International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 12 (2020) pp. 297-309
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Received: 6 April, 2018; Accepted: 5 April, 2019; Published: 7 August 2020

Wind Turbine Performance Monitoring Using Advanced Adaptive Filters

Ashima Arora

Computer Sci. and engg.,
Maulana Azad National Institute of Technology,
Bhopal, M.P., India-462003
arora.ashima1993@gmail.com

Abstract: **This paper analyze the role of adaptive filters for monitoring the performance of the wind turbine. Advanced adaptive filters are implemented using various algorithms like standard least mean square, normalized least mean square, generalized normalized gradient descent, weighted least mean square, recursive least square, and affine projection algorithms. A Comparative analysis is done on the basis of parameters like mean absolute error, root mean square error, R-squared (R^2) score to determine which adaptive filter is suitable for estimating wind-power generation. Additional parameters such as convergence rate, computational complexity, and stability of the system are also analyzed to monitor the performance of each filter. Factual power data sets of two different sites are taken from resource file of National Renewable Energy Laboratory. The comparison is done under similar assumptions on both the data sets in order to extract accurate filter performance.**

Keywords: Wind power generation, Adaptive filters, Affine projection algorithm, Generalized normalized gradient descent algorithm, Least Mean Square, Normalized least mean square, Recursive least square algorithm

I. Introduction

Wind energy has proved itself an imperative resource in power generation. Nowadays, high share of wind energy is generated from wind turbines in the energy sectors [1]. There are various factors which influence the efficiency of the wind turbine in wind-power generation. Factors such as altitude at which the turbine is placed, the layout of the wind farm, i.e., air density and temperature of the farm, blade aerodynamics of the turbine, integration of generated power into grids [2]. This paper focuses on monitoring the performance of wind turbine on the basis of total power generated from the turbine. Monitoring of generated output power is essential because it may inhere a lot of uncertainties and deviations. Consequently, power grids' performance may be highly affected. Moreover, energy management systems may also expose themselves to some serious issues [3]. Subsequently, it becomes necessary to effectively integrate wind-power into the power frameworks. Once the integration is completed, the performance of the turbine is monitored for operational management of wind energy.

One of the criteria to analyze the performance is to evaluate the respective power curve generated by the wind turbine [4]. Manufacturers often supply their theoretical power curves assuming ideal meteorological and geomorphological conditions. These theoretical power curves only offer nominal wind-power readings and do not cover all aspects of the practical environment. In the real world, wind turbines are never subjected to ideal conditions which result in a substantial difference between the factual power curves and the theoretical ones. This difference in the power curves is due to various factors like wind direction, temperature, pressure, precipitation, wind velocity distribution, altitudes etc. These may act independently or in combination with each other effects during wind energy generation [5]. To enhance the accuracy of the generated output power, several statistical methods were used to fit the empirical power curve of a wind turbine and evaluate total power generated. Methods like polynomial regression, locally weighted polynomial regression [6], spline regression, and models based on probabilistic distributions, logistic distributions [7] were used to determine power curve. But these models were restricted by their nature and did not solve the problem of non-linearity in the data sets. This results in failure and estimated power curves were not as close as observed data subject to the smoothness of the fit [8]. To overcome these limitations and to enhance the accuracy and quality of the power curve other models like fuzzy logic methods, fuzzy cluster center models [9], neural networks [10], feed forward multi-layer perceptron [11], the k-nearest neighbor [12] were preferred as they precisely model a wide range of possible shapes of power curves as stated by authors in [7]. In research work of M. Morshedizadeh et al. [13], authors proposed a model, which preferred the blending of two methods: fuzzy logic and neural networks to model the power curves. But these methods also face drawback as they fail to adapt to changing environment. Practically, there is no one technique that could produce least error and overrule all others upon every single possible observation which are obtained from various wind turbines. It is observed that a specific method might be restricted to excel only on few data sets, but on alternative data sets, other methods might be more applicatory [8]. Moreover, the computational complexity is relatively high when statistical methods mentioned above were used to fit the power curve of a wind turbine. So, another approach called adaptive filters can be used to asses error in wind-power generation. Earlier applications of adaptive filters were limited to signal processing, channel equalization and identification, adaptive feedback cancellation, signal control, prediction etc. This paper extends the use of adaptive filters for estimating wind-power generation. Because adaptive filters serve best when applied in a time-varying environment. This feature of adaptive filter is known as *online adaptation*. A wind turbine is always subjected to an environment which changes with respect to time. Wind speed, wind direction, air temperature, pressure etc. are various parameters which change in accordance with time. Thus, adaptive filters applications can be used for estimation purposes. In addition, online adaptation is nowadays a trending method that is useful for batch machine learning algorithms. Its application is specially used to sample large databases stochastically and finding quick solution subject to least error. Determining relevant algorithms which offer a better trade-off between computational complexity, convergence rate, and memory requirement is a need for today [14]. The adaptive filter is one such concept that is preferred for sampling large databases over statistical methods described above [15].

An adaptive filter is a time-varying filter which modifies its coefficients in order to optimize a cost function in accordance with the unknown environment. It is additionally used to fulfill some predetermined optimization paradigms. These filters process the input signals and are concerned with following three major parameters: the design of the filtering structure, analyzing the input-output signals, and implementing a system which changes its structure in accordance with incoming signals [16]. It works in online mode, i.e., input data arrive continuously with respect to time. This scheme is known as *time iterative schemes*. These online algorithms work on single data point at a time. Unlike batch processing methods which process the whole block of data as a single unit. Another advantage of using online algorithms is that they do not require prior knowledge of the data set upon which calculation of error takes place. At the same time, the computational complexity of these algorithms is low as discussed in section III-C.

The main attribute of the adaptive filter is that it can automatically adapt itself to scenarios like when there is a change in the environment or when requirements of the existing system change with respect to time. Also, these filters can be trained to perform particular filtering tasks or decision-making tasks based on some set of training guidelines or updating equations [17]. Since these algorithms work in an online mode they are able to track even smallest variations involved during the process. These abilities make adaptive filters a compelling device for control operations and signal processing. In this paper, we focus on online learning techniques for estimating the parameters of the unknown system and then monitoring the performance of the adaptive filters. In the first stage, we will implement adaptive filters using each technique and then evaluate the performance of each adaptive filter on various parameters. Section II of this paper describes various online techniques like - Standard Least Mean Square Algorithm (SLMS), Normalized Least Mean Square Algorithm (NLMS), Weighted Least Mean Square Algorithm (WLMS), Generalized Normalized Gradient Descent Algorithm (GNGD), Recursive Least Square Algorithm (RLS) and Affine Projection Algorithm (APA).

II. Advanced Adaptive Filters For Wind-Power Estimation

There are numerous cases when fixed specifications of the system are unknown or time-invariant filters cannot satisfy the given specifications. Under such circumstances, adaptive filters have proved to be an asset. In this paper, different adaptive filters are implemented based on algorithms mentioned in section I [18].

An adaptive filter is a data-processing device that enables to find the correlation between two signals continuously in an iterative manner. Adaptive filters can be one of the following: a set of program instructions or set of logic operations which run on arithmetical processing devices. Additionally, these can be implemented on field - programmable gate array or a custom VLSI integrated circuit. Purpose of the adaptive filter is to assess the future values with respect to the past values of the input signals and make the adaptive filter comparable to the unknown system[19].

Four basic aspects define an adaptive filter. *The signals* that is fed to the filtering structure for producing the output signal. *The filtering structure* that evaluates all the output signals of the filter on the basis of respective the input signals; *the parameters* which change themselves with every iteration in order to recalibrate the filter's input-output relationship; *the adaptive algorithm* that describes how the parameters will adjust or update themselves from one time to time during intervals. Each adaptive filter consists of different sets of parameters and algorithms. When a specific adaptive filtering structure is selected, the number and type of parameters associated with it can be modified as per requirement. The adaptive algorithm is also used as a form of an optimizer that minimizes the error for a particular data set [20]. Each adaptive filter is associated with a rate at which the coefficients of the filter converge to their optimized values. This rate is known as *speed of convergence*. Speed of convergence varies for each filter depending on the algorithm used during the implementation of the adaptive filter. Speed of convergence depends on various factors such as amplitude and correlation statistics of the input-output signals, initial weights of the filter, filter length, step size etc. Theoretically, we can assume that the speed of convergence increases as the step-size increases. Step-size cannot be infinitely increased to achieve higher convergence. There is upper bound imposed to the step-size that is half the maximum value that is required to achieve the stability of the system. Another parameter that affects the speed of convergence is the length of the filter *L*. Speed of convergence is inversely proportional to the length of the filter. As the length of the filter rises, speed of convergence decreases [21]. Therefore, L should be chosen as short as possible but at the same time long enough to precisely model the unknown system.

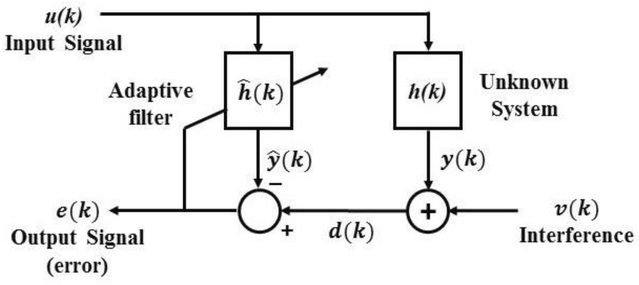


Figure. 1: A typical Block Diagram Of Adaptive

Proper selection of all these parameters is an utmost important task while implementing an adaptive filter. The elementary structure of an adaptive filter is presented in fig. 1. Parameters in fig. 1 are defined as follows: $\hat{u}(k) = [u(k), u(k-1), \ldots, u(k-p+1)]^T$ is the input signal, $h(k) = [h_0(k), h_1(k), \ldots, h_{p-1}(k)]^T$; $h(k) \in c^P$ is the unknown system, $\hat{h}(k)$ is the estimated filter which estimates the filter after k samples, k is the number of current input samples, $(.)^T$ is the (Hermitian or conjugate transpose), $e(k)$ is the error signal, p is the order of the filter. Weights $w(k) = w_1(k), w_2(k), \ldots, w_n(k)$ corresponds to adaptive filter coefficient vector of size $N\, x\, 1$. Using these parameters various adaptive algorithms adjust its coefficients and bring the system as close as possible to the unknown system. This section presents the general filtering structure of advanced adaptive filters along with their implementation for updating the weights after every iteration. Thereupon, determining the error form for each adaptive filter.

A. Standard Least Mean Square Algorithm (SLMS)

The standard LMS algorithm is a class of linear adaptive filters. These filters are utilized to produce the desired filter after it evaluates the minimum mean square of the error signal. The desired filter is produced by finding the filter coefficients that identify even unsubstantial variation in the mean square error signal. The filter coefficients of the algorithm are modified in order to acquire minimum cost function[22]. In general, SLMS algorithm performs two basic processes *filtering process* and *adaptive process*. Role of the filtering process is to compute the output signal of the filter in accordance with input signals. Once the output signal is generated, it is then compared with the desired response to estimate net error. The aim is to automatically adjust the coefficients of the parameters in accordance with the estimated error. Using the parameters mentioned in section II, SLMS performs the following operations in order to update the coefficients of an adaptive filter: Firstly, the output signal $\hat{y}(k)$ from the adaptive filter is calculated. Is Using the output signal, the error signal $e(k)$ is calculated using the following equation for $k = 0,1,2,\ldots,k_{max}$:

$$e(k) = d(k) - \hat{y}(k) \qquad (1)$$

where, $\hat{y}(k) = \hat{w}(k).\hat{u}^T(k)$ and $d(k) = y(k) + v(k)$ Is the desired signal. Then, following equation is used to update the filter coefficients over the iteration:

$$\hat{w}(k+1) = \hat{w}(k) + \mu.e(k).\hat{u}(k) \qquad (2)$$

where, $\hat{w}(k+1)$ represents the weights vector at instance $+1$, $\hat{w}(k)$, is the coefficients vector of the filter at the instant k, and $\hat{u}(k)$ is the input vector stored in the delayed line of the filter, $e(k)$ corresponds to the error signal of the filter, μ is the step-size of the adaptive filter (also known as the convergence factor) and referred to as *Widrow-Hoff rule* named after its founders[23]. An increment of 1 is seen in the time-index of the coefficient vector after it is updated, it states that sample-by-sample update is performed over iterations. The convergence factor μ determines how close the algorithm has converged towards the solution. Inverse relation exists between the convergence factor μ and the minimal error. On the other hand, this factor μ is directly proportional to the convergence speed [24].

In LMS, μ is constant so whenever new input data comes it tries to update the estimates in order to produce small values of error. As observed, exact values of estimation cannot be used while applying LMS algorithms. That is why optimal weights are not achievable in an absolute sense. However, LMS algorithms converge to mean value. This property helps to achieve optimal weights, even-though, the change in weights is very small. When the change in weights is large, a problem of variance arises. Then convergence towards mean value can be misleading. Thus, proper selection of variable parameter is necessary. An upper bound and lower bound is given to step-size μ and is called *condition of convergence*, defined as follow:

$$0 < \mu < \frac{2}{\lambda_{max}} \qquad (3)$$

where λ_{max} is the maximum eigenvalue in the auto-correlation matrix which contains only non-negative eigen-values. The algorithm becomes unstable if μ does not satisfy the relation in (3). Also, by analyzing the relation in (3) we can conclude that inverse relationships exist between the convergence speed of the algorithm and the eigen value spread of the correlation matrix.

Therefore, slow rate of convergence is seen when the eigen-values of the correlation matrix are widespread. Eigen value spread function is evaluated by taking the ratio of the largest eigen value to the smallest eigen value. This eigen value spread function basically denotes required time for the k_{th} mode to reach $1/e$ of its initial value [25]. The choice of μ is the deciding factor for calculating convergence speed. When μ is small then convergence takes place slowly. When the value of μ is large then, a fast convergence of the algorithm is seen. However, upper bound of the μ should always be taken into consideration as the algorithm may tend to move to an unstable state if μ is high. To improvise the stability of the algorithm, max achievable convergence speed is given by:

$$\mu = \frac{2}{\lambda_{max} + \lambda_{min}} \qquad (4)$$

where λ_{min} and λ_{max} are the smallest and largest eigen values of the function respectively. Hence, faster convergence is achieved when λ_{max} is close to or equal to λ_{min}. This can also be represented in the form :

$$\pm\frac{\rho-1}{\rho+1} \quad (5)$$

where, $\rho = \frac{\lambda_{max}}{\lambda_{min}}$. Thus, maximum convergence speed is solely dependent on the spread of the eigen values of the covariance matrix. When it comes to storage requirement it takes small memory footprint as it stores the present weight of the filter only. So, this makes LMS compatible for use in non-stationary environments as well as online settings [26]. Its faster computation gives reasonable results. Moreover, it can be used for more complex models where there are more error fluctuations.

B. Normalized Least Mean Square Algorithm (NLMS)

To eliminate the drawbacks of standard least mean square algorithm (SLMS), NLMS algorithm comes into play. SLMS algorithm is easily affected whenever its input signals are calibrated. This causes difficulty in choosing appropriate learning rate µ. NLMS is the mutated form of the standard LMS algorithm [27]. When the new input data arrives, the parameters of an adaptive system are distributed in a least possible manner. There are two integral aspects of the NLMS algorithm. Firstly, if large amount of fluctuations are observed in the power levels of the input signals, its repercussions are compensated at the adaptation level. Secondly, if input signals have large input vector length its effect is compensated by decreasing the step size of the adaptive algorithm [28]. For a given input signal $\hat{u}(k) = [u(k), u(k-1), \ldots, u(k-p+1)]^T$ where p is the order of the filter; this algorithm uses the following equations to update its coefficients:

$$\hat{w}(k+1) = \hat{w}(k) + \mu.e(k)\frac{\hat{u}(k)}{\left[||\hat{u}(k)||\right]^2} \quad (6)$$

Numerical difficulties may arise if $\hat{u}(k)$ is very close to zero, to overcome this a constant $0 < a < 1$ is used:

$$\hat{w}(k+1) = \hat{w}(k) + \frac{\mu}{a + \left[||\hat{u}(k)||\right]^2}.e(k)\hat{u}(k) \quad (7)$$

In NLMS, μ the adaptation constant is dimensionless. On the other hand in LMS, the adaptation constant has the dimensioning of an inverse power.

Substituting $\mu(k) = \frac{\mu}{a+\left[||\hat{u}(k)||\right]^2}$ in (7), the above equation becomes:

$$\hat{w}(k+1) = \hat{w}(k) + \mu(k).e(k)\hat{u}(k) \quad (8)$$

Here, (8) illustrates that the NLMS algorithm becomes the same as the SLMS algorithm. The only differentiable factor here is time-varying step size $\mu(k)$. It is also known as data-dependent adaptation step size. Proper selection of μ is necessary to get best results. Theoretically, it is also observed that proper selection of μ while implementing NLMS filter can show faster convergence than LMS adaptive filter. Hence, lower bound and upper bound is given to the dimensionless constant $\mu \in (0,2)$. Such modifications and constraints in step size and weight vector help in achieving high convergence speed and thus, making NLMS flexible to adaptation. With all these advantages NLMS also have some drawbacks. For instance, it shows slow convergence for colored input signals. To ameliorate these issues a class of equivalent algorithms has been proposed such as the Generalized Normalized Gradient Descent Algorithm (GNGD), the affine projection algorithm (APA) [29] which is briefly discussed in section II-C and II-F.

C. Generalized Normalized Gradient Descent Algorithm (GNGD)

GNGD algorithm is the abbreviation of a generalized normalized gradient descent algorithm which represents a special type of variable regularized NLMS. This algorithm was proposed to overcome the limitations of least mean square based algorithms. Limitations like LMS based algorithms require prior knowledge of the eigen values of the correlation matrix of the input signal in order to show good performance. But in real world scenario it is not feasible to keep record of such information. Secondly, NLMS shows slow convergence when subjected to colored input signals. GNGD is one such algorithm which when given such input signal sit shows fast convergence and small steady-state mis-adjustments when regulated in stationary backgrounds [30]. On the other hand in non-stationary background GNGD adjusts its learning rate as per the subjected input signal and try to achieve better performance than other least mean square based algorithms. GNGD algorithm has additional gradient adaptive term in the learning rate on the contrary with NLMS's learning rate equation. Because of this additional term in GNGD, this algorithm is able to adapt its learning rate with the dynamics of the input signal vector. GNGD in its general form is defined as follows[31]:

$$\hat{w}(k+1) = \hat{w}(k) + \mu(k).e(k)\hat{u}(k) \quad (9)$$

where, $e(k) = d(k) - w^T(k)\hat{u}(k)$ is the estimation error for time index k, $d(k)$ is the desired signal, $w(k+1)$ and $w(k)$ are the adaptive filter coefficients vectors at time $k+1$ and k respectively. GNGD algorithm has modified $\mu(k)$ defined as:

$$\mu(k) = \frac{\mu}{\left[||\hat{u}(k)||\right]^2 + \delta(k)} \quad (10)$$

where, $\delta(k)$is the regularization factor which is added to avoid divisions by zero or very small numbers, µ is the normalized step-size with value ranging from 0 to 2, i.e., $0 < \mu < 2$, $||\hat{u}(k)||$ is a vector containing the length of k recent samples of the input signal. Now using stochastic gradient method $\delta(k)$ is made gradient adaptive by following equation:

$$\delta(k) = \delta(k-1) - \rho\nabla_{\delta(k-1)}J(k) \quad (11)$$

where, ρ is the step size parameter for the adaptation of the regularization factor with range $0 < \rho < 1$ and $J(k)$ is the cost defined as follows:

$$J(k) = e^2(k) \quad (12)$$

To evaluate the value of $\rho\nabla_{\delta(k-1)}J(k)$, chain rule is used as follows[30]:

$$\frac{\partial J(k)}{\partial \delta(k-1)} = \frac{\partial J(k)}{\partial e(k)} \frac{\partial e(k)}{\partial y(k)} \frac{\partial y(k)}{\partial \widehat{w}(k)} \frac{\partial \widehat{w}(k)}{\partial \mu(k-1)} \frac{\partial \mu(k-1)}{\partial \delta(k-1)}$$

$$= \frac{e(k)e(k-1)u^H(k)u(k-1)}{\left(||\mu(k-1)||^2 + \delta(k-1)\right)^2} \qquad (13)$$

Now substituting the value from equation (13) to equation (11). Updated equation of $\delta(k)$ is formulated as:

$$\delta(k) = \delta(k-1) - \rho\, u_0 \frac{e(k)e(k-1)\mu^H(k)u(k-1)}{\left(||\mu(k-1)||^2 + \delta(k-1)\right)^2} \quad (14)$$

Equation(14) uses previous values of $e(k-1)$ and $u(k-1)$ so initialization of these parameters should be properly accounted at time index $k = 0$. Also an assumption follows that for $k \le 0$,value of input signal $u(k) = 0$. Thus, initial input signal vector $u(0) = 0$. Another assumption that follows is $\delta(1) = \delta(0)$. Due to this assumption, updation of the regularization factor δ(k) is postponed till the second step of the algorithm. As a caution, the value of δ(k) should be a very small positive number in order to avoid division by zero during first few iterations as input signal may not be present. The initial value for $e(0)$ can be set to zero to simplify the initial calculations of the regularization factor. Another highlighting feature of GNGD algorithm is that it does not need any prior knowledge of the correlation matrix of the input signal. Also, it robustly shows variations in the initialization of its parameters. This robustness and improved stability make this algorithm ideal for working in a non-stationary environment. Although when the computational aspect is considered, GNGD shows high computational complexity with respect to other least mean square based algorithms due to obvious reasons. Reasons like additional stabilization term called regularization factor and multiplication calculations of $u^H(k)u(k-1)$ [32]. Nevertheless, these product evaluations can be computed in a recursive manner that can help decrease computational complexity up to a point.

D. *Weighted Least Square Algorithm (WLMS)*

When adaptive filter is produced using least mean squares algorithm, an assumption follows that variance in errors is constant over the time. This is known as homoscedasticity. In other words, each value in the data-set is treated with equal weight. Despite the fact that some values of variables may be more likely accountable while calculating error. Thus, weighted least square algorithm is used when there is voilation in the assumption of constant variance in the errors. The term heteroscedasticity is used which represents non-constant variance. That is, few values in the dataset has more weightage with respect to other values in the same dataset [33]. A normally distributed weight matrix with mean vector 0 and non-constant variance-co-variance is shown below:

$$\begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{bmatrix}$$

Assuming $Var(\varepsilon_i) = \frac{\sigma_i^2}{w_i}$ for known $w_i, 1 < i < n$. Let matrix W be a diagonal matrix of N x N containing weights:

$$\begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix}$$

Now, the equation of weighted least square is given below:

$$y_i = \mu \cdot x_i^T + \varepsilon_i \qquad (15)$$

where ε_i is assumed to be non-constant variance-co-variance matrix and normally distributed with mean vector 0, μ represents the weighted least square estimate which is calculated by minimizing the error ε, y is the responses that reflect in the data set. To proceed with the minimization, let:

$$y_i^* = \sqrt{w_i}y_i \quad \& \quad x_i^* = \sqrt{w_i}x_i \qquad (16)$$

Then eq. (15) can be written as:

$$y_i^* = x_i^{*T}\mu + \sqrt{w_i}\,\varepsilon_i \qquad (17)$$

where $Var(\sqrt{w_i}\,\varepsilon_i) = w_i Var(\varepsilon_i) = \sigma^2$ To evaluate the weighted least square estimate minimize the error:

$$S = \sum_{i=1}^{n} w_i (y_i - x_i^T \mu)^2 \qquad (18)$$

In a matrix form value of the weighted least square estimator of μ for overall model is defined as:

$$(Y - \mu \cdot X)^T W (Y - \mu \cdot X) \qquad (19)$$

where, W is a diagonal matrix mentioned above. Now, taking the derivative with respect to μ, the solution is:

$$\mu = (X^T W X)^{-1} X^T W Y \qquad (20)$$

According to equation (20), inverse relation exists between weights and error variance which clearly reflects the information in particular set of observations. Therefore, an observation with large weights will have small error variance since it contains relatively more information than an observation with small weights and large error variance. It is useful in cases where some observations need to be omitted. This can be simply done by setting its weight to zero. Moreover, it helps to suppress the outliers in order to curtail their impact on the filter. In small datasets, WLS works best in retrieving maximum information. It is the only method that can be applied to datasets where data points are of varying quality. However, application of this requires the exact knowledge of the weights, which is not always feasible [34]. Estimating the weights may give unpredictable results, especially in small data sets. So, this technique should only be used when the weights estimates are accurate and precise.

E. *Recursive Least Squares (RLS) Algorithm*

RLS is also a class of adaptive filter. RLS filters recursively find the coefficients relating to the input signals. The objective of this algorithm is to minimize the cost function. This is done by selecting the filter coefficients judiciously and updating the filter with new incoming data. For updating the old estimates, recommended initial conditions are taken into consideration along with the information contained in new data samples. The RLS algorithms work best in time-varying environments or non-stationary environment but this algorithm leads to higher computational complexity and stability problems. In RLS, the input vector is simultaneously given to traversal filter unit as well as to the adaptive weight control mechanism unit. Once the output is generated, it is matched with the desired response to evaluate the error. This error is then fed into adaptive weight control mechanism unit where it is recursively called till stability state is achieved [35].

There are two main reasons to use RLS: a) when the number of variables in the linear system exceeds the number of observations, under these circumstances the ordinary least-squares problem becomes ill-posed which makes it impossible to fit as infinitely many solutions are obtained for optimization. RLS algorithm enables to overcome this limitation by introducing further constraints that uniquely determine the solution. b) Whenever learned model suffers from poor generalization RLS comes into play. Poor generalization can occur even when the number of variables does not exceed the number of observations. RLS is one such algorithm which put constraints at training time in order improve the generalization ability of the model [36]. Although, RLS algorithms do not put constraints on the input data structure. As a result, the computational complexity of the RLS algorithms increases and is N^2 per iteration where N is the size of the data matrix. To solve RLS, a weighting factor is introduced to the sum of errors-squares definition:

$$\xi(k)=\sum_{i=1}^{k}\beta(k,i)|e(i)|^2 \qquad (21)$$

where, weighting factor has the property $0<\beta\leq 1$ for $i=1,2,\dots,k$. This weighting factor is used to ensure that less weight is given to older error samples, so that statistical variations in the data-set can be observed more easily when the filter operates in the non-stationary environment [37].In other words, more emphasis is given to newer input sample. The most familiar weighting factor is called exponential weighting factor or forgetting factor. It is defined as $\beta(k,i)=\lambda^{k-i}$ for $i=1,2,\dots,k$ where, λ is a positive constant, $0<\lambda<1$. The filter tap weight vector for this algorithm is updated using following equations:

$$\widehat{w}(k)=\widehat{w}(k-1)+K(k)\cdot\hat{e}_{k-1}(k) \qquad (22)$$

$$K(k)=\frac{X(k)}{\lambda+u^T(k)X(k)} \qquad (23)$$

$$X(k)=\widehat{w}_{\lambda}^{-1}(k-1)u(k) \qquad (24)$$

where, λ is a small positive constant whose value is close to 1 but not equal to 1. Eq.(23) and (24) are the intermediate gain vectors which compute tap weights. Applying the filter tap weights of above mentioned iteration and the current input vector, the output of the filter is calculated in (25) and (26).

$$\hat{y}_{k-1}(k)=\widehat{w}^T(k-1)u(k) \qquad (25)$$

$$\hat{e}_{k-1}(k)=d(k)-\hat{y}_{k-1}(k) \qquad (26)$$

where $\hat{y}_{k-1}(k)$ represents output of previous samples and $\hat{e}_{k-1}(k)$ is the error for instance $(k-1)$. RLS algorithm requires higher memory necessities during its execution as it uses the estimate of previous sample of error signals, output signals, and filter weights. Another highlighting feature of RLS algorithm is that, when this algorithm is implemented it enables evenly distribution of computational load in each iteration. Also, RLS algorithm is not suitable for online filtering due to time consuming computations of inverse matrix least squares methods. This is one of the limitation of implementing adaptive filter using RLS algorithm.

F. Affine Projection Algorithm (APA)

Affine projection algorithms are also introduced to overcome the drawbacks of least mean square based algorithms [38]. These algorithms belong to data reusing family, i.e., past data is used at every instant of time. This reusability of past data ensure that the algorithm learns and adapt itself at a faster rate. Affine Projection algorithm is also recognized as the extensive version of NLMS discussed in section II-B. NLMS is scrutinized as one-dimensional projection whereas APA projections are made in multiple dimensions. In other words, the filter update equation of affine projection uses *N* vectors of input signal instead of using uni-vector input signal. This is called *N* projection order. The greater value of *N* , i.e., high projection order results in the high convergence speed of the weight vector. On the contrary, to least Mean Square algorithms which show slow convergence speed. Thus, the performance of LMS based adaptive algorithms can be improvised using the applications of affine projection algorithms. Especially in cases where input-data is highly correlated [29]. Projection order *N* is also one of the significant parameters for determining the computational cost of the APA.

In this algorithm, high convergence speed can be achieved but at the same time, the computational complexity of the algorithm also increases. Computational cost and its steady-state behavior are solely dependent on the initial configuration of the system, especially its projection order *N* . Hence, proper selection of projection is important to attain meaningful computational savings. Many versions of affine projection have been implemented so far. Algorithms such as *Fast APA, Gauss-Seidel pseudo APA, Dichotomous Coordinate Descent APA, Variable Order APA* [39]. Each version is different from one another and contains its own characteristic parameters. One such parameter for creating these versions is the weight updating equation which uses multiple or delayed input signal vectors. In the implementation of APA, *N* input data vectors used for updating filter weights which are not necessarily the recent *N* input signals. Thus, using various combinations of order *N* of input signals different versions of affine projection algorithms can be implemented[40]. Algorithms such as the partial rank affine projection algorithm abbreviated as PRA [41], the NLMS with orthogonal correction factors abbreviated as NLMS-OCF [42], affine projection algorithm with regularization abbreviated as R-APA [43], Levenberg Marquardt regularized APA abbreviated as LMR- APA [44] etc.

One of the most commonly used version is standard affine projection algorithms abbreviated as APA which is discussed in this paper. Two basic equations are used to define APA algorithm as follows [38]:

$$e_k = s_k - U_k^T . w_{k-1} \tag{27}$$

$$w_k = w_{k-1} + \mu . U_k [U_k^T . U_k]^{-1} . e_k \tag{28}$$

where, superscript T denotes transpose, μ is the variable parameter which will be optimized in section III-C, e_k is the error vector of length N and s_k is system output, w_k is the adaptive tap weight vector of length k, U_k is the excitation signal matrix of size $L \times N$ where k is the index and has the following structure:

$$U_k = \left[u_k, u_{k-1}, \dots, u_{(k-(N-1))}\right] = \begin{bmatrix} \propto_k^T \\ \propto_{k-1}^T \\ \vdots \\ \propto_{k-L+1}^T \end{bmatrix}$$

and $u_k = \left[u_k, u_{k-1}, \dots, u_{(k-(L-1))}\right]^T$ is the L length excitation vector with time index equal to k, $\propto_k = \left[u_k, u_{k-1}, \dots, u_{(k-N+1)}\right]^T$ is the N length excitation vector. Adaptive tap weight vector is defined as $w_k = \left[w_{0,k}, w_{1,k}, \dots, w_{L-1,k}\right]$ where $w_{i,k}$ is the ith tap at sample period k. There may be cases when $U_k^T . U_k$ matrix have eigen values close to zero. This condition will cause difficulty in finding the inverse. To avoid such uncertainties, δI term is added to $U_k^T . U_k$. The term $U_k^T . U_k + \delta I$ has smallest eigen value for δ which is large enough to produce well mannered inverse. Updated equation for w_k is given by:

$$w_k = w_{k-1} + \mu . U_k [U_k^T . U_k + \delta I]^{-1} . e_k \tag{29}$$

If N is set to 1, then Eq. (27) and (28) can be reduced to NLMS algorithm described in section II-B. Moreover, the stability of APA algorithm is seen for $0 < \mu < 2$ which is also similar to NLMS algorithm. Thus, it can be concluded that APA algorithm is the generalized version of NLMS algorithm. When complexity is calculated it is seen that affine projection algorithms have high complexity with respect to other least mean square algorithms. This is because of the various operations performed over the matrix. Operations like matrix inversion that alone gives $O(n^3)$ as the computational complexity [45]. Another distinguishing characteristic is that k^{th} order APA adaptive filter try to minimize the previous k instantaneous errors by appropriately selecting the weight w_k. Unlike, least mean square algorithm based adaptive filters which focus on minimizing the instantaneous error e_k.

III. Real Data Application

A. Description data datasets\

For experimental purpose, two data sets are taken from the resource file of National Renewable Energy Laboratory (NREL), which specializes in renewable energy efficiency, research, and development. Data-set *A* and data-set *B* corresponds to site *ID_A* 124693 and site *ID_B* 126541 respectively. Geographical location of the site ID_A is *longitude:-120.005463* and *latitude: 46.901657* with an average wind speed of 6.744 m/sec. Site ID_B has *longitude: -123.375778* and *latitude: 48.64072* with an average wind speed of 5.296 m/sec. NREL obtained all these observations from SCADA (supervisory control and data acquisition) system's wind plant which is located at height of 100 meters. Both the data sets have approx. hundred thousand entries. Each entry in both data set is recorded every 5 minutes. These entries include attributes like Year, Month, Day, Hour, Minute, power (in Mega Watts), wind direction at 100m (deg), wind speed at 100m (*m/s*), air temperature at 2m (K), surface air pressure (Pa) and density at hub height (*kg/m*^{3}). The observation period for wind-power data-set is from January 2012 to December 2012. As a matter of fact, data collection methods have few loop holes. This may result in out of range values. Cases like data points with negative or zero power values, missing values, inconsistent data combination such as data points with high wind speed producing low power values and vice versa. Using unclean data set can lead to misleading results. Therefore, before implementing the adaptive filters both the data-sets are pre-processed one by one. Pre- processing is done using inbuilt libraries and tools. Once the pre-processing is completed cleaned data sets *A* and *B* are obtained which are used to monitor the performance of the adaptive filters.

B. Parameters for error evaluation

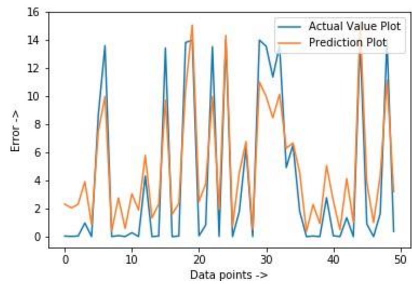
In order to evaluate the performance of all the algorithms discussed in section-II, few measures are used to compare their performance. These expressions will help to assess the adaptive filters on the basis of goodness of fit parameters and will help to verify the accuracy of the filter with respect to the particular algorithm. Let us assume a scatter plot of k points, where point t has coordinates (y_t, x_t). Expressions for evaluation purpose are as follow:

$$MAE = \frac{1}{k}\sum_{t=1}^{k} |y_t - x_t| \tag{30}$$

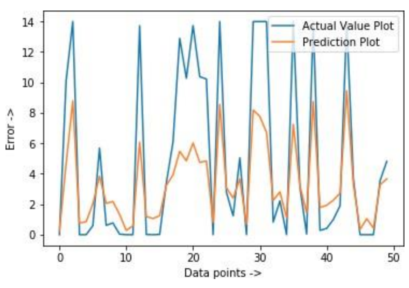
$$RMSE = \sqrt{\frac{1}{k}\sum_{t=1}^{k} (y_t - f_t)^2} \tag{31}$$

$$R^2 = 1 - \frac{\sum_{t=1}^{k}(y_t - f_t)^2}{\sum_{t=1}^{k}(y_t - y_{tm})^2} \tag{32}$$

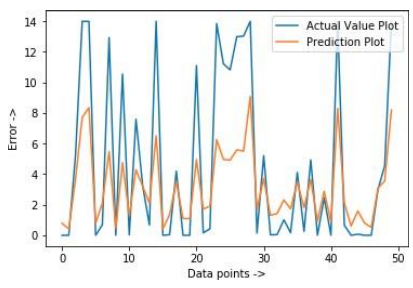
where, MAE is mean absolute error, RMSE is root mean square, R^2 is R-squared value, y_t is the actual wind power at time t, x_t is the value corresponding to respective y_t in the plot, f_t is the estimated value of the power at time t and y_{mt} is the mean of actual wind power for k samples. All these evaluation expressions together can be used to estimate goodness of the fit of the adaptive filter. MAE helps to clearly interpret the difference between the absolute values of y_t and x_t with respect to time t. It gives the absolute vertical distance between each point and Y=X line. RMSE is the standard deviation for the errors accounted during estimation.



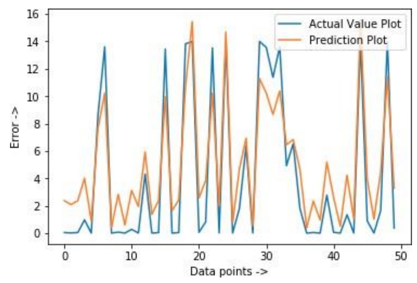
(a) LMS



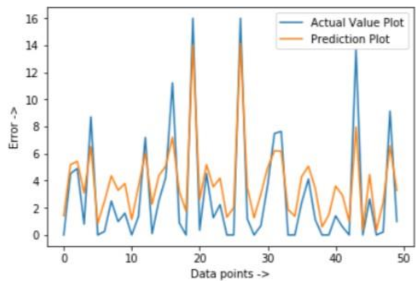
(b) NLMS



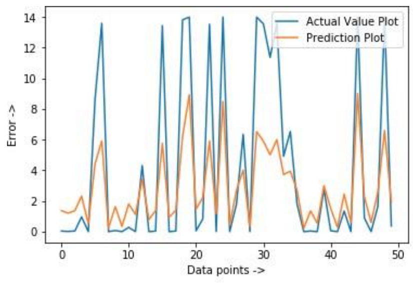
(c) GNGD



(d) WLMS



(e) RLS



(f) AP

Figure. 2: Error Representation of each algorithm for *dataset A*

S. No.	Algorithms	µ	MAE	RMSE	R^2	Rank
1.	**SLMS**	0.00001	1.994	2.298	0.684	3
2.	**NLMS**	0.0001	2.678	4.037	0.431	5
3.	**GNGD**	0.0001	2.698	4.082	0.418	6
4.	**WLMS**	0.00001	1.994	2.288	0.841	2
5.	**RLS**	1	1.961	2.173	0.852	1
6.	**AP**	0.1	2.613	3.659	0.548	4

Table 1: Comparison of Adaptive Algorithms for dataset *A*

S. No.	Algorithms	µ	MAE	RMSE	R^2	Rank
1.	**SLMS**	0.00001	1.994	2.298	0.684	3
2.	**NLMS**	0.0001	2.678	4.037	0.431	5
3.	**GNGD**	0.0001	2.698	4.082	0.418	6
4.	**WLMS**	0.00001	1.994	2.288	0.841	2
5.	**RLS**	1	1.961	2.173	0.852	1
6.	**AP**	0.1	2.613	3.659	0.548	4

Table 2: Comparison of Adaptive Algorithms for dataset *B*

Basically, it represents the standard deviation of the difference between observed values y_t and estimated values f_t. It also determines the intensity with which the data set is concentrated around the best fit line. It also acts as a measure of accuracy and helps to compare estimation errors of various models for a particular data set and not between two or more data-sets. MAE and RMSE both are directly proportional to the error. The lesser values of MAE and RMSE means lesser the error while estimating wind-power energy.

On the other hand, inverse relation exists between R^2 score and error ,i.e., a larger value of R^2 score means good fit with least error values. It determines how closely our model fits a particular dataset. Value of R^2 score lies between 0 and 1. Zero value of R^2 score means that the response variable does not move around its mean. In other words, a higher value of R^2 the better an algorithm fits the data set. Theoretically, negative values of R^2 does not exist. But in practical scenario, when an algorithm fails to fit a data set negative values of R^2 can be observed. R^2 score have certain limitations too. There are certain cases where low R^2 values do not necessarily mean a bad fit and similarly, high R^2 values do not necessarily mean a good fit. So, to fairly check the performance of the algorithms all the measures discussed in equation (30), (31), and (32) respectively are taken together and tabulated in table (1) and table respectively. After obtaining the statistical values from the table (1) and table (2), all the algorithms are also visually represented. Figure (2) and (3) shows the plot between error and data points for all adaptive filter algorithms for both datasets. Error on the Y-axis and Data points on the X-axis. Each plot shows visual representation between ”Actual Value Plot” and ”Prediction Plot”. Actual value plot i.e. actual data points are represented using a blue color line. On the other hand, the predicted plot is made using a red line. Each plot helps understand how close the algorithm is fitting the actual value plot.

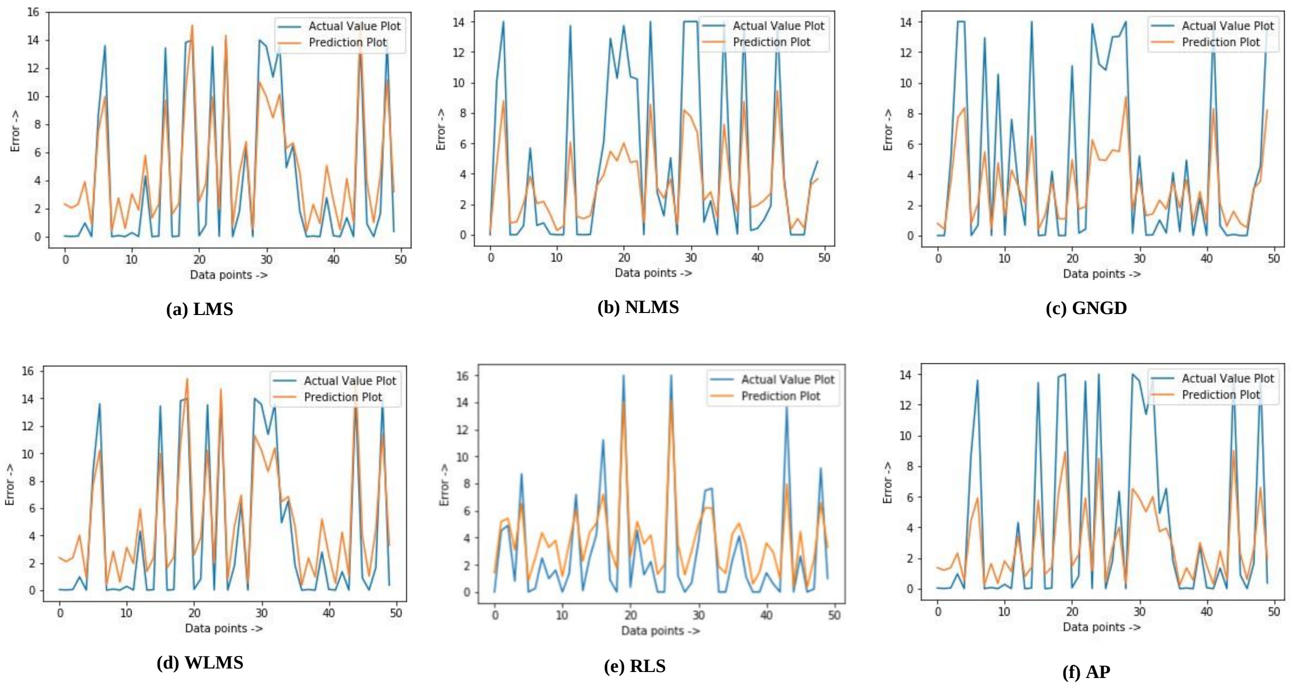


Figure. 3: Error Representation of each algorithm for *dataset B*

With these visuals, we will verify the behavior of each algorithm for both the datasets and draw conclusive results. Furthermore, additional parameters like computational complexity, convergence rate and stability of the system are discussed in table (3) on the basis of figure (4a) and (4b) respectively.

C. Experiment and Results

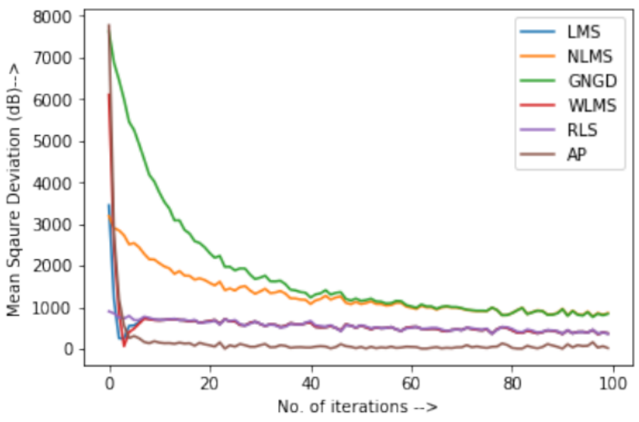
In the first simulation, adaptive filters are implemented based on each algorithm. Weights are initialized in three fashion: a one-dimensional array with initial weights of the filter size; creating zero value weights and creating random value weights. Once the weights are initialized and adaptive filters are implemented they are optimized for variable parameter. Here, step-size (μ) is the variable parameter in all the adaptive filters. It is also known as learning rate for algorithms. To achieve faster convergence larger values of variable parameter need to be selected. But smaller values of step-size results in better steady state square error. Therefore, proper selection of μ is an utterly important task. It should be able to balance the trade-off between convergence speed and mean absolute error values.

For each filter, the value of variable parameter μ is chosen by k-fold cross validation[46], where the value of k=10.Initially, the value of μ is varied between $(0.1)^{-4}$ to 1 and corresponding error of the data set is checked. Since the aim is to minimize the error, the optimized value of μ is selected for which the filter produces least error. Once μ is final for a filter, its value is fixed and is used to evaluate final error expressions. This process is repeated for each filter using its respective optimized μ because the value of μ is different for each algorithm.

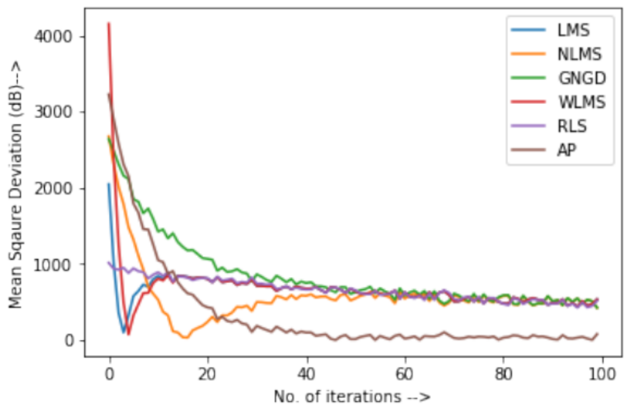
During simulations, few assumptions follow: the error $e(k)$ and input signals $u(k)$ sequences are statistically independent of one another. Also, during the entire procedure, the coefficients of the filter are statically independent of the input data present in filter memory. The performance of the filter is also dependent on parameters like convergence rate, estimated error. Table (1) and (2) shows the performance of the adaptive filters on basis of Mean absolute error (MAE), Root Mean Square Error (RMSE) and R-squared value (R^2) for each algorithm for both the data sets respectively. Table (3) shows the performance of the adaptive filters on basis of computational complexity, convergence rate, and stability of the system.

From table (1), for data set *A* it is clear that the RLS algorithm shows least values for MAE and RMSE both. In fact, it has the highest value of R^2 score which means that RLS was able to fit the data set as close as possible to observed values. Therefore, RLS scores rank first and have proved itself to be the best among other algorithms. At the same time, WLMS also shows good results. It shows second highest value for R^2score and hence ranked second on the list. It’s MAE, RMSE values are close to that of RLS which means these two algorithms show almost similar behavior for dataset *A*. Poor performance is shown by NLMS and GNGD algorithms. As discussed in section II-B, NLMS filter is supposed to show faster convergence than SLMS adaptive filter. At the same time, GNGD is also expected to achieve better performance than other least mean square based algorithms because it has additional gradient adaptive term in the learning rate as discussed in Sec. II-C. Upon experimentation, these two algos (NLMS,GNGD) failed to overpower the results shown by SLMS. NLMS and GNGD showed high value for both error estimation and lowest values for R^2 score. Hence these two are ranked lowest on the list. In table (1), AP algorithm has close values of MAE when compared with MAE values of NLMS and GNGD algorithms. At the same time, AP shows low RMSE value which means that it was able to provide a better fit around the best fit line with respect to NLMS and GNGD algorithms.

This can also be verified with its R^2 score which is relatively higher than both NLMS and GNGD. SLMS and WLMS have same values of MAE but RMSE values differ which clearly states that WLMS provides a better fit as it has less value of RMSE.



(a) For Data set A



(b) For Data set B

Figure 4: Convergence Curves for Advanced Adaptive Filter Algorithms

Again this can be verified by looking at the R^2 score as higher value of this parameters how show good the algorithm has fitted the data set. That is why SLMS is ranked third below the rank of WLMS. So, experimental results obtained from SLMS and WLMS verifies the theoretical aspects sec. II-A and II-D respectively.

From table (2), for data set *B*, RLS algorithm has again proved itself top notch in the list. It has highest R^2 score value along with least MAE and RMSE values. Henceforth, it is proved that RLS algorithms work best in time-varying environments. WLMS stands at rank second on the list with second highest R^2 score and error values. RMSE values of GNGD algorithm and WLMS algorithm is same this does not mean that both have performed equally. As there is a significant difference in MAE values and R^2 score WLMS clearly wins the race. SLMS again shows average performance for this data set as well. GNGD, NLMS, AP again failed to provide a good fit. All three have R^2 score less than 50% and they dissolute their purpose. On the basis of MAE, RMSE and R^2 score GNGD,NLMS, and AP are ranked fourth, fifth and sixth respectively. Hence, it can be concluded that the performance of algorithms is dependent on the dataset also. It does not matter how good an algorithm is theoretically it needs to prove its worth during experimentation with respect to data sets.

After discussing the statistics of all the algorithms upon experimentation, let's now look at the visual representation of these algorithms for both the datasets. Figure 2 and 3 shows the plot between error and data points for all adaptive filter algorithms for both datasets. Each plot shows visual representation between "Actual Value Plot" and "Prediction Plot". Actual value plot i.e. actual data points are represented using a blue color line. On the other hand, the predicted plot is made using a red line. From fig. 2 and 3 it is easy to infer how the algorithms have performed within the dataset. Close proximity of between the lines of actual value plot and prediction plot means that the particular algorithm was able to provide a good fit for the dataset. Similarly, if the predicted plot is not close to the actual value plot then that algorithm failed to provide a better fit for the dataset. The same can also be verified by looking at the R^2 score values of the respective algorithm in tab. 1 and tab.2.

Let's consider two plots of RLS algo 2e and 3e from fig. 2 and 3 respectively. In both the plots, we can observe that Prediction plot was able to provide a very close fit to the actual value plot. This can also be verified by the RLS algorithm's R^2 score in the tab. 1 and 2 respectively. Higher the R^2 score better the fit. RLS's R^2 score is 0.852 and 0.758 for dataset A and B respectively. This score is close to 1 which is the maximum value for R^2 score. Similarly, let's consider the plots of NLMS in 2b and 3b from fig.2 and 3 respectively. In both the plots, we can observe that the prediction plot is not close to the actual value plot. There is a considerable difference between the blue line and red line of the plots. This proves that NLMS has provided a bad fit for both the datasets. Again this can be verified by the R^2 score. of NLMS algo in the tab. 1 and 2 respectively. For both datasets, NLMS's R^2 score was less than 0.5. Hence, the NLMS plots 2b and 3b correctly described the behavior of NLMS algorithm for both the datasets. Similar conclusions can be drawn from the rest of the plots in fig. 2 and 3. The plots of WLMS 2d and 3d also justifies the higher R^2 score value that WLMS has shown upon experimentation in tab.1 and 2. Rest plots for algorithms SLMS, AP and GNGD also satisfy the statistical results obtained in tab. 1 and 2. We can see that for all these algorithm's plots there is a considerable difference in the predicted plot and actual value plot. This difference in the predicted plot and actual value plot shows why these three algorithms: SLMS, AP, GNGD have poor value for R^2 score. That is why they are ranked based on statistics of MAE, RMSE and R^2 score values.

Results in table (3) is computed with help of figure (4a) and (4b) together. It depicts the computational complexities, the convergence rate of each algorithm, and the stability of the system. Figure (4a) and (4b) illustrates the convergence curves for advanced adaptive filter algorithms for data sets *A* and *B* respectively. From table (3), it can be concluded that RLS has shown steady convergence rate and has taken the system to a stable state for both the data sets.

Though it has greater computational complexity it successfully outperformed every other algorithm in terms of error and convergence rate and took the system to a highly stable state.

S. No.	Algorithms	Complexity	Convergence Rate	Stability
1.	**SLMS**	2N	Very Fast	Less Stable
2.	**NLMS**	3N+1	Fast	Stable
3.	**GNGD**	5N+2	Slow	Stable
4.	**WLMS**	2N+1	Very Fast	Less Stable
5.	**RLS**	$3N^2$+4N	Moderate	High Stable
6.	**AP**	N^2+2N	Moderate	Stable

Table 3: Comparison of Adaptive Algorithms on basis of Complexity, Convergence Rate, Stability of the Algorithms

SLMS(or LMS) and WLMS have also shown very fast convergence rate. SLMS is comparable to WLMS in terms of computational complexity. The computational complexity of GNGD is approximately two times that of NLMS as it uses additional gradient adaptation term called regularization factor. Although its complexity can be reduced by imposing some hard bounds like stopping its adaptation after convergence. However, no such constraints were used while experimenting here. Theoretically, APA algorithm was supposed to outperform all other algorithms specially SLMS, NLMS, and GNGD as discussed in Sec. II-F. But it failed to do so. It exhibited a slower convergence rate and more error values for both data sets. Therefore, ranked fourth for data set *A* and sixth for data set *B*.

IV. Conclusion and Future Works

All the algorithms discussed in this paper are stochastic gradient approximations to the steepest-descent method. These algorithms are implemented with the aim to minimize the mean absolute error between the desired signal $d(k)$ and output signal $y(k)$. As the result of performance analyses, RLS algorithm proved to be the most suitable algorithm for estimating wind-power generation. Therefore, it is ranked first for both the datasets *A* and *B*. At the same time, RLS has proved to be best in terms of convergence speed as well as the stability of the system. Though its computational complexity is high w.r.t to all other algorithms.

WLMS is found to be comparable to RLS and can be used in similar data sets. It also showed good convergence which is remarkable to take the system towards a stable state. Henceforth, WLMS takes second place in the list. The next algorithm in the race is SLMS. SLMS algo exhibited a very fast convergence rate but at the same time, the stability of the system was not so high. That is why SLMS is ranked third in the list. Experimental results obtained from SLMS and WLMS verifies the theoretical aspects discussed in Sec. II-A and II-D respectively. When other algorithms like APA is seen its resultant MAE and RMSE is greater than that of SLMS and NLMS. Despite the fact, APA algorithm should have performed better as sole purpose of this algorithm was to overcome the drawbacks of least mean square based algorithms as discussed in section II-F. However, the performance of APA can be improved by increasing the projection order *N* which will result in high computational complexity. Another reason for AP's high complexity is that it performs various operations over the matrix. Operations like matrix inversion which alone gives $O(n^3)$ as the computational complexity [45]. APA is ranked fourth considering its performance from tab. (1), tab. (2) and tab. (3). In theoretical aspects, proper selection of μ while implementing NLMS filter was expected to show faster convergence than SLMS adaptive filter. This is because NLMS algorithm differs from SLMS algorithm only on one factor: time-varying step size $\mu(k)$ known as data-dependent adaptation step size. Though while experimentation, NLMS did not offer better trade-off over SLMS, while μ was properly optimized to get best results. That is why NLMS is ranked third on the list. GNGD stands among the worst performers in the list. GNGD algorithm has an additional gradient adaptive term in the learning rate which enables it to adapt its learning rate with the dynamics of the input signal vector. But in spite, this additional gradient adaptive term GNGD has highest MAE, RMSE values and lowest R^2 scores. Thus, we can conclude that no matter how good an algorithm is theoretically it is always dependent on the dataset to which it is applied. Performance of the algorithms can be further improved but at the cost of increasing its computational complexity. Other algorithms too can be implemented to form adaptive filters which may show better performance in terms of error, complexity, convergence, and stability.

REFERENCES

[1] Ipakchi, A., Albuyeh, F. Grid of the future, *IEEE Power and Energy Magaz.*, vol. 7, no. 2, pp. 52-62, 2009.

[2] Tchakoua, P., Wamkeue, R., Ouhrouche, M., Hasnaoui, F.S., Tameghe, T.A., Ekemb, G. Wind Turbine Condition Monitoring: State-of-the-Art Review, New Trends, and Future Challenges, *Journ. of Energies*, vol. 7, pp. 2595-2630, 2014.

[3] International Energy Agency (IEA), *World energy outlook*, IEA, Paris, Tech. Rep., 2013.

[4] Carrillo, C., Montaño, A.O., Cidrs, J., Diaz-Dorado, E. Review of power curve modeling for wind turbines, *Renewable and Sustain. Energy Reviews*, vol. 21(C), pp. 572-581, 2013.

[5] Lydia, M., Selvakumar, A., Kumar, S., Kumar, G. Advanced algorithms for wind turbine power curve modeling, *IEEE Trans. Sustainable Energy*, vol. 4, no. 3, pp. 827-835, 2013.

[6] Giorsetto, P., Utsurogi, K. F. Development of a new procedure for reliability modeling of wind turbine generators, *IEEE Trans. Power App. Sys.*, vol. PAS-102, no. 1, pp. 134-143, 1983.

[7] Shokrzadeh, S., Jozani, M.J. Wind Turbine Power Curve Modeling Using Advanced Parametric & Non-parametric Methods, *IEEE Trans. on Sustain. Energy*, vol. 5, no. 4, pp. 827-835, 2013.

[8] Gasch, R., Twele, J. Wind Power Plants in *Fundamentals, Design, Const.* & *Operation*, Springer-Verlag., Berlin, Germany, 2012.

[9] Stuntas, T.U., Sahin, A.D. Wind turbine power curve estimation based on cluster center fuzzy logic modeling, *Jour. of Wind Engg.* & *Industrial Aerodynamics*, vol. 96, no. 5, pp. 611-620, 2008.

[10] Morshedizadeh, M., Kordestani, M., Carriveau, R., Ting, D.S., Saif, M. Improved power curve monitoring of wind turbines, *Journ. of Wind engg.*, vol. 41, no.4, pp. 260-271, 2017.

[11] Marvuglia, A., Messineo, A. Monitoring of wind farms power curves using machine learning tech., *Article in Applied Energy*, vol. 98, pp. 574-583, 2012.

[12] Kusiak, A., Zheng, H., Song, Z. Models for monitoring wind farm power, *Art. in Renewable Energy*, vol. 34, no. 3, pp. 583-590, 2009.

[13] Morshedizadeh, M., Kordestani, M., Carriveau, R., Ting, D.S., Saif, M. App. of imputation tech. and Adaptive Neuro-Fuzzy Inference Sys. to predict wind turbine power production, *Jour. of Energy*, vol. 138, pp. 394- 404, 2017.

[14] Deepika, M., Sujatha, A. Noise Cancellation In Speech Signal Processing Using Adaptive Algorithm, *Intern. Jour. on Recent and Innovation Trends in Computing and Comm.*, vol. 1, no.9, pp. 743-746, 2013.

[15] Weifeng, L., José, C.P., Haykin S. *Kernel Adaptive Filtering: A Comprehensive Introd.*, John Wiley & Sons, 2011.

[16] Madisetti, V.K., Williams, D.B. *Introg. to Adaptive Filters Digital Signal Processing Handbook* 2nd *Ed.*, Boca Raton, CRC Press LLC, 1999.

[17] Macchi, O. *Adaptive Processing: The Least Mean Squares Approach with Applications in Transmission*, vol. 23, no. 11, pp. 45-78, Wiley, 1995.

[18] Arora A., Wadhvani R. (2018) Comparative Analysis of Adaptive Filters for Predicting Wind-Power Generation (SLMS, NLMS, SGDLMS, WLMS, RLMS). In: Abraham A., Muhuri P., Muda A., Gandhi N. (eds) Intelligent Sys. Design and App.. ISDA 2017. Advances in Intelligent Sys. & Computing, vol 736. Springer, Cham

[19] Patil, A.P., Patil, M.R. Computational Complexity of Adaptive Algorithms in Echo Cancellation, *SSRG Intern. Jour. of Elec. and Comm. Engg. (SSRG-IJECE)*, vol. 2, no. 7, 2015.

[20] Clarkson, P.M. *Optimal and Adaptive Signal Processing*, CRC Press, 1993.

[21] Douglas, S.C. *Convergence Issues in the LMS Adaptive Filter*, CRC Press LLC, 2000.

[22] Dhiman, J., Ahmad, S., Gulia, K. Comparison between Adaptive filter Algorithms (LMS, NLMS and RLS), *Intern. Jour. of Sc., Engg. and Tech. Research (IJSETR)*, vol. 2, no. 5, 2013.

[23] Haykin, S. *Adaptive Filter Theory*, Pearson Ed. India, 2008.

[24] Karne, V.M., Thakur, A.S., Tiwari, V. Least Mean Square (LMS) Adaptive Filter for Noise Cancellation, *International Jour. of App. or Innovation in Engg.* & *Management (IJAIEM)*, S.I. for National Conf. On Recent Advances in Tech. and Management for Integrated Growth, 2013.

[25] Widrow, B., Hoff, M.E. Adaptive switching circuits, *IRE WESCON Convention Record*, vol. 4, pp. 96-104, 1960.

[26] Shoval, A., Johns, D.A., Snelgrove, W.M. Comparison of DC Offset Effects in Four LMS Adaptive Algorithms *IEEE Trans. on Cir.* & *Syst.-II: Analog and Digital Signal Proc.*, vol. 42, no. 3, pp. 176-185, 1995.

[27] Sharma, A., Juneja, Y. Acoustic Echo Cancellation of from the Signal Using NLMS Algorithm, *Intern. Jour. of Research in Advent Tech.*, vol.2, no. 6, 2014.

[28] Sharma, L., Mehra, R. Adaptive Noise Cancellation using Modified Normalized Least Mean Square Algorithm, *Intern. Jour. of Engg. Trends and Tech.(IJETT)*, vol. 34, no. 5, 2016.

[29] Sankaran, S.G., Beex, A.A. Convergence Behavior of Affine Projection Algorithms, *IEEE Trans. On Signal Proc.*, vol. 48, no. 4, 2000.

[30] Hur, M., Choi, J.Y., Baek, J.S., Seo, J. Generalized Normalized Gradient Descent Algorithm Based on Estimated a Posteriori Error, *Proc. in Intern. Conf. on Adv. Comm. Tech. ICACT*, vol. 1, pp. 23-26, 2008.

[31] Mandic, D.P. A generalized normalized gradient descent algorithm, *Jour. of IEEE Signal Proc. Society*, pp. 115-118, January 2014.

[32] Paleologu, C., Vladeanu, C., Ciochina, S., Enescu, A.A. Generalized Normalized Gradient Descent Algorithm with Direct Update of the Step-Size Parameter, In *Proc. of IEEE Intern. Symposium on Sig., Cir.* & *Syst.*, Iasi, Romania, 2007.

[33] Clarkson, P.M. Elec. Engg. systems, in *Optimal and Adaptive Signal Processing*, CRC press, 1993.

[34] Haykin, S. *Adaptive filter theory*, Prentice Hall, 1996.

[35] Sayed, A.H., Kailath, T. *Recursive Least-Squares Adaptive Filters*, John Wiley & Sons, 2000.

[36] Zulfiquar, Md., Bhotto, A., Antoniou, A. New Improved Recursive Least-Squares Adaptive-Filtering Algorithms, *IEEE Trans. on Cir.* & *Syst.*, vol. 60, no. 6, 2013.

[37] Marshall, D.F., Jenkins, W.K., Murphy, J.J. The use of orthogonal transforms for improving performance of adaptive filters, *IEEE Trans. on Cir.* &*Syst.*, vol.36, pp. 474-485, 1989.

[38] Ozeki, K., Umeda, T. An adaptive filtering algorithm using an orthogonal projection to an affine sub-space and its properties, *Jour. of Elec.* & *Comm.*, vol. 67-A, no. 5, pp. 19-27, May 1984.

[39] Gonzaleza, A., Ferrera, M., Albub, F., Diego, M. Affine Projection Algorithms: Evolution To Smart And Fast algorithms And Applications, In *Proc. of* 20th *European Signal Proc. Conf.*, Bucharest, Romania, 2012.

[40] Satyanarayana, J., Khan, G.A. Affine Projection Algorithm Applied to Adaptive Noise Cancellation, *The In- tern. Jour. of Elec.* & *Com. Tech.*, vol. 3, no. 1, 2012.

[41] Kratzer, S.G., Morgan, D.R. The partial Rank Algorithm for Adaptive Beam Forming, In *Proc. of SPIE conference*, vol. 564, pp. 9-14 1985.

[42] Sankaranand, S.G., Beex, A.A. Normalized LMS algorithm with orthogonal correction factors, In *Proc. of* 31st *Asilomar Conf. on Signals, Sys.* & *Computers*, vol. 2, pp. 1670-1673, 2002.

[43] Gay, S.L., Benesty, J. *Acoustic signal processing for telecomm.*, 2nd ed., Kubler Acad. Pub., 2001.

[44] Waterschoot, T.V., Rombouts, G., Moonen, M. Optimally regularized adaptive filtering algorithms for room acoustic signal enhancement, *Jour. of Signal Processing*, vol. 88, no. 3, pp. 594-611, 2008.

[45] Sankaran, S.G., Beex, A.A. Convergence Behavior of Affine Projection Algorithms, *IEEE Trans. On Signal Proc.*, vol. 48, no. 4, 2000.

[46] Fushiki, T. Estimation of prediction error by using K-fold cross-validation, *Springer Sci.* & *Business Med.*, vol. 21, no. 2, pp. 137-146, 2011.

Author Biography

Ashima Arora was born in Gwalior, Madhya Pradesh, in 1993. She received the B.Tech. degree in computer sci. and engg. from the Amity University, M.P. in 2016. Also, M.tech deg. (2016-2018) in Computer Networks, Dept. of Computer Sci. and Engg., Maulana Azad National Inst. of Tech., Bhopal. She joined as Software Engineer-Data Scientist at Philips Innovation Campus, Bangalore in 2018. She has authored a paper in Intelligent Systems Design and Applications organized by MIR LABS, USA- (Springer Publication). Her research interests include machine learning, algorithm design and analysis, data struct., database management and sys., computer graphics and operating sys..