculated using the formula

$$ERLE = 10log_{10}(PowerofDesiredSignal/PowerofNoise)$$

   The best mean ERLE obtained was 21.9740. The best SNR improvement or best variance or ERLE obtained was 30.96 . These were also the optimum points of hyper parameters.

   The voice signal is ”I will not condone a course of action that will lead us to war”

B. *Observation Set 2*

Improving the performance by improving the filter order is accepted but, it may lead to over fitting of the trained data. In real time the data we obtain for training and testing is not going to be similar and may vary and hence if we have over-fitted it, it will cause high error rates, but in our case as the training and testing is of the same signal type obtained from the same source overfitting will not be a problem up-to a level.

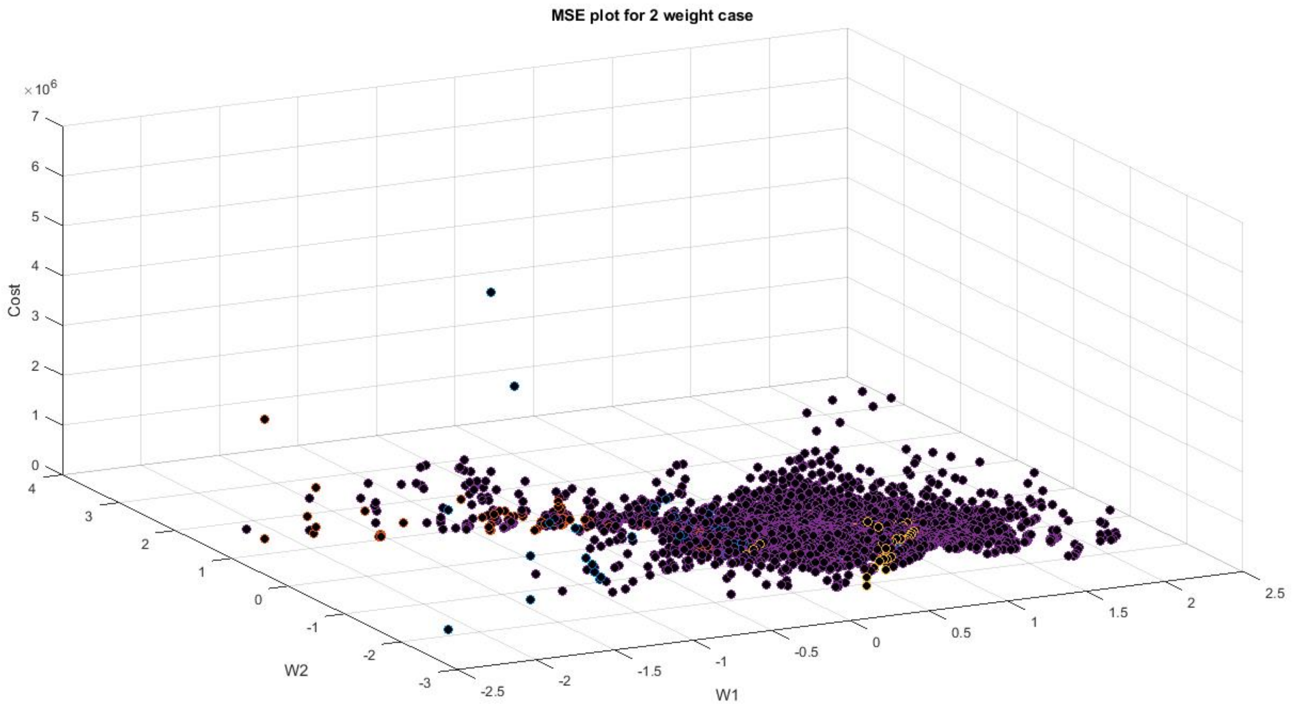

Fig. 4. MSE plot for 2 Weight Case View 1

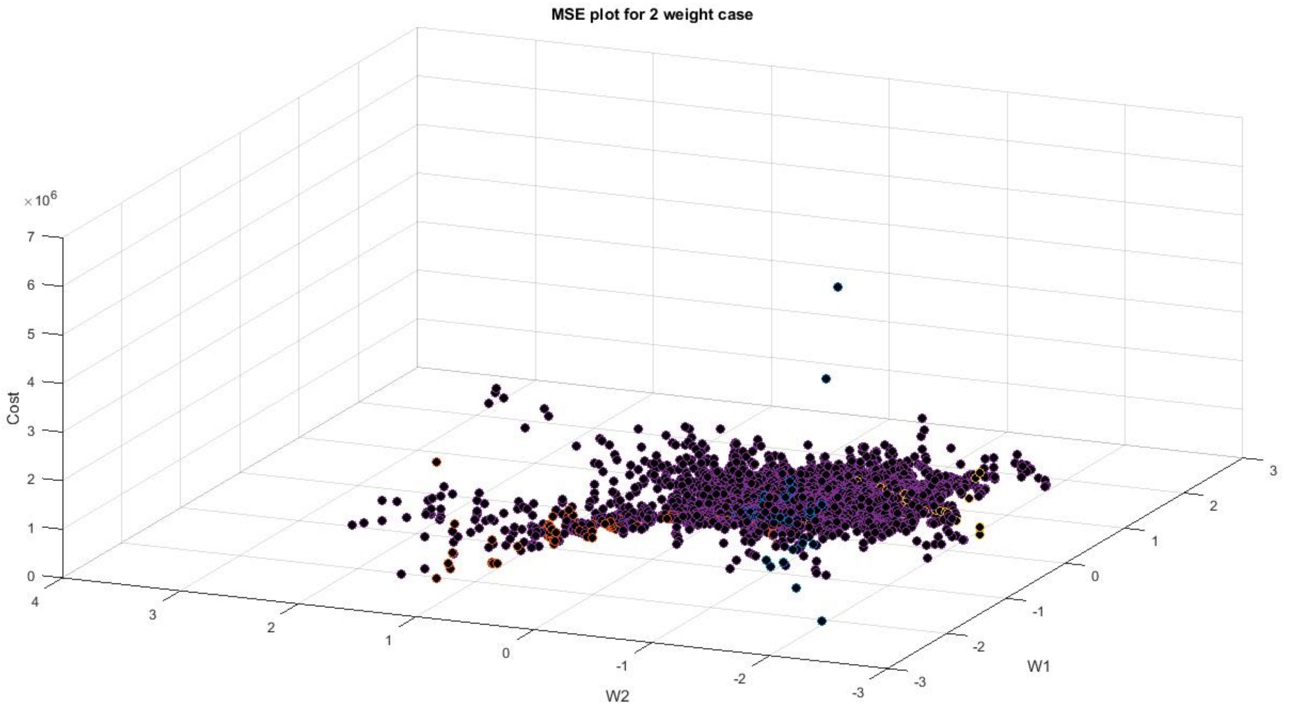

Fig. 5. MSE plot for 2 Weight Case View 2

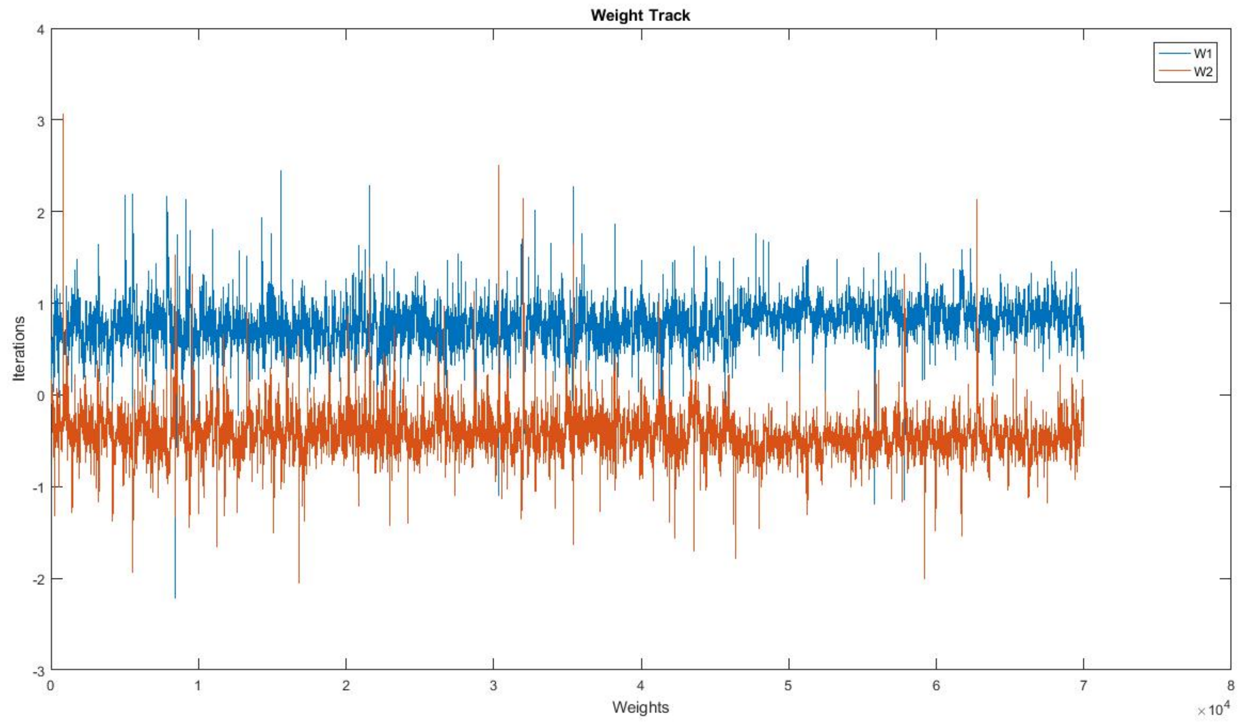

Fig. 6. WeightTracks Over Iterations

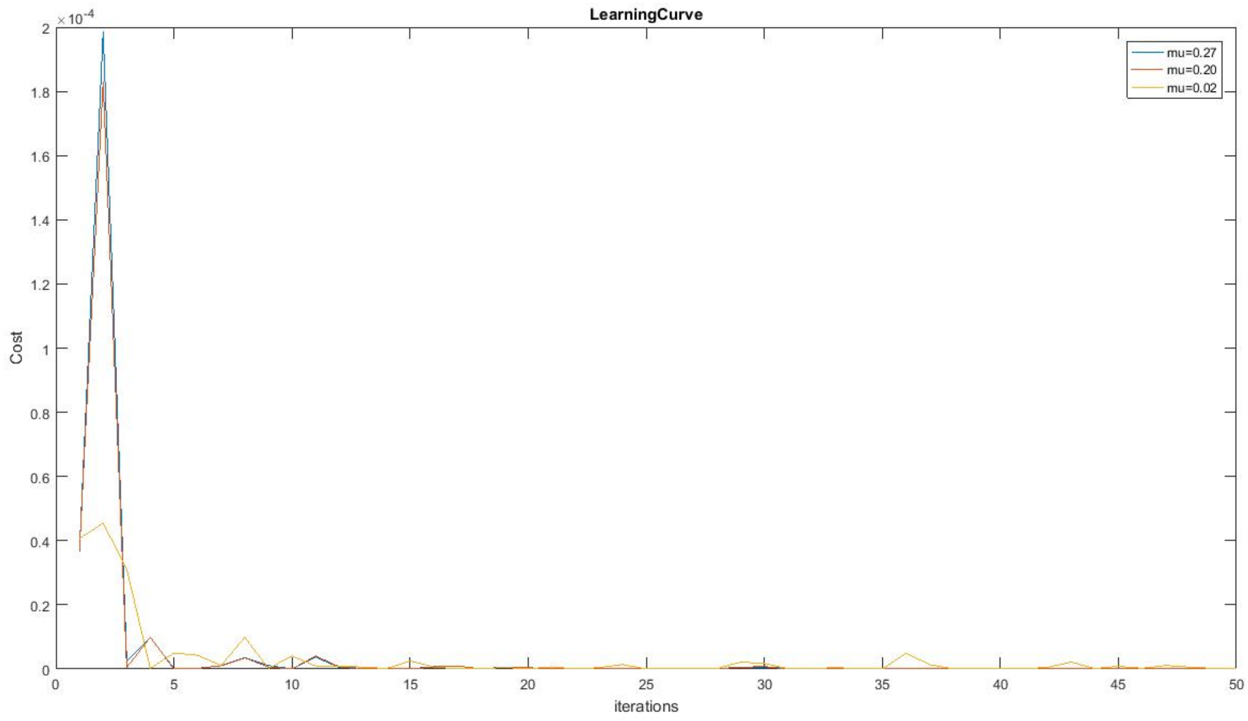

Fig. 7. Learning Curve zoomed in for 50 iterations

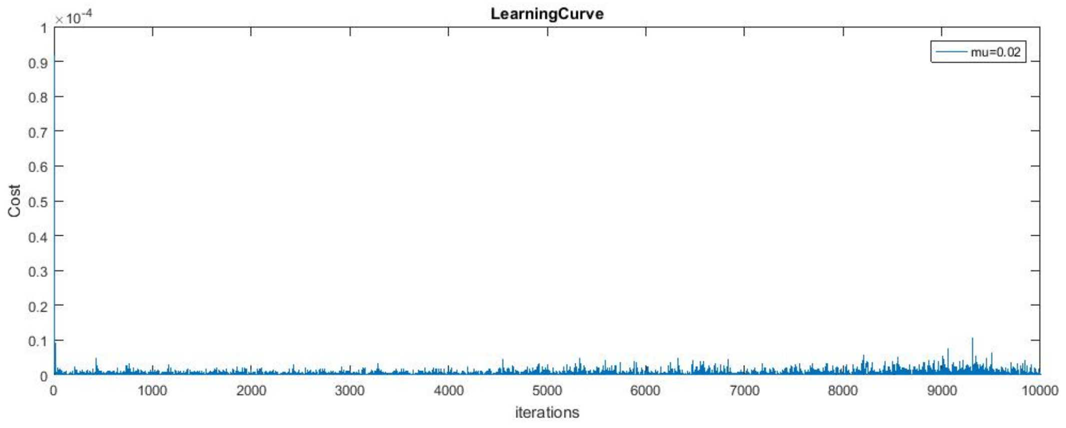

Fig. 8. LearningCurve for mu = 0.02

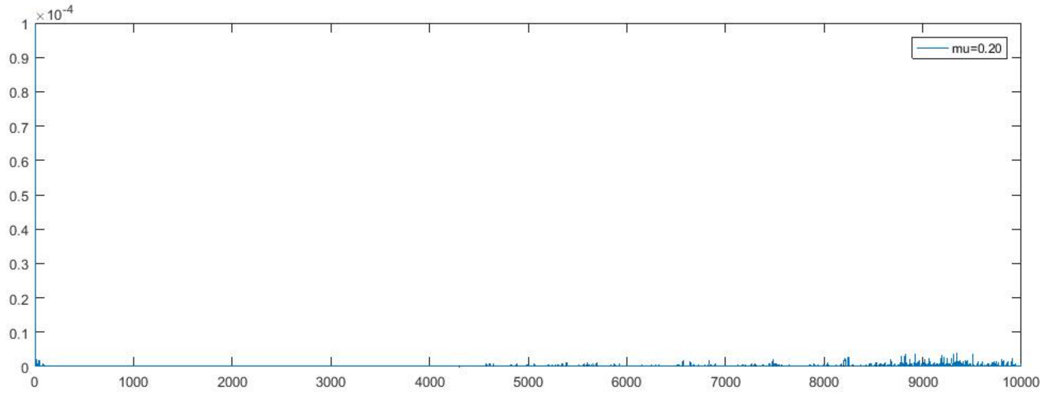

Fig. 9. LearningCurvefor mu = 0.20

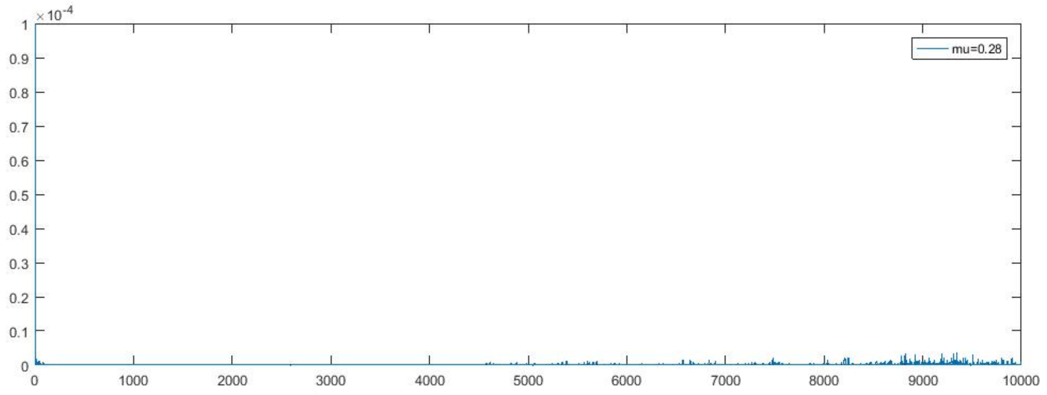

Fig. 10. LearningCurve for mu = 0.28

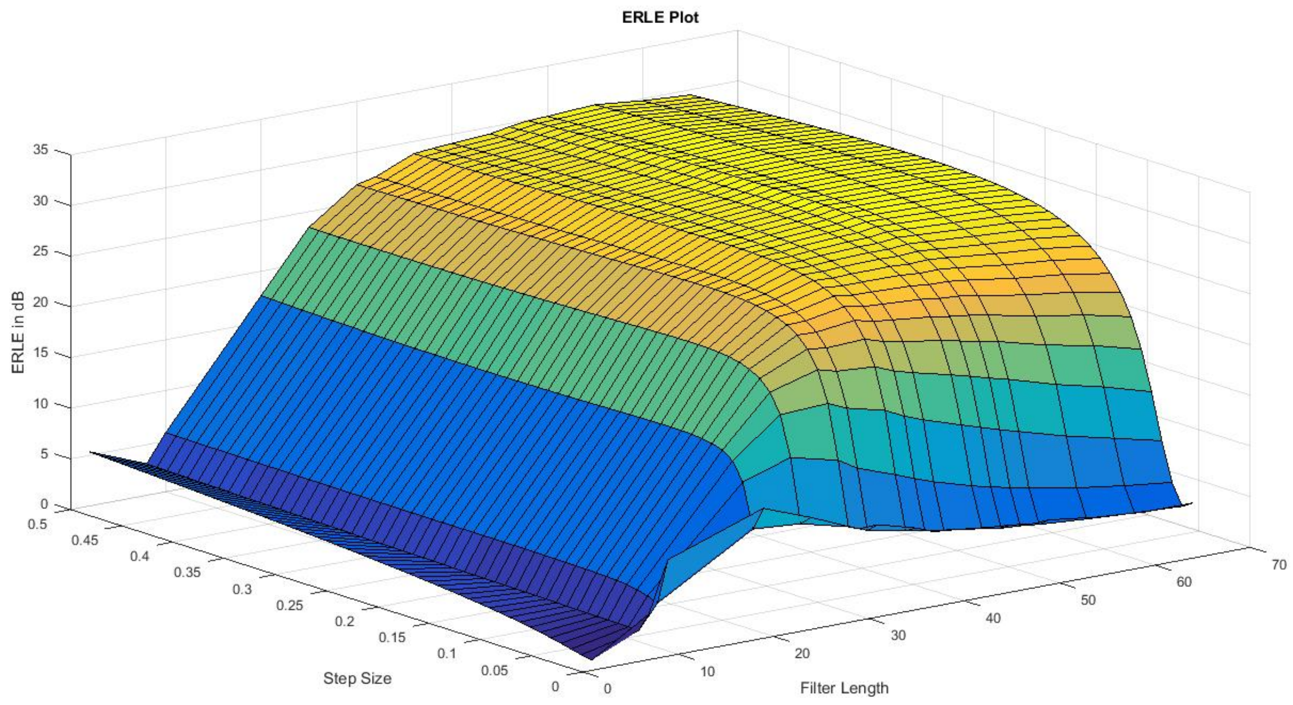

Fig. 11. ERLE vs Hyperparameters

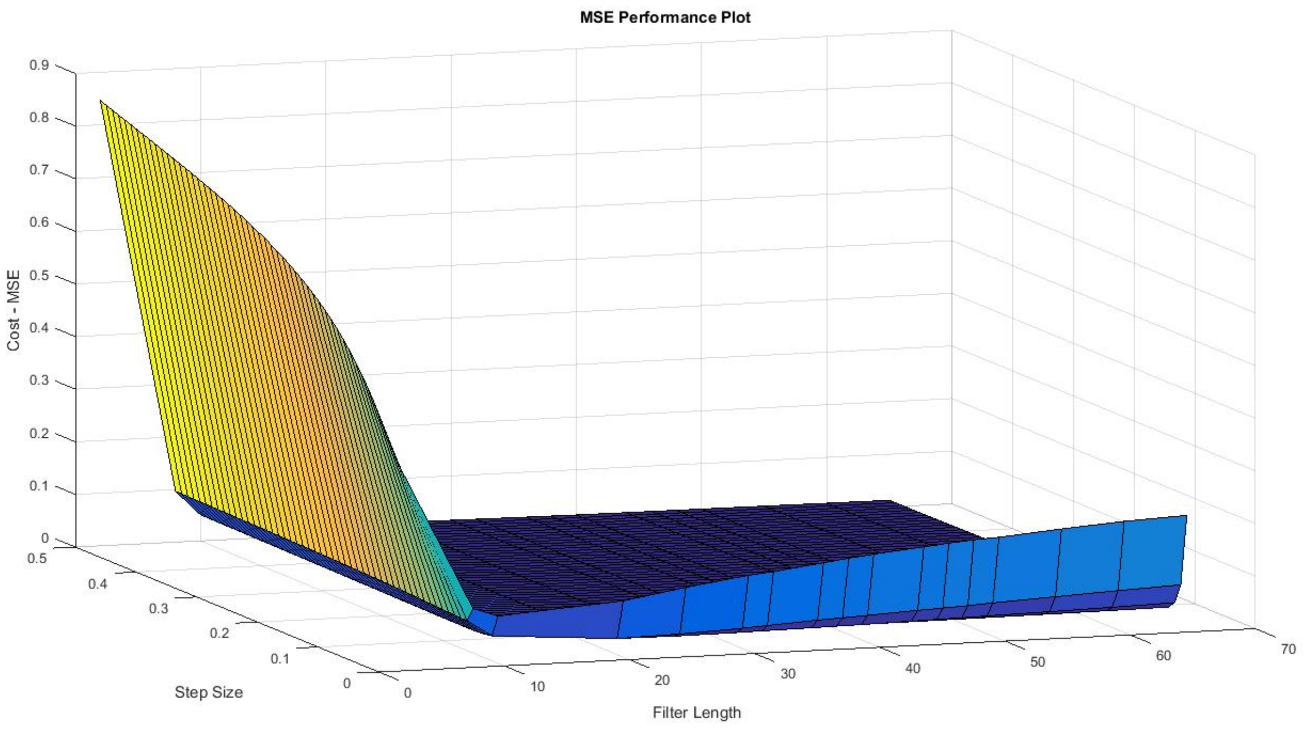

Fig. 12. MSE vs Hyperparameters

Yes we can cross validate based on ERLE, this has been explained in the observation 3 in observation set 1. The ERLE plot can be used similar to the MSE performance surface plot but, we will have to choose a point of maximum ERLE change compared to the lowest point in MSE surface plot. The plots are available in figures 12 & 11.

1) The plot for the filter orders of 20, 30, and 50 against different step sizes vs MSE is available in figure 13. From the plot we can see that as the filter order increases the performance increases(ERLE increases).
   Looking at the plot we get maximum change in ERLE of 30.9439 at $stepsize = 0.26$ and $filterorder = 50$ which is similar to the hyper-parameters we obtain in the MSE plot which is available in figure 14
2) We get the best performance at step size = 0.26 which is little lesser than that for filter order 48 which we had chosen as our optimal parameter. This again proves the point that increasing the filter order we can increase the performance (but might lead to overfitting). The learning curve for the best performance and the 2 other learning curves for comparison are provided in figures 15 16 17. A zoomed version of it is available in 18, we can see that this is very similar to the one obtained in the observation set 1.
3) The best filter does give the best speech intelligibility. The speech signal is obtained at the very least order itself but it still has a component of noise in it. As we increase the filter order the speech intelligibility increases and the vacuum cleaner noise reduces(it is as though the noise moves further away from the voice signal). Thus best voice signal is obtained in the filter order 50 in our second observation.

VI. Conclusion

1) The Normalized LMS converges much faster than the LMS algorithm
2) The voice signal is separated and it is from the famous movie Star Wars Episode 1- Phantom Menace ”I will not condone a course of action that will lead us to war”.
3) ERLE ; Filter order 10 step size 0.26 = 5.6255
4) ERLE ; Filter order 20 step size 0.26 = 17.1791
5) ERLE ; Filter order 50 step size 0.26 = 30.9439
6) ERLE ; Filter order 48 step size 0.25 = 30.9686 , This is also the optimal point for our hyper parameters.

References

[1] B. Widrow et al., ”Adaptive noise cancelling: Principles and applications,” in Proceedings of the IEEE, vol. 63, no. 12, pp. 1692-1716, Dec. 1975. doi: 10.1109/PROC.1975.10036

[2] P. Howells, ?Intermediatefrequency side-lobe canceller,? U S . Patent 3 202 990, Aug. 24,1965

[3] B. Widrow and M. Hoff, Jr., ?Adaptive switching circuits,? in IRE WESCON Conv. Rec., pt. 4, pp. 96-104,1960.

[4] N.Nilsson ,LearningMachines. New York: McGraw-Hill,1965

[5] J. Glover, ”Adaptive noise canceling applied to sinusoidal interferences,” in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 25, no. 6, pp. 484-491, Dec 1977

[6] JIMENEZ-LOPEZ, Fabin Rolando; PARDO-BEAINY, Camilo Ernesto and GUTIERREZ-CACERES, Edgar Andrs. Adaptive filtering implemented over TMS320c6713 DSP platform for system identification. Iteckne [online]. 2014, vol.11, n.2 [cited 2016-10-14], pp.157-171

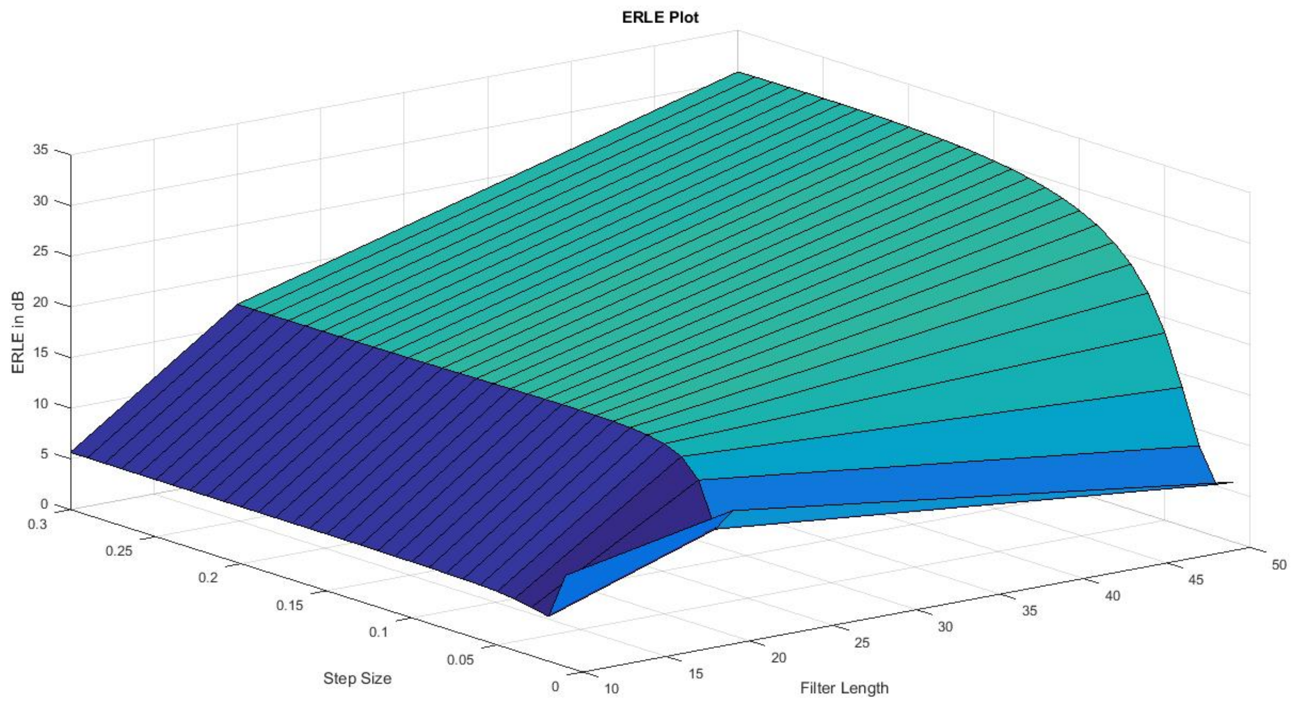

Fig. 13. ERLE for Filter Order 10 , 20 and 50

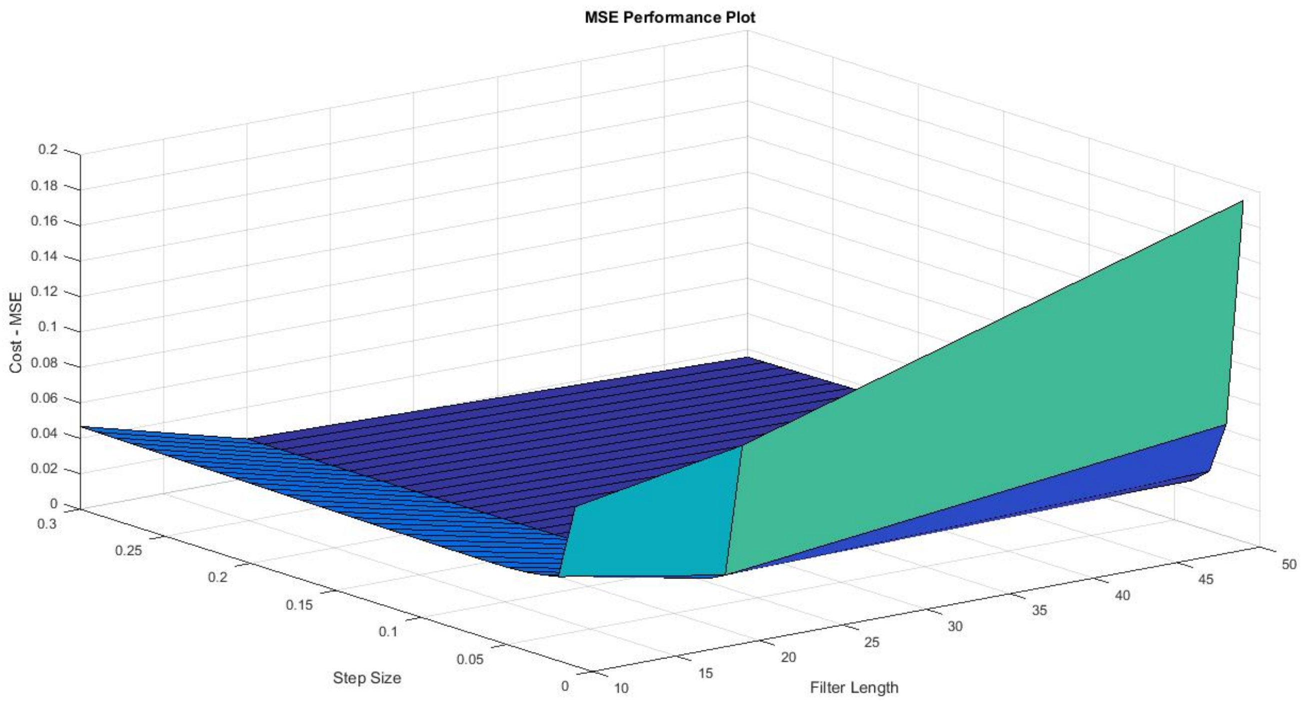

Fig. 14. ERLE for Filter Order 10 , 20 and 50

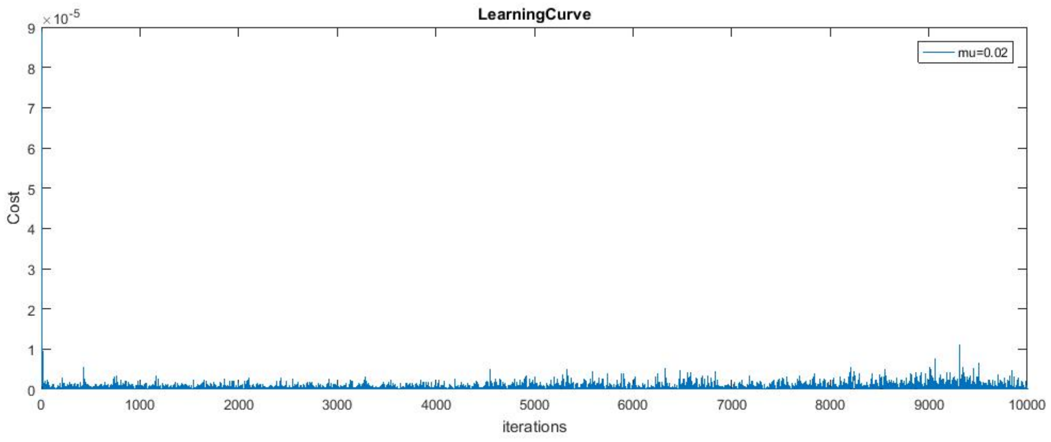

Fig. 15. Learning Curve Filter order 50 mu 0.02

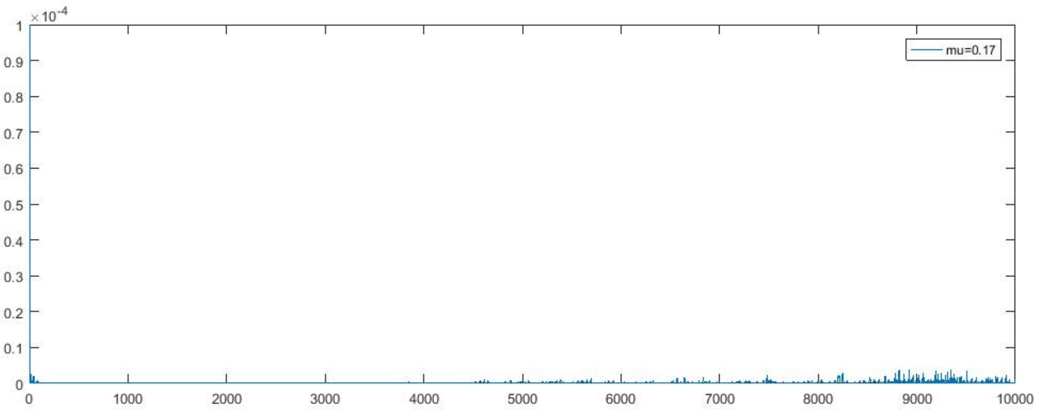

Fig. 16. Learning Curve Filter order 50 mu 0.17

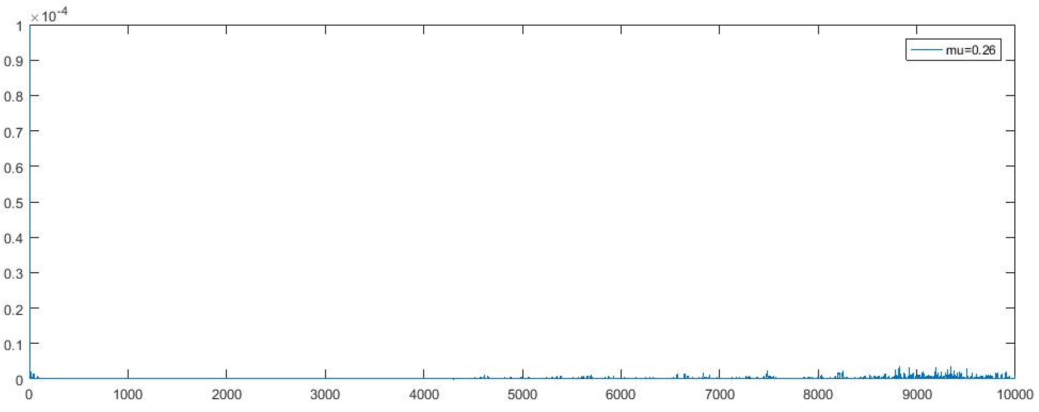

Fig. 17. Learning Curve Filter order 50 mu 0.26

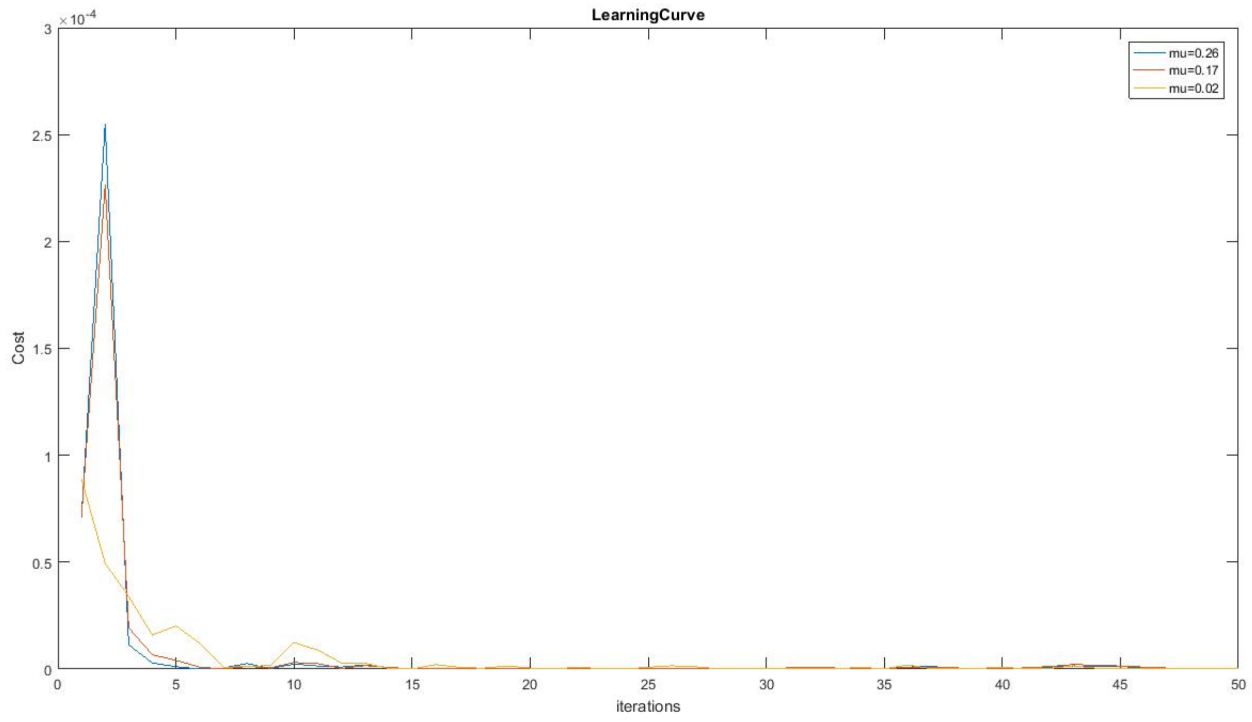

Fig. 18. Learning Curve zoomed in for Order 50