IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Effect Of Learning Rates On Adaptive Filter Algorithms And Implementation Of Kernel Adaptive Filters

[1]Arpit Singh, [2]Rahul Pandya, [3]Ashish Sharma, [4]Tarush Singh, [5]Dr. P. N. Sonar

[1,2,3,4,5]Department of Electronics and Telecommunications,

[1,2,3,4,5]MCT's Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India.

Abstract: This The Adaptive filter techniques being used for implementations of noise removal in signal processing systems are very significant. The use of different algorithm specific makes this more versatile and effective because of their intensive arithmetic architectural nature and thereby improving the efficiency of the results. This research paper deals with the implementation of the LMS, NLMS, RLS and Affine projection algorithms along with the kernel versions of LMS and RLS that is KLMS and KRLS. Experiments performed give the brief ideology about how these filters operate for noise and echo cancellation systems. The effect of learning rates of this filter plays a very significant aspect as it decides the stability and efficiency of the respective filters. Learning rates and step sizes make the use of adaptive filters more application specific. The kernel versions are operating on the previous iterated values with different regularization parameters. Adaptive filters can be diversely utilized at application specific purposes while using smart antenna systems, advanced FPGA signal processors and many other digital analytic machines.

Index Terms **– Adaptive filters, KLMS, KRLS, LMS, Noise cancellation, RLS**

I. INTRODUCTION

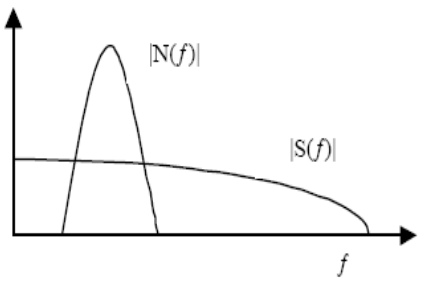
The primary aim of the adaptive filters is that to design a filter algorithm which can learn and interpret the conditions of the operating environment and then adjust or modify the respective parameters accordingly. These adaptive filters that estimate and adjusts its metrics could be designed in two ways:

1) Stochastic method.
2) Deterministic formulation.

When the signals are residing in separate frequency bands, by making the use of linear convolutional simple filters we can obtain the desired signal facilely. The challenge arises when the desired signals reside within the same frequency bands and the removal of the noise becomes difficult so as not to harm the required frequency response. Furthermore, removing only a particular signal can still be done conveniently, but when the signal responses are of varying frequencies within the same bandwidth, we need redefine the coefficients. Thus, adaptive filters are required for such purposes.

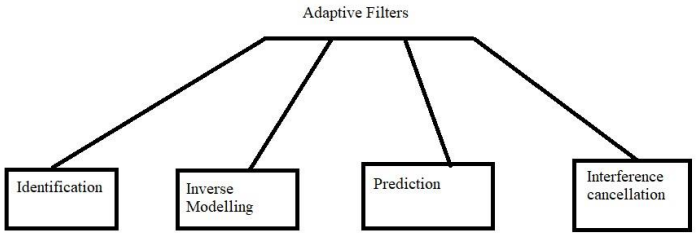
A basic example to be considered for the adaptive filters is the overlapping(interference) case of a wideband signal and a narrow band signal. The filter coefficients are calculated whilst a training sequence where the pattern and series of the transmitted data is well versed.

Figure 1. Overlapping of narrow band and wide band signals



Owing to the complicatedness of the algorithmic approach of the adaptive filters, almost all adaptive filters are digital filters. Depending upon the manner in which the information needs to be extracted adaptive filters are further classified into further sub categories. [1]

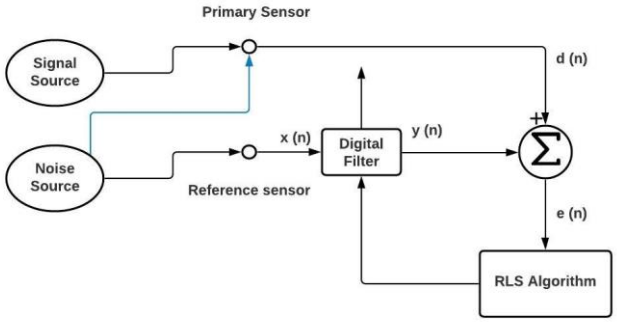
Figure 2. Classification of Adaptive Filters.



Adaptive filters are having varied applications in the field of:

- Noise and Volume equalizers
- Echo Cancelling systems
- Speech Modulators

Figure shows a basic block implementation of a RLS algorithm adaptive filter for noise cancelation system.
Figure (3). RLS adaptive filter echo cancellation system.



II. ADAPTIVE FILTERS

2.1 The Wiener Filter

The Wiener filter was developed in the year 1949 by Norbert Wiener. [2] This filter is used as a reference to design the adaptive filters. Wiener filters are dependent on the Mean Squared Error Criterion. The Mean Squared error correction works on the basic principle of selecting parametric functions that would result into the least possible error as the error reaches zero, and the output of the filter approaches the desired signal. Consider a simple example of a filter with input signal u(n) fed to a filter with the taps (M). We get the following relation:

$$u(n) = [u(n), u(n-1) \dots u(n-M+1)]^T \qquad (2.1)$$

Similarly, the weight vectors are assigned as:

$$w = [w(n), w(n-1) \dots w(n-M+1)]^T \qquad (2.2)$$

From the above equations, we can formulate the squared error as:

$$e_n^2 = d_n^2 - 2d_n u_n^T w + w^T u_n u_n^T w \qquad (2.3)$$

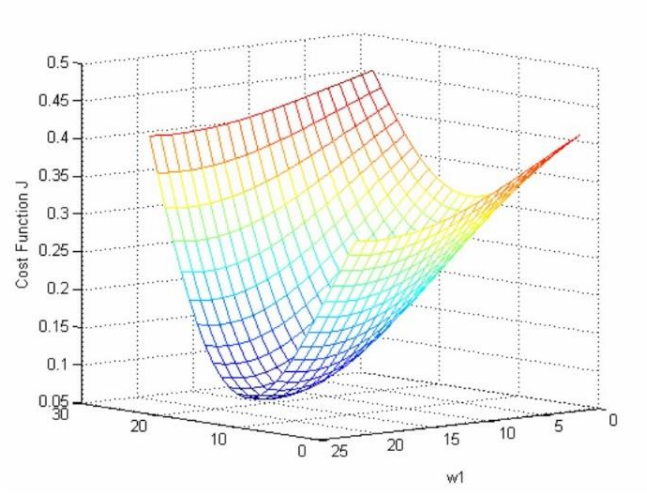
Mean squared error = J = $E[e_n^2] = \sigma^2 + 2p^T w + w^T R w$ (2.4)

Where, σ = variance,
R = autocorrelation matrix of u, and
p = cross-correlation vector.
d = target vector.

Plotting this gives us a non-negative bowl shaped figure with the minimum points at the optimal weights. This is generally known as the error performance surface given by:

$\nabla = \frac{dJ}{dw} = -2p + 2Rw$ (2.5) [3]

Figure (4). Error Performance Function



The optimal Weiner filter can be obtained from the Weiner-Hopf equation which can be written as:

$$Rw_0 = p \quad (2.6)$$

Where the filter coefficients are : $w_0 = [w_{00}, w_{01}, \dots w_{0(M-1)}]^T$ (2.7)

The above equation produces required results but the Weiner-Hopf equations need to be recalculated as the variations change. Therefore, this method is used to generate other adaptive algorithms based on the approach.

2.2 Least Mean Square (LMS) Algorithm.

The least mean squares algorithm was first implemented in 1950 by Widrow and Hoff in 1960 [4]. This is one of the simplest and widely used algorithm for implementation. It is a part of the stochastic gradient algorithms owing to its very easy solving methodology adopted.

The LMS algorithm is very subtle and simple in terms of complexity to be implemented. The very fundamental use of this algorithm is to simplify the solvability of the Weiner Hoff equation. It discards the computation of inversion matrix and auto correlation matrices. The LMS filter has a definable step size parameter μ which can control the stability and the convergence of the algorithm.

Algorithmic approach of the LMS:

- Step1: Take the output of the Finite impulse response with inputs from filter coefficients as tap inputs.
- Step2: Calculation of the estimated error with respect to the output needed.
- Step 3: Redefinition of the filter coefficients with respect to the calculated error.

In mathematical representation:

$y(n) = w^T(n)\, x(n)$ (2.8) is the Filter output

$e(n) = d(n) - y(n)$ (2.9) is the Error.

Hence, we get the Filter coefficients and the input given to the filters as:

$$w(n) = [w_0,\ w_1, \dots w_{M-1}]^T \quad (2.10)$$

With inputs: $x(n) = [x(n),\ x(n-1), \dots x(n-M+1)]^T$ (2.11)

The filter weights for the LMS filter can be described as the following adaption with the learning rate μ.

$\Delta w(n) = \frac{1}{2} \frac{\partial e^2 n}{\partial w(n)} = \mu . e(n) . x(n)$ (2.12) [5]

2.3 NLMS Filter

The NLMS or normalized least mean square filter is just an extended version of the standard LMS filter. Here the learning rate of the filter is defined or normalized. The generalized learning rate μ is replaced by η(k) with each and every single iteration upcoming as:

$\eta(k) = \frac{\mu}{\epsilon + ||x(k)||^2}$ (2.13) [6]

Here, in the denominator ϵ is the regularization parameter for ensuring the stability of the filter. It improves efficiency for input norms of $||x(k)||^2$ which are very close or limiting to zero and thus preventing the arise of indeterminate functions.

2.4 Recursive Least Square Algorithm

In practicality, the implementation of adaptive filters requires more advancements. There is a need of all the past samples that were iterated upon along with the new updated coefficients along with the results. This can be achieved by the implementation of the RLS or recursive least squares algorithm. The RLS algorithm has faster computational processes as compared to the standard LMS algorithms.

Algorithmic approach of the RLS algorithm:

- Step 1: Calculation of the gain vector.
- Step 2: Process of Filtering.
- Step 3: Estimation of the respective errors.
- Step 4: Vector calculation of n taps.
- Step 5: Updating the coefficients.

Mathematically, the standard RLS equation can be quoted as:

$$\hat{w}(n) = \hat{w}(n-1) + k(n)\,\hat{e}_{n-1}(n) \quad (2.14)$$

where, k(n) = gain vector

and $\hat{e}_{n-1}(n)$ = estimation error.

Another way of representing the working of an RLS algorithm is:

$y(k) = x^T(k).w(k)$ (2.15) with order of filter n,

$x(k) = [x_1(k), \dots, x_n(k)]$ (2.16) as the input vector,

with update vector w as $w(k+1) = w(k) + R(k)e(k)x(k)$ (2.17)

Here, R(k) is the inverse auto-correlation matrix defined as:

$R(k) = \frac{1}{\mu}[\,R(k-1) - \frac{R(k-1).x(k).x(k)^T.R(k-1)}{\mu + x(k)^T.R(k-1).x(k)}$ (2.18) [7]

where μ is the forgetting factor and the initial value of this matrix R(k) is taken as:

$R(0) = \frac{1}{\delta}I$ where δ is the positive constant parameter.

2.5 Affine Projection Algorithm

The affine projection filter is another extended version of the LMS filter just like the NLMS filter. The tweak in the affine projection is that it feeds multiple input vectors at a time during each iteration sample. The number of vectors that are fed at a time to the filter is designated as the projection order of that filter. [8]

Input: $x_{AP}(k) = [\,x(k), \dots . x(k-L)]$ (2.19)

with L as the projection order of the filter.

Output: $y_{AP}(k) = {x_{AP}}^T(k).w(k)$ (2.20)

for target $d_{AP}(k) = [d(k), \dots . , d(k-L)]^T$ (2,21) in time k,

With error as: $e_{AP}(k) = d_{AP}(k) - y_{AP}(k)$ (2.22)

III. Methodology Adopted

An experimental procedure was adopted for the implementation of the both LMS and RLS algorithms for observing the computational properties and generate significant results. The following experiments were performed in Python software. One is Acoustic echo cancellation experiment where the mean squared error is calculated along with the convergence comparison with different step sizes. The basic LMS and RLS algorithms are performed and analyzed by plotting of the MSE to the number of iterations. The second methodology involves the performance of the algorithms in the respective learning rates defined helping us define a better learning rate for optimal analysis. The third experiment includes the kernel implementation of LMS and RLS algorithms popularly recognized as the KLMS and KRLS algorithms.

3.1 System Blueprint

Performing this experiment requires basic open source software**,** that is, Python. Here, it has been performed using the scripting language python because of its advanced libraries and ease of use with compatibility functions. A software Anaconda needs to be installed of the latest available version or above 3.6.6. It is configurable in both set of windows. Python of version above 3.0 should be used along with importing of the libraries mentioned below.

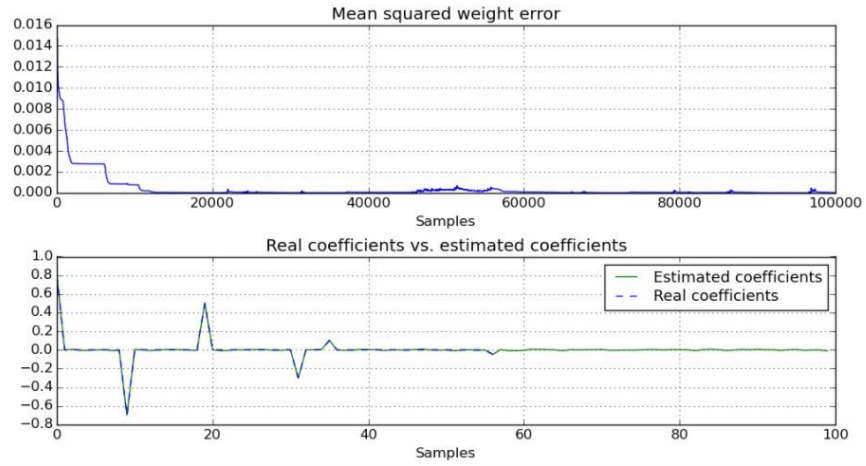
Required system libraries:

- Pandas – it is used to generate the CSV for the pulled and recorded tweets.
- Numpy package – it is used for numerical calculations in python library.
- Seaborn – it is a library used for statistical data visualisation.
- Scikit learn – it is used for unsupervised and supervised python learning libraries.
- Padasip – it is used to design adaptive filtering tasks in python. [9]
- Matplotlibrary – it is used for visualisation of graphical and statistical analytics.
- Adaptfilt 0.2 – an adaptive filter designing library in python. [10]

3.2 Echo Cancellation System

This simple experiment involves sending of a voice signal from one place to the other where the message signal sent comprises of a noise along with the original signal. The receiver at the output end need to cancel the noise from the echo. Considering the scenario, the signal sent by the second person is u(n) and the signal received is d(n). Now to remove the noise from the output filter, adaptive filter needs to be used. Here an NLMS filter needs to be applied so as to remove the noise.
Therefore, we get the following parameters:
Input: $u(n)$
Error signal: $e(n) = d(n) - y(n)$

Figure (6). Comparison of Coefficients.



Whilst performing the experiment, we generate the received signal d(n) using random coefficients. The filter tap count is taken ass hundred with a step size of 0.1. The mean squared error is then calculated and a plot of real coefficients along with approximated coefficients is obtained. The plot is shown in the Fig. (6). where (a) Mean squared errors and (b) is the coefficients plot.

3.3 Implementation with respect to learning curve

Implantation of the LMS and RLS algorithms with learning curve. That involves plotting of the mean squared error values (MSE) to the number of iterations performed on it 'i'. There are two ways of doing this:

1) The Ensemble Average Approach
2) Testing Mean Square Error (MSE)

The previous approach involves adaptive filters having the similar set of configurations of step size, updating parametric, number of iterations, initializations, etc. The input values and the desired values are different for each filter. For each filter, we plot the sample learning curve, which is simply the squared value of the estimation error versus the number of iterations and therefore a lot of noisy components are present in it owing to the inherently stochastic nature of the adaptive filter. The second step involves calculating the average of these sample learning curves on the ensemble of adaptive filters used in the experiment. This process helps in the eradication of the effects of noise. The averaged learning curve is called the ensemble – averaged learning curve. This method is applicable irrespective of the environment, stationary or nonstationary. [11]

The latter one involves setting aside a testing data set before the training. For each iteration, the weight estimates are computed. The mean square error is calculated on the testing data set. Then, we plot the testing MSE versus the number of iterations. This approach involves the use of only one adaptive filter unlike the earlier one. This method has a limitation as it is not applicable in situations where the environment is nonstationary. [10] Fig. 7(b) and 7(c)shows the implementation of the LMS and RLS algorithms using this approach.

Figure (7a). Normalised data samples for the LMS algorithm

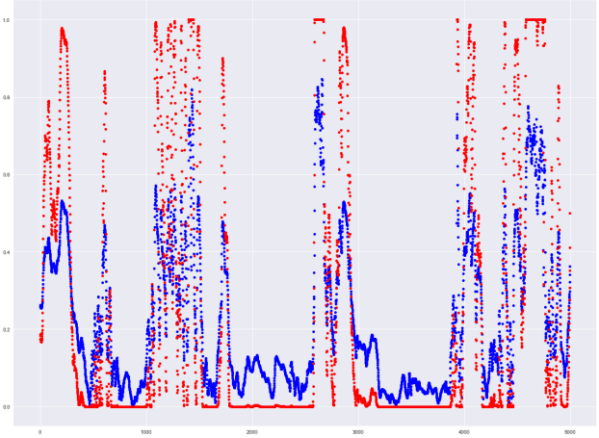
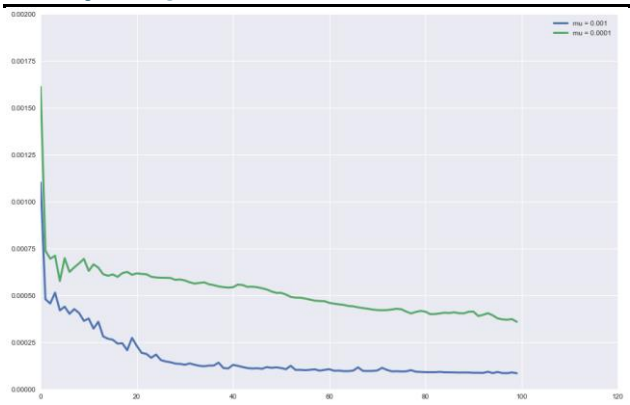
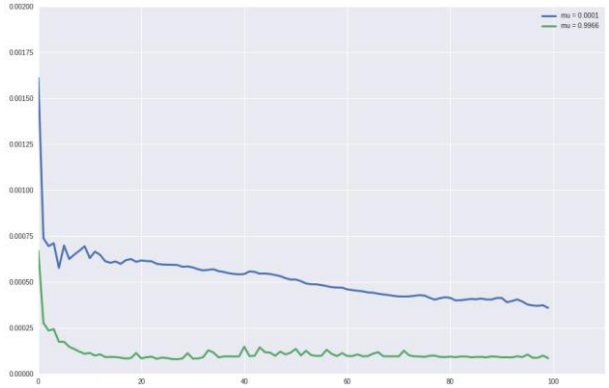


Figure (7b) Plot of the effect of mu on MSE and accuracy using LMS



The LMS algorithm has a step size of 1000 with a testing limit of 200 and the upper and lower limits for the data define as 100000 and 0 respectively. It performs well under the defined learning rates and generates significant results as compared to the LMS algorithm with similar input parameters.

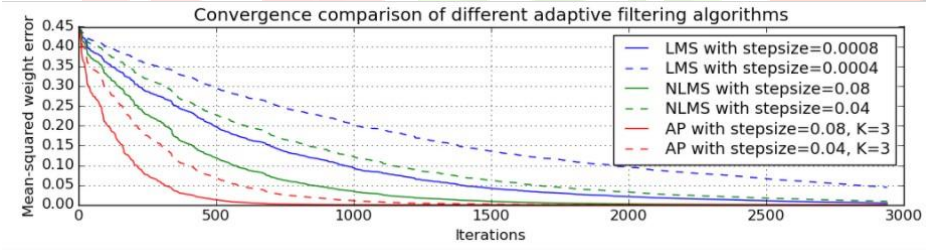
Figure (7c) Plot of effect of mu on MSE and accuracy after using RLS



While performing the experiment, it was observed that the filters for LMS and RLS algorithms using the second approach as mentioned above, the perform the best in the 'mu' that is the learning rate of [1e-3,1e-4] for LMS and [0.0001,0.9966] for RLS algorithms. Defining this learning rate is a very important step while implementing this procedure as a very high value may result in unstable result and a very low value may have an effect on the input data to the filter.

On generating a union plot for the LMS, NLMS and the AP (affine projection) with different step sizes over three thousand samples we obtain the performance graph of mean squared weights with respect to the iterations performed as shown in the Fig.8.

Figure (8). Comparison of filters based on step sizes



It can be observed in the Fig. (8). that the Affine projection algorithm with order of filter 3 and with a step size of 0.08 performs more precisely over the course and the number of iterations whereas the LMS with the least step size takes a much longer time minimize the error. This can be summarized as in Table (3.1)

Table (3.1). Filter Performance

Filter type	Step size	Approximate iterations required for minimising the error	Performance
LMS	0.0008	2400	Average
	0.0004	>3000	
NLMS	0.08	1500	Good
	0.04	2500	
AP	0.08	500	Best
	0.04	1000	

3.4 Effect of learning rates on LMS, NLMS, RLS and AP

On performing the experiment using the 'padasip' library in python for all the four algorithmic approaches of the filters we can obtain a better visualisation and result on the determination of the learning error rates of the respective filters and their performance on similar designed metric input samples.

Figure (9a). LMS with 'mu' =1.

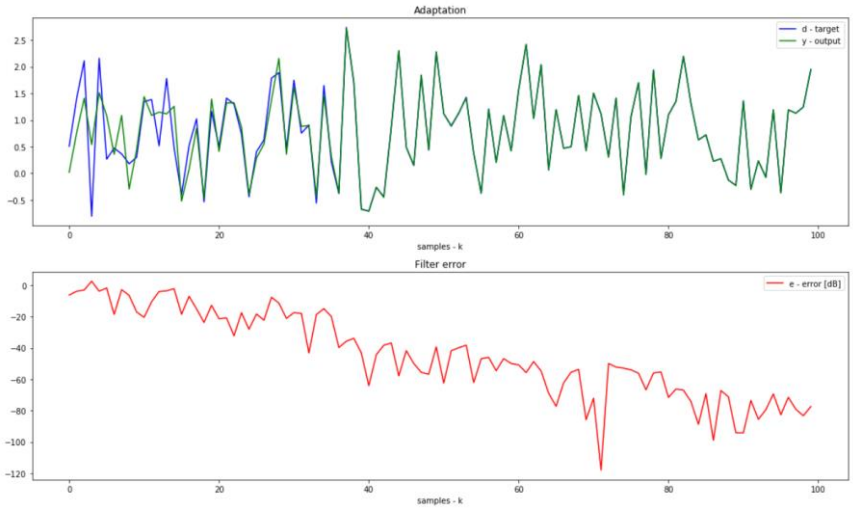
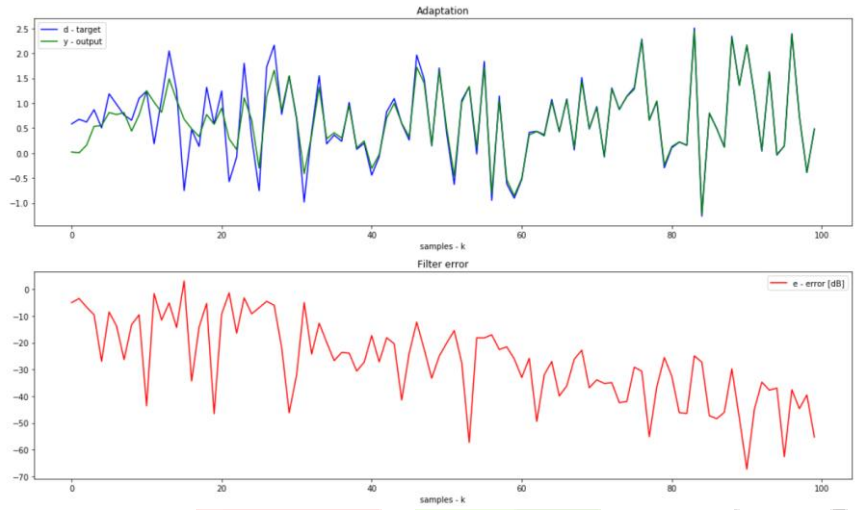


Figure (9b) NLMS with 'mu' = 1



For both the adaptive filters LMS and NLMS with same input vector size of three, the number of samples taken as hundred and the learning rate of 1 we obtained the above results as shown in Fig. 9(a) and Fig. 9(b). Both the filters generated an accuracy score of approximated 99 percentages. As seen in the graphs, the LMS filter was able to meet the target at about after twenty-five to thirty iterations than the NLMS which required more iterations with the same learning rate. The log values of the error rates lie within the range of [-67.37 to 3.12] for NLMS and [-118.020 to 2.57] for LMS in dB.

Figure 9(c) RLS filter with 'mu' = 0.5

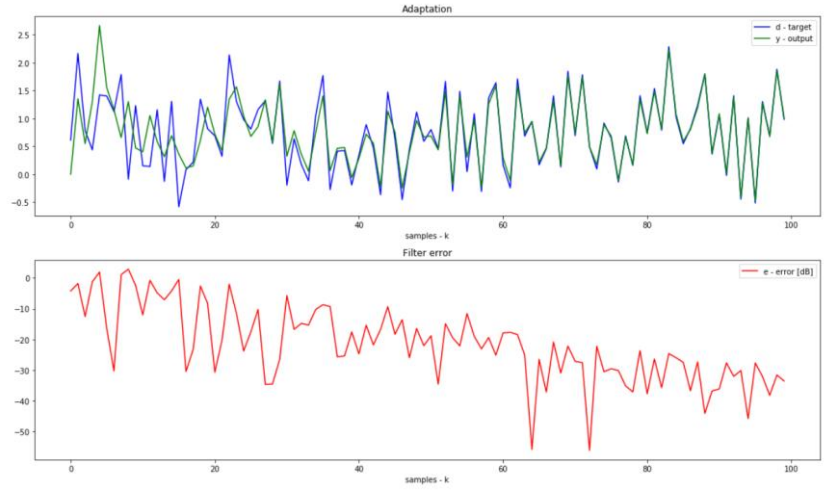
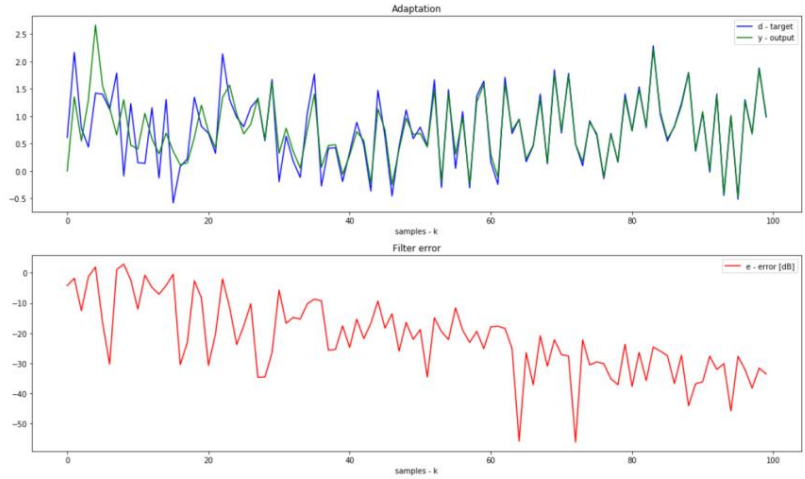


Figure 9(d) Affine Projection 'mu' = 0.5



It can be observed in Fig. 9(c) The RLS adaptive filter performed the best after about 25 iterations and did match the target effectively as soon as the weights got updated more efficiently. The order of filter is three and number of samples are hundred with learning rate as 0.5. With Affine projection an initial offset covariance of 0.001, filter length 4 and order of filter 5 we obtained similar results as seen in Fig. 9(d). Both the filters had an accuracy of more than 99 percentages for matching the target values.

3.5 Kernel Adaptive Filters

This experimental procedure involves the implementation of LMS and RLS algorithms along with the kernel versions of the same. In the kernel versions the input data is transformed into a high dimensional space data. This high dimensional linear feature space maps the signal, and then a non-linear function is applied on it. This is estimated as the sum of kernels. The kernel versions are operated upon without weights as it works on the sum of kernels of previously operated data. This application has mainly these advantages:

1) Convex loss functions, with no local minima.
2) Very feasible to implement with a very less computational complexity.
3) Less time consuming.

In this experiment, a Gaussian Kernel is used with an appropriate step size and a cross validation approach is implemented for analysis. The Gaussian Kernels is a blurred version of the Heavyside stepfunction [12]. For the implementation of the KLMS algorithms a dataset is used containing two variables of speed and power with n samples. The data is loaded in the csv and then normalized to max of 100000. After the initialization of the kernel adaptive filter we define the lower and upper limits as 0 and 1000 with a step size of a hundred. Then the input parameters are passed in the filter and we generate the predictive kernel estimated errors for a thousand iterations.

Similar methodology is used for KRLS. In KRLS the regularization parameters are lowered for more data. The parameters are defined as 0.1, 0.01 and 0.001. The step size and the number of iterations remain the same. The following plots as shown in Fig. 10(a) and 10(b) are obtained for the regularization parameters for KRLS and KRLS algorithms respectively. The simplest form of the RLS algorithm involves minimizing of ‘i’ as the sum of the squared errors.

Figure 10(a). Plot of KLMS algorithm with regularization parameters 0.1 and 0.01.

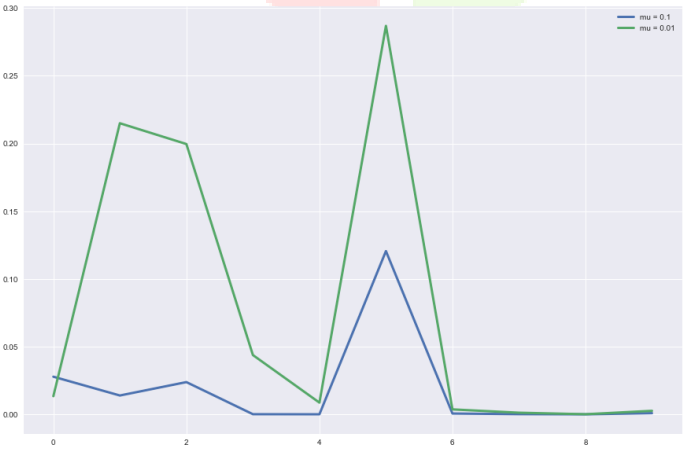
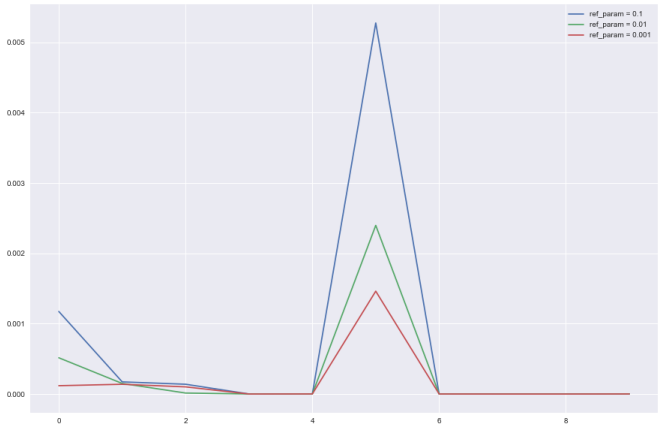


Figure 10(b). Plot of KRLS algorithm with reg parameters 0.1,0.01 and 0.001 respectively



The plots in the figures give us the brief idea of how different regularization parameters effect the designed LMS and RLS filters in the kernel versions. Both these approaches make the use of the Mercer kernel function. [13] It helps in giving a sample by sample update to the kernel with the proceeding iterations. The non-linear prediction problems can then be optimized easily using these algorithms with predictive linear regression machine learning algorithms.

IV. Results and Conclusions

Both the algorithms LMS and RLS in adaptive filters are suitable for use in the respective arenas whilst using adaptive filters. The results obtained while performing of the experiments conclude that the LMS algorithm is very efficient in noise cancellation or echo cancellation purposes while transmission and receiving of signal data.

From the Experiments performed it is observed that specifying the order of filter and the learning rate are very important aspects of the filter in terms of efficiency and stability of the filter. The learning rates based on the stability factor can be summarized as in the Table (4.1).

Table (4.1). Range of 'mu' and optimal stability

Filter Type	Stability Factor	Range of 'mu' selection	Optimisation
LMS	$\lvert 1 - \mu . \lvert\lvert x(k) \rvert\rvert^2 \rvert < 1$	0.1 to 0.00001	Should be set to a very small number for most cases.
NLMS	$0 \leq \mu \leq 2 + \frac{2\varepsilon}{\lvert\lvert x(k) \rvert\rvert^2}$	0 to 2 1 is more optimal	Optimal values are case specific.
RLS	$0 \leq \lambda \leq 1$	0.98 to 1 $\lambda = 1$ is known as growing window RLS	Smaller the value, more sensitivity to more recent values.
AP	$0 \leq \mu \leq 2$	0 to 1	For noise enhancement smaller values are preferred.

The kernel versions are very appropriate to be utilized as the computational complexity gets highly reduced and the weight calculation dependency gets eradicated completely. The kernel versions operate on the previously iterated values of the sample data and updates continuously as per the defined learning rate and thereby improves the accuracy of MSE prediction and obtaining the desired signal.

V. Applications and Future Scope

The adaptive filters are mainly utilised only for noise and echo cancellation in signal processing and in wireless communication system. This can be further enhanced by using the ANN and deep learning methods in which the algorithm performs based on the results of the previously generated errors and stores them for future reference. As implemented in the experiment, different learning rates have different effects on the filter parameters making them application specific. Adaptive filters can also be utilised in advanced FPGA and VHDL designed systems for signal retrievals and digital analytics systems.

Expansion of the adaptive filters includes the use of deep learning and neural networks or GANs (General Adversarial Networks) for prediction of the future values accurately, and calculate the probability of recurrences of errors and get more updated respectively.

REFERENCES

[1] Adaptive filters: Available at: Mehendale, Ninad & khandhar, jay & vora, ruchik & Shah, Dhruvil. (2012). ADAPTIVE FILTER IMPLEMENTATION USING FPGA. Technofocus. 2. 133.

[2] Wiener, Norbert (1949). Extrapolation, Interpolation, and Smoothing of Stationary Time Series. New York: Wiley. ISBN 978-0-262-73005-1.

[3] E. C. Ifeachor and B. W. Jervis, Digital Signal Procesing, A Practical Approach, Prentice Hall, 2002

[4] Widrow and Hoff lms filter. https://isl.stanford.edu/~widrow/papers/c1960adaptiveswitching.pdf

[5] LMS filter. Available at: Ali H Sayed. Fundamentals of adaptive filtering. John Wiley & Sons, 2003.

[6] RLS filter. Available at: Ali H Sayed and Thomas Kailath. Recursive least-squares adaptive filters. The Digital Signal Processing Handbook, pages 21–1, 1998.

[7] Affine projection. Available at: Alberto Gonzalez, Miguel Ferrer, Felix Albu, and Maria de Diego. Affine projection algorithms: evolution to smart and fast algorithms and applications. In Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European, 1965–1969. IEEE, 2012.

[8] Padasip library. Available at: https://matousc89.github.io/padasip/index.html#

[9] Adaptfilt library. Availble at: https://pypi.org/project/adaptfilt/#description

[10] Kernel Adaptive Filtering Weifeng Liu, Jos C. Prncipe, Simon Haykin."Kernel Adaptive Filtering", Wiley, 2010

[11] Liu. "Kernel Least-Mean-Square Algorithm",Kernel Adaptive Filtering, 02/12/2010.

[12] Gaussian Kernel. http://pages.stat.wisc.edu/~mchung/teaching/MIA/reading/diffusion.gaussian.kernel.pdf.pdf

[13] Mercers Kernel Function. Available at: https://ai-master.gitbooks.io/kernel-svm/content/mercers-theorem.html