INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 11, NO. 1, MARCH 2006, 17-25

System Identification Based on Generalized ADALINE Neural Network

Wenle ZHANG

Abstract

System identification of linear time-varying systems consists of estimation of system parameters that change with time. In this paper, we present an online identification method for such systems based on a generalized ADAptive LINear Element (ADALINE) neural network. It is well known that ADALINE is slow in convergence and, hence, not appropriate for online application and identification of time varying systems. ADALINE is generalized such that the input now consists of a Tapped Delay Line of the system input signal and a Tapped Delay Line of the system output feedback. Two techniques are proposed to speed up the convergence of learning, thus increase the capability of tracking time varying system parameters. One idea is to introduce a momentum term to the weight adjustment during the convergence period and the learning curve is smoothed by turning off the momentum once the error is within a given small number – epsilon. The other technique is to train the generalized ADALINE network multiple epochs with data from a sliding window of the system's input output data. Simulation results show that the proposed method provides a much faster convergence speed and better tracking of time varying parameters. The low computational complexity makes this method suitable for online system identification and real time adaptive control applications.

Key words – **System identification, neural network, ADALINE, tapped delay line feedback.**

1. INTRODUCTION

In general, system identification is to determine the model structure and parameters for a dynamic system based on the measurable input and output data of the system. A model structure in its simplest form for a single input single output (SISO) linear system is determined by the system order. Once the order is determined, identification reduces to a parameter estimation problem which is considered to be well solved by traditional identification methods, such as least squares method, maximum likelihood method and instrumental variable method (Ljung 1999, Söderström and Stoica 1989). Increased interests in the system identification area based on neural network techniques (Haykin 1999, Mehrotra 1997) have been seen for the past decade.

Manuscript received on January 12, 2006; revised on February 20, 2006.
The author is with School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701. Email: zhangw@ohio.edu

A neural network is a massively parallel distributed processor made up of simple processing units, called neurons, which has the natural propensity for storing experiential knowledge and making it available for use (Haykin 1999). Most neural networks used for system identification are in the category called recurrent (feedback) neural networks, such as, Multi-Layer Perceptron (MLP) (Rumelhart 1986) with feedback and Hopfield neural network (Hopfield, 1982). Most neural network methods studied for system identification are mainly for nonlinear systems based on MLP (Sjöberg et al. 1994, Narendra and Parthasarathy 1990, Qin et al. 1992). Other types of neural networks include Hopfield (Chu et al. 1990), Radial Basis Function (Valverde 1999), Support Vector (Gretton 2001) and Self-Organizing Map (Abonyi and Szeifert 2002).

In this paper, the author presents an online identification method based on a generalized ADAptive LINear Element (ADALINE) neural network, called GADALINE, for linear time varying systems which can be described by a discrete time model. In such systems, the current system output depends on past outputs and on both the current and past inputs. GADALINE needs to have output feedback and to remember past input/output data. The proposed neural network employed a Tapped Delay Line (TDL) for both input and output of the system to remember the past input/output data. GADALINE is then a linear recurrent type of neural network. Adaptive learning is generalized by adding a momentum term to the GADALINE weight adjustment. This momentum term is turned on only during the convergence period and the learning curve is therefore smoothed by turning off the momentum once the learning error is within a given small number – epsilon. The author further generalizes the online training by using each set of samples for several epochs obtained from a sliding window of the system's input and output data. Simulation results show that GADALINE provides a much faster convergence speed and better tracking of time varying parameters. Due to its simple structure, the learning algorithm has an exceptionally low computational complexity and makes this method suitable for online system identification and real time adaptive control applications. The rest of the paper is organized as: Section 2 introduces the linear systems to be considered; Section 3 presents GADALINE; Section 4 provides the identification methods using GADALINE; Section 5 discusses some simulation results; and Section 6 concludes the paper.

2. LINEAR SYSTEM AND IDENTIFICATION

Consider a discrete time linear SISO system. The observable input and output data can be given in the form of a time series: $\{u(k\mathrm{T}), y(k\mathrm{T})\}$, T is the sample period, a constant, thus the data is often simply written as $\{u(k), y(k)\}$. Then the system can be modeled by the following difference equation,

$$y(k) + a_1 y(k-1) + a_2 y(k-2) + \ldots + a_n y(k-n) = b_1 u(k-1) + b_2 u(k-1) + \ldots + b_m u(k-m) \quad (1)$$

where n and m are system structure parameters, $m \le n$, n is the order of the system; a_i and b_j are system parameters, $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. If a_i and b_j are constants, the system is said to be time invariant, otherwise time varying. In the time varying case, a_i and b_j can be slowly changing with time or can be switched to different values at different time instants.

A unit time delay operator is often used to simplify this model. Assume z^{-1} is such an operator. Then $z^{-1}y(k)$ is the output of the system at time instant $(k-1)\mathrm{T}$, that is, $y(k-1) = z^{-1}y(k)$; $z^{-2}y(k)$ is the output of the system at time instant $(k-2)\mathrm{T}$; and so on. Then, the model in (1) can be reduced to a polynomial form,

$$\mathrm{A}(z^{-1})\, y(k) = \mathrm{B}(z^{-1})\, u(k) \quad (2)$$

where

$$\mathrm{A}(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}$$

$$\mathrm{B}(z^{-1}) = b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}$$

By adding an error term to the right side of (2), the AutoRegressive with eXorgenous input model, called ARX model (Ljung, 1999) can be obtained, which is the most widely used model for the system considered, as shown in Fig. 1.

$$\mathrm{A}(z^{-1})\, y(k) = \mathrm{B}(z^{-1})\, u(k) + e(k) \quad (3)$$

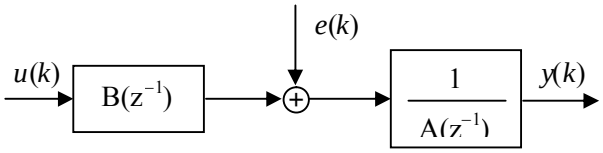


Fig. 1. ARX model.

Then the system identification becomes a parametric identification problem. That is, given n and m, determine coefficients of A and B polynomials from the input and output data of the system according to a certain criterion, such as minimizing an error function, which is a measure of how close the model is to the actual system. Figure 2 shows a general system identification architecture, where $y_u(k)$ is the deterministic output, $n(k)$ is the noise (white) and $e(k)$ is the error.

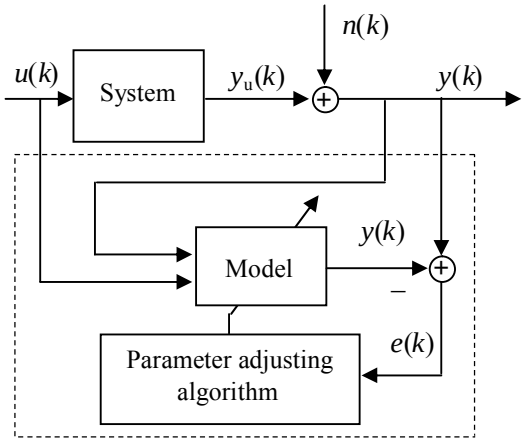


Fig. 2. System identification for ARX model.

In the following, the paper will discuss how the model in the dashed line box in Fig. 2 can be approached by training a generalized ADALINE, whose weights correspond to coefficients of both polynomials $\mathrm{A}(z^{-1})$ and $\mathrm{B}(z^{-1})$.

3. THE GENERALIZED ADALINE

ADALINE (Widrow and Lehr 1990) was first developed to recognize binary patterns so that if it was reading streaming bits from a phone line, it could predict the next bit. MADALINE was the first neural network applied to a real world problem, using an adaptive filter that eliminates echoes on phone lines. The names come from their use of multiple ADAptive LINear Elements. The structure of an ADALINE is shown in Fig. 3, where a threshold logic unit can also be attached to the output.

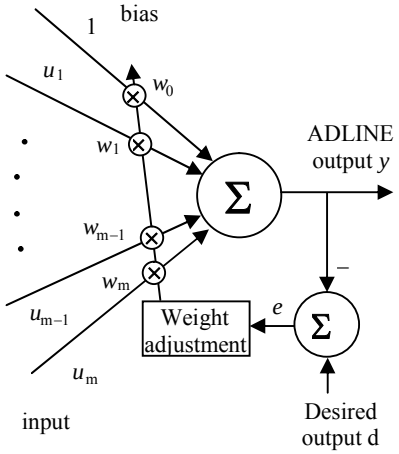


Fig. 3. An ADLINE unit.

To apply the ADALINE to system identification, we need to modify both the structure and the learning algorithm.

3.1 Structure of GADALINE

In the GADALINE structure, the bias input is removed, and each input is expanded through a TDL to become a number of delayed inputs and the output feedback is introduced to the input, also passing through a TDL to become a number of inputs. If we still keep the notation for inputs as $u_1 \sim u_m$, then the output y can be found by,

$$y = \Sigma_1^m w_i u_i = \boldsymbol{u}^T \boldsymbol{w}$$

where $\boldsymbol{u}$ is the input vector and $\boldsymbol{w}$ is the weight vector,

$$\boldsymbol{u} = [u_1\ u_2\ \ldots\ u_m]^T$$

$$\boldsymbol{w} = [w_1\ w_2\ \ldots\ w_m]^T$$

3.2 GADALINE Learning Algorithm

The GADALINE learning algorithm is based on the Widrow-Hoff rule with two types of generalizations to speed up convergence. The learning algorithm is derived in the following and the implication on the generalized learning will also be discussed.

Widrow-Hoff's delta rule or Least Mean Square algorithm (LMS), is based on minimizing the cost function,

$$E(\boldsymbol{w}) = 1/2\ e^2(t)$$

where t is the iteration index and $e(t)$ is given by,

$$e(t) = d(t) - y(t)$$

where the desired output $d(t)$ at each iteration t is constant. The gradient vector of the cost function is,

$$\boldsymbol{g} = \nabla E(\boldsymbol{w}) = [\partial E/\partial w_1, \partial E/\partial w_2, \ldots, \partial E/\partial w_m]^T$$

According to the LMS algorithm, the weight adjustment is

$$\Delta \boldsymbol{w}(t) = \boldsymbol{w}(t+1) - \boldsymbol{w}(t) = -\eta \boldsymbol{g}(t)$$

where η is the learning parameter, a small positive number.

To show that the LMS algorithm reduces error, we use a first order Taylor series expansion around $\boldsymbol{w}(t)$,

$$E(\boldsymbol{w}(t+1)) \cong E(\boldsymbol{w}(t)) + \boldsymbol{g}^T(t)\,\Delta \boldsymbol{w}(t) = E(\boldsymbol{w}(t)) - \eta \boldsymbol{g}^T(t)\boldsymbol{g}(t) = E(\boldsymbol{w}(t)) - \eta |\boldsymbol{g}(t)|^2$$

So the cost function is decreased as the algorithm goes from one iteration to the next.

Now differentiating $E(\boldsymbol{w})$ with respect to the weight vector $\boldsymbol{w}$ yields,

$$\partial E(\boldsymbol{w})/\partial \boldsymbol{w} = e(t)\ \partial e/\partial \boldsymbol{w}$$

and

$$e(t) = d(t) - y(t) = d(t) - \boldsymbol{u}^T(t)\boldsymbol{w}(t)$$

So,

$$\partial e/\partial \boldsymbol{w} = -\boldsymbol{u}(t)$$

Hence,

$$\boldsymbol{g} = \partial E(\boldsymbol{w})/\partial \boldsymbol{w} = -e(t)\boldsymbol{u}(t)$$

Then,

$$\Delta \boldsymbol{w}(t) = -\eta \boldsymbol{g}(t) = \eta\ e(t)\boldsymbol{u}(t) \qquad (4)$$

and

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \Delta \boldsymbol{w}(t) \quad \text{or} \quad = \boldsymbol{w}(t) + \eta\ e(t)\ \boldsymbol{u}(t) \qquad (5)$$

where the learning–rate parameter is usually in the range of $0 < \eta < 2$.

It is well known (Haykin 1999), the LMS algorithm has a slow convergence speed. Especially for bigger learning parameter, the convergence trajectory exhibits a zigzag form. This can be stabilized partially by normalizing input vector to a unit vector in the weight adjustment equation (5) so that the weight adjustment magnitude is independent of the magnitude of the input vector. That is, the weight adjustment is given by,

$$\Delta \boldsymbol{w} = \eta\ e\ \boldsymbol{u}/\|\boldsymbol{u}\|$$

where $\|\boldsymbol{u}\|$ is the Euclidean norm of the input vector $\boldsymbol{u}$ or the length, which is calculated as,

$$\|\boldsymbol{u}\| = \sqrt{u_1^2 + u_2^2 + \ldots + u_m^2}$$

In order to speed up the convergence and thus to increase the capability of tracking time varying system parameters, we propose two techniques: i) introducing a momentum term into the weight adjustment equ. (4); and ii) training multiple epochs with data from a sliding window of the entire set of input samples.

3.2.1 Momentum in Learning

With a momentum term introduced into the weight adjustment equ. (4), the new weight adjustment now becomes,

$$\Delta \boldsymbol{w}(t) = \eta\ e(t)\boldsymbol{u}(t) + \alpha \Delta \boldsymbol{w}(t-1) \qquad (6)$$

where $1 > \alpha \geq 0$ is a small non-negative number.

To see the effect of the momentum, we can expand the equation starting from $t = 0$,

$$\Delta \boldsymbol{w}(0) = \eta e(0)\boldsymbol{u}(0)$$

$$\Delta \boldsymbol{w}(1) = \eta e(1)\boldsymbol{u}(1) + \alpha \Delta \boldsymbol{w}(0) = \eta e(1)\boldsymbol{u}(1) + \alpha\eta e(0)u(0)$$

$$\Delta \boldsymbol{w}(2) = \eta e(2)\boldsymbol{u}(2) + \alpha \Delta \boldsymbol{w}(1) = \eta e(2)\boldsymbol{u}(2) + \alpha\eta e(1)\boldsymbol{u}(1) + \alpha^2 \eta e(0)\boldsymbol{u}(0)$$

…

$$\Delta \boldsymbol{w}(t) = \eta e(t)\boldsymbol{u}(t) + \alpha \Delta \boldsymbol{w}(t-1) = \eta e(t)\boldsymbol{u}(t) + \alpha\eta e(t-1)\boldsymbol{u}(t-1) + \ldots + \alpha^{n-1}\eta e(1)\boldsymbol{u}(1) + \alpha^n \eta e(0)\boldsymbol{u}(0)$$

$$= \eta \sum_{\tau=0}^{t} \alpha^{t-\tau}\, e(\tau)\boldsymbol{u}(\tau)$$

Remember $e(t)\boldsymbol{u}(t)$ is just $-\partial E(t)/\partial \boldsymbol{w}(t)$, then

$$\Delta \boldsymbol{w}(t) = -\eta \sum_{\tau=0}^{t} \alpha^{t-\tau}\, \partial E(\tau)/\partial \boldsymbol{w}(\tau)$$

So, we have an exponentially weighted time series. We can then make the following observations,

i) If the derivative $\partial E(t)/\partial \boldsymbol{w}(t)$ has opposite sign on consecutive iterations, the exponentially weighted sum $\Delta \boldsymbol{w}(t)$ reduces in magnitude, so the weight $\boldsymbol{w}(t+1)$ is adjusted by a

small amount, which reduces the zigzag effect in weight adjustment.

ii) On the other hand, if the derivative $\partial E(t)/\partial \boldsymbol{w}(t)$ has same sign on consecutive iterations, the weighted sum $\Delta w(t)$ grows in magnitude, so the weight $w(t+1)$ is adjusted by a large amount. The effect of the momentum term is to accelerate descent in steady down hill region.

Because of this added momentum term, the weight adjustment continues after convergence, that is, when even the magnitude of the error term e is small. For linear system identification purpose, these weights correspond to system parameters. It is desirable to keep these parameters fixed once convergence is seen. So we should turn off the momentum term after convergence is detected, say $|e| < \varepsilon$, $\varepsilon > 0$ is a small number. However, when the system parameters change at a later time, another convergence period will begin. Then it is better to turn the momentum back on.

GADALINE training is performed by presenting the training pattern set – a sequence of input-output pairs. During training, each input pattern from the training set is presented to the GADALINE and it computes its output y. This value and the target output from the training set are used to generate the error and the error is then used to adjust all the weights. Figure 4 summarizes the generalized LMS learning algorithm, called GLMS.

GADLINE **Training Algorithm I**

Start with a randomly chosen weight vector $\boldsymbol{w}(0)$
Let $t = 1$
While % online version is an infinite loop
  Let $\boldsymbol{u}(t) = (u_1, .., u_m)$ be next input vector, for which $d(t)$ is the desired output
  Calculate $e(t) = d(t) - \boldsymbol{u}^T(t)\boldsymbol{w}(t-1)$
  $\Delta \boldsymbol{w}(t) = \eta e(t)\boldsymbol{u}(t)$
  If $|e(t)| > \varepsilon$, **then**
    $\Delta \boldsymbol{w}(t) = \Delta \boldsymbol{w}(t) + \alpha \Delta \boldsymbol{w}(t-1)$
  Adjust the weight vector to
    $\boldsymbol{w}(t) = \boldsymbol{w}(t-1) + \Delta \boldsymbol{w}(t)$
  Increment t
End while

Fig. 4. The GLMS algorithm.

3.2.2 Multiple Epochs Training with a Sliding Window

Another technique to increase the convergence speed and the capability of tracking time varying system parameters is to train the ADALINE network multiple epochs with data obtained from a sliding window of the entire set of input samples. Assume that the size of the sliding window is s. Then only the most recent s data samples will be used to adapt the weights of the ADALINE. At iteration t, the data set for an epoch is,

$\{\boldsymbol{u}(t-s+1), \ldots, \boldsymbol{u}(t)\}$

And the data set for iteration $t+1$ becomes,

$\{\boldsymbol{u}(t-s+2), \ldots, \boldsymbol{u}(t+1)\}$

Since the old data are discarded as t increases, this technique intuitively can follow well with the time changing parameters. And training multiple epochs on each sliding data set further enhances this tracking capability. When the size of the sliding window is set to one, $s=1$, the training method is equivalent to normal LMS. Normally, this size is set to the number of inputs to the ADALINE. A bigger sliding window, however, tends to stabilize the learning process. More details will be discussed in the next section. Figure 5 summarizes the training procedure based on this sliding window technique, called LMS-SW.

GADLINE **Training Algorithm II**

Let s be the size of sliding window
Let n_e be the number of epochs
Let $t = s$
Data set of current sliding window is
  $\boldsymbol{U} = \{\boldsymbol{u}(0), \ldots, \boldsymbol{u}(s-1)\}$, $\mathbf{D} = \{d(0), \ldots, d(s-1)\}$
Start with a randomly chosen weight vector $\boldsymbol{w}(t-1)$
While % online version is an infinite loop
  Let $\boldsymbol{u}(t) = (u_1, .., u_m)$ be the next input vector, for which $d(t)$ is the desired output
  Slide window by one unit and update $\boldsymbol{U}$, $\mathbf{D}$
    $\boldsymbol{U} = \{\boldsymbol{U}(2\sim s), \boldsymbol{u}(t)\}$, $\mathbf{D} = \{D(2\sim s), d(t)\}$
  $\boldsymbol{w}'(0) = \boldsymbol{w}(t-1)$
  For $j = 1$ to n_e
    For $i = 1$ to s
      Calculate $e(i) = \mathbf{D}(i) - \boldsymbol{U}^T(i)\boldsymbol{w}'(i-1)$
      $\Delta \boldsymbol{w} = \eta e(i)\boldsymbol{U}(i)$
      Adjust the weight vector to
      $\boldsymbol{w}'(i) = \boldsymbol{w}'(i-1) + \Delta \boldsymbol{w}$
    end for i
    $\boldsymbol{w}'(0) = \boldsymbol{w}'(s)$
  end for j
  $\boldsymbol{w}(t) = \boldsymbol{w}'(s)$
  Increment t
End while

Fig. 5. The LMS-SW algorithm.

In the SWLMS algorithm, the size of the sliding window and the number of epochs on each data set are two tuning parameters. The larger the sliding window the slower is the convergence and the larger the number of epochs the faster the convergence and thus better parameter tracking capability. Large sliding window and increased number of epochs also increase the computational efforts in the multiples of sn_e, though today's computer is more powerful than enough. More discussion on this is given in Section 5 on simulation results.

4. IDENTIFICATION USING GADALINE

As can be seen earlier, GADALINE is a simple linear neural network which can be trained to adaptively model a linear or near-linear system. Adaptive training (also called online/incremental training, as opposed to batch training) can be used so that any change in parameters of the system also changes the weights of the neural network. Thus, at any instant of time, GADALINE can track changes in the parameters of a linear time varying systems. The MTLMS is a pure adaptive training algorithm, but the SWLMS is a hybrid type training algorithm, which adapts from window to window, but training on each window should be taken as a batch. In this section, GADALINE application to the identification of the type of systems given in Section 2 will be considered.

The idea is to use a GADALINE in place of the block of Fig. 2 in the dashed line box. For system identification, GADALINE can be configured in two ways, as shown in Fig. 6:

i) Parallel model, and

ii) Series-parallel model

as proposed by (Narendra and Parthasarathy 1990). In both models, the system input u_S is passed through a TDL to feed to the linear combiner of the ADALINE. Model i) is a recurrent version of the ADALINE (Widrow and Lehr 1990) and it takes the ADALINE's output y (as predicted system output) to pass through a TDL as feedback. Model ii) takes the measured system output y_S to pass through a TDL as feedback.

As a linear function approximator, model i) has limited performance. In this paper, we only consider model ii). Then the input vector to the GADALINE can be set up as,

$$\boldsymbol{u}(k) = [u_S(k-1) \ \ldots \ u_S(k-m) - y_S(k-1) \ \ldots \ -y_S(k-n)]^T \quad (7)$$

and the weight vector

$$\boldsymbol{w}(k) = [w_1(k) \ w_2(k) \ \ldots \ w_m(k) \ w_{m+1}(k) \ \ldots \ w_{m+n}(k)]^T \quad (8)$$

corresponds to an estimate of the system parameters,

$$\boldsymbol{\theta} = [b_1 \ b_2 \ \ldots \ b_m \ a_1 \ a_2 \ \ldots \ a_n]^T$$

in the linear system model (2).

Then, we have the following system identification procedure.

1) **Experiment design**: As in traditional system identification, the input signal should be selected such that it is "rich" enough to excite all modes of the system. This is called persistent exciting. For linear systems, input signal should excite all frequencies and amplitude is not so important. Commonly used persistent exciting signals that can be tried are: random signal, multi-sine signal and random binary sequence signal (RBS) – the signal switches between two levels with given probability.

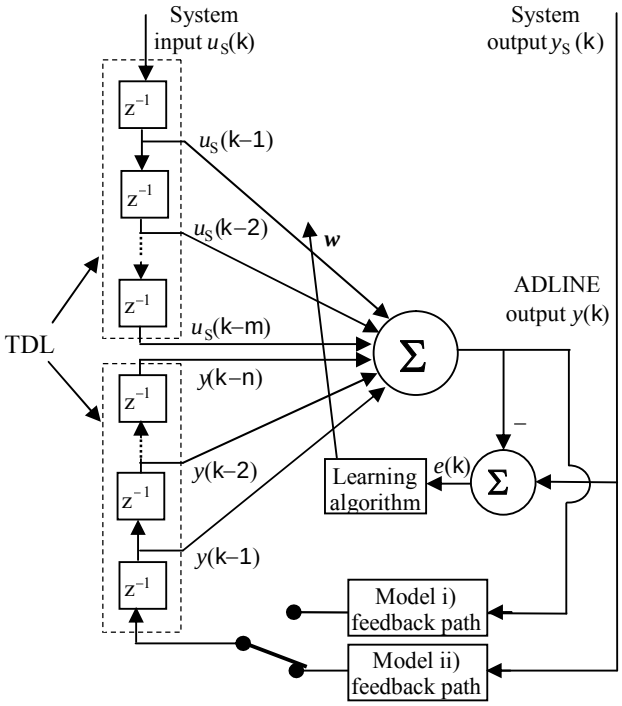


Fig. 6. GADLINE configuration.

2) **Data collection**: Measurement of input-output data $u_S(k)$ and $y_S(k)$ and to set up $\boldsymbol{u}(k)$.

3) **Parameter estimation:** Apply one of the two training algorithms MTLMS or SWLMS to estimate the system parameters $\boldsymbol{\theta}(k) = \boldsymbol{w}(k)$.

The setup for MTLMS parameter estimation is straight forward. Only the setup for SWLMS is discussed here.

The starting time index k_0 should be $s+n$, s – the size of the sliding window and n – the system order. Then the data set at time, $k \geq k_0$, is represented as an $n+m$ by s matrix,

$$\boldsymbol{U} = (\boldsymbol{U_1}, \boldsymbol{U_2}, \ldots, \boldsymbol{U_s})$$

$$= \begin{bmatrix} u_S(k\text{-s}) & \ldots & u_S(k\text{-}1) \\ \vdots & & \vdots \\ u_S(k\text{-s-}m+1) & \ldots & u_S(k\text{-}m) \\ y_S(k\text{-s}) & \ldots & y_S(k\text{-}1) \\ \vdots & & \vdots \\ y_S(k\text{-s-}n+1) & \ldots & y_S(k\text{-}n) \end{bmatrix}$$

where $\boldsymbol{U_s} = \boldsymbol{u}(k)$ as given by (7). In the next window, as the next input vector $\boldsymbol{u}(k+1)$ becomes available, $\boldsymbol{U}$ becomes,

$$\boldsymbol{U} = (\boldsymbol{U_2}, \ldots, \boldsymbol{U_s}, \boldsymbol{u}(k+1))$$

where the oldest input vector $\boldsymbol{U_1}$ is discarded.

5. Simulation Results

Presented in this section are simulation results obtained for a classic representative example. Comparison on the convergence speed and capability of tracking time varying systems' parameters among the normal LMS training algorithm, the proposed GLMS and the LMS-SW training algorithms is conducted.

Consider a second order linear SISO system described by the following difference equation,

$$y(k) - 1.5\,y(k-1) + 0.7\,y(k-2) = 1.0\,u(k-1) + 0.5\,u(k-2)$$

The system is to be simulated with an RBS input signal switching between -1 and 1 with equal probability from k=1 to k=1200, the first 100 samples of u(k) are shown in Fig. 7. Assume at k=800, that system parameters b_1 and b_2 are switched to 0.8 and 0.7, respectively. Figure 8 shows the comparison of parameter trajectories obtained by GLMS algorithm with different α values. When α = 0, the GLMS is the normal LMS algorithm.

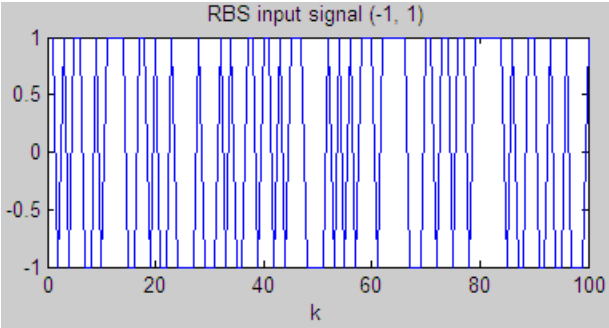


Fig. 7. RBS input signal.

From Fig. 8, we can observe that when α is set to 0.5, the average convergence speed of all parameters is doubled and the trajectories are smoother after convergence than the case where no momentum is added, *i.e.*, α=0. The capability of tracking time varying parameters is improved when α=0.5 as opposed to α=0. When α = 0.8, we start to see parameter adjustment overshooting during the convergence period with almost the same convergence speed.

Figure 9 shows the comparison of parameter trajectories obtained by LMS-SW algorithm with different number of epochs and window size combinations.

Compared with results by GLMS, the results by LMS-SW showed much faster convergence speed and much better capability of tracking time varying parameters. The bigger sliding window Fig. 9(b) vs. (a) did not show apparent improvement on the convergence speed, but did provide much smoother parameter trajectories. By setting the number of epochs to 4, Fig. 9(c) showed both improved convergence speed and smoother trajectories, vs. Fig. 9(a) and (b).

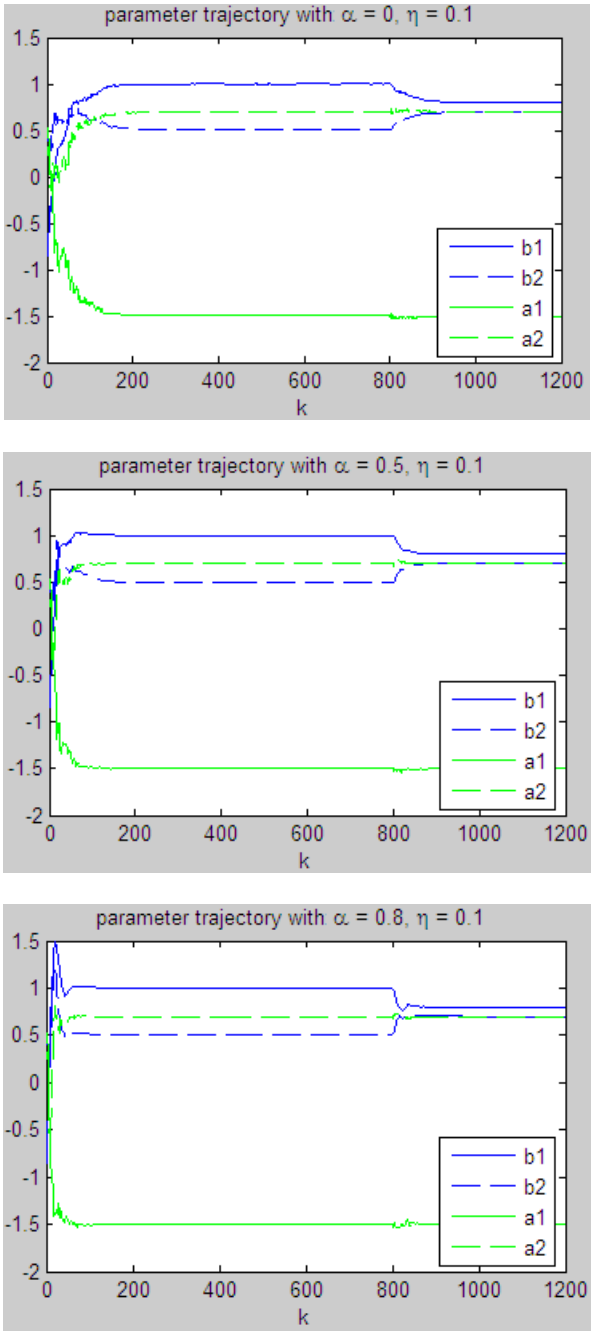
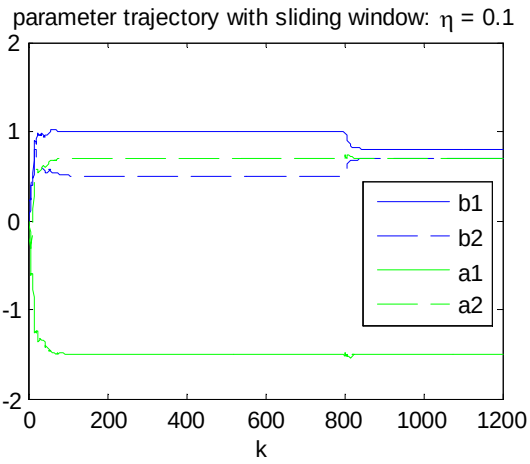
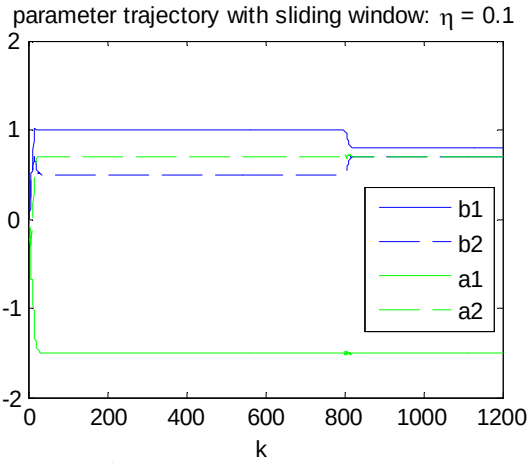


Fig. 8. GLMS trajectories for different α.

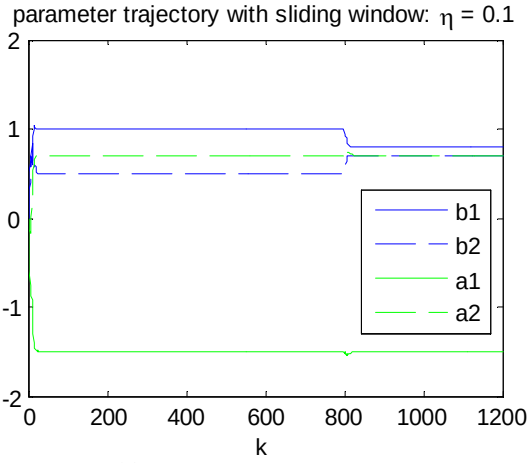
The system is also simulated with the output contaminated with a white noise signal of variance $\sigma^2 = 0.05$. Figure 10 shows the performance comparison on results by GLMS. Increased α improves performance while makes the parameter trajectories more fluctuating. Note that the learning rate parameter η is now 5 times smaller due to the added noise to the system's output data.



(a) n_e =1, $s = n_a+n_b$, η = 0.1



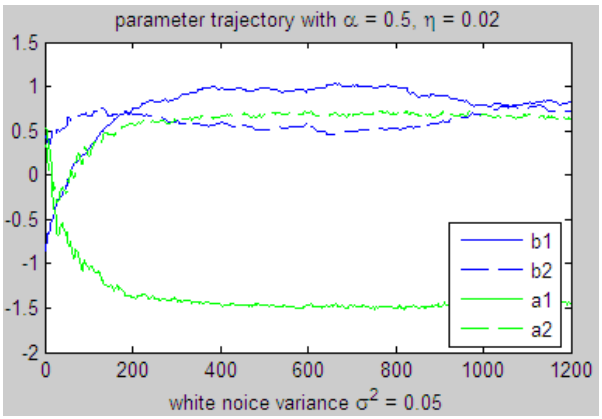
(b) n_e =1, $s = 4(n_a+n_b)$, η = 0.1



(c) n_e =4, $s = n_a+n_b$, η = 0.1

Fig. 9. LMS-SW trajectories with different n_e and s.

Figure 11 shows the performance comparison on results by LMS-SW. Due to the added noise to the system's output data, parameter trajectories are more fluctuating. Increased sliding window size smoothes the curves, Fig. 11(b) vs. (a). Again, the increased number of epochs speeds up the convergence, but it also tracks the noise mixed in the system's output, as shown in Fig. 11(c). Therefore, tradeoff on the two tuning parameters should be considered.



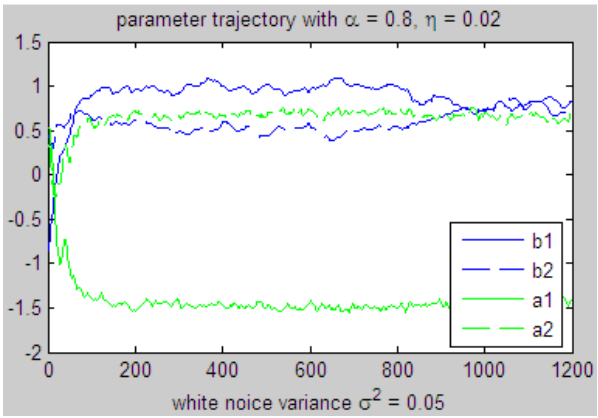
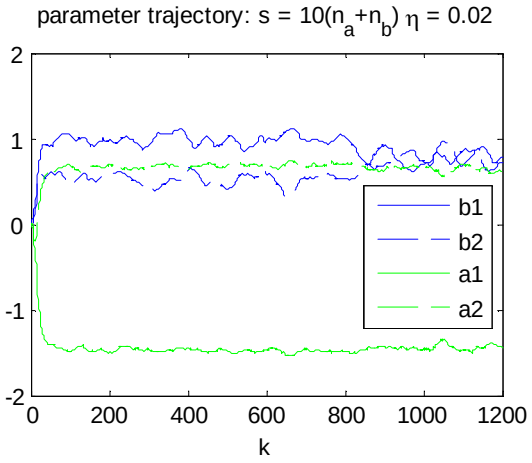


Fig. 10. GLMS trajectories for output with noise.

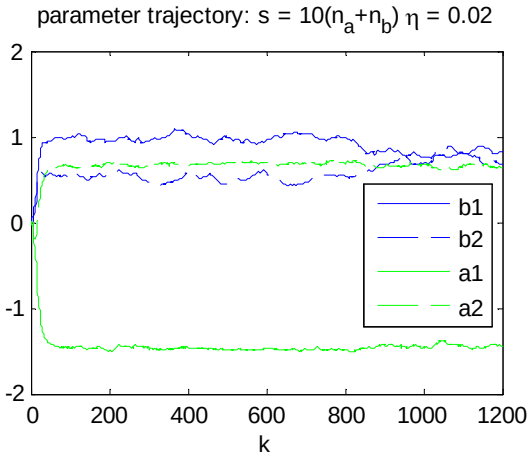
6. CONCLUSIONS

The ADALINE neural network is generalized and used for adaptive system identification of linear time varying systems. Two training methods are proposed and compared via simulation. The first method introduces a momentum term into the LMS learning algorithm. The momentum term helps speed up the learning process and reduce the zigzag effect during the convergence period of learning. However, the momentum should be turned off once the error falls below a given small number so that the weights are kept smooth after convergence. The second training algorithm uses each set of samples for several epochs obtained from a sliding window of the system's input output data. These simple generalized LMS learning algorithm with very low computational complexity make the GADALINE based identification method suitable for real time application. The increased convergence speed improves the capability of tracking time varying parameters. Simulations results show that with an appropriate momentum constant α value, GADALINE offers a better performance as compared to original LMS learning and the sliding window training algorithm provides even better performance when the

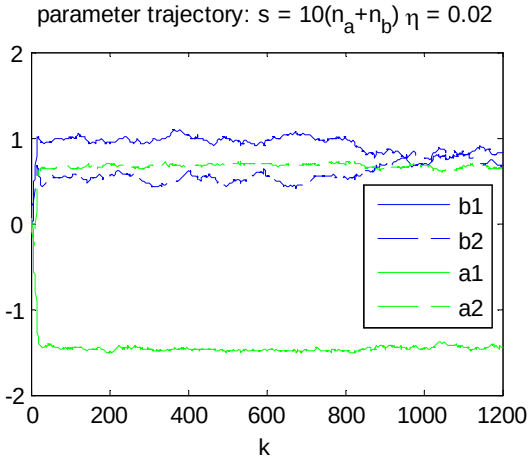
number of epochs and window size are tuned to appropriate values. A shortcoming is the curse of dimensionality as for all neural networks. Thus the method does not seem suitable for high order systems.



(a) n_e =1, $s = 10(n_a+n_b)$, $\eta = 0.02$



(b) n_e =1, $s = 20(n_a+n_b)$, $\eta = 0.02$



(c) n_e =4, $s = 20(n_a+n_b)$, $\eta = 0.02$

Fig. 11. LMS-SW trajectories for output with noise.

REFERENCES

[1] Abonyi, J. and F. Szeifert, "System Identification using Delaunay Tessellation of Self-Organizing Maps," in *Proc. 6th International Conference on Neural Networks and Soft Computing*, Poland, Zakopane, June 11-15, 2002

[2] Atencia, M. and G. Sandoval, "Gray Box Identification with Hopfield Neural Networks," *Revista Investigacion Operacional*, Vol. 25, No. 1, pp, 54-60, 2004

[3] Bhama, S. and H. Singh, "Single Layer Neural Network for Linear System Identification Using Gradient Descent Technique," *IEEE Trans. on Neural Networks*, Vol. 4, No. 5, pp.884-888, 1993.

[4] Chu, S. R., R. Shoureshi and M. Tenorio, "Neural networks for system identification", IEEE Control Systems Magazine, Apr., 31-34, 1990.

[5] Gretton, A., A. Doucet, R. Herbrich, P. Rayner, B. Schölkopf, "Support Vector Regression for Black-Box System Identification," in *Proc. 11th IEEE Workshop on Statistical Signal Processing*, 2001.

[6] Haykin, S., *Neural Networks, A Comprehensive Foundation*, Second Edition, Prentice Hall, 1999.

[7] Hopfield, J., "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat'l. Acad. Sci. USA*, 79:2554–2558, 1982.

[8] Ljung, L., *System Identification - Theory for the User*, Second Edition, Prentice-Hall, 1999.

[9] Mehrotra, K., C. Mohan and S. Ranka, *Elements of Artificial Neural Networks*, MIT press, 1997

[10] Narendra, K. S. and K. Parthasarathy, "Identification and Control of Dynamical Systems using Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 1, pp 1-27, 1990.

[11] Qin, S.Z., H.T. Su and T.J. McAvoy, "Comparison of four neural net learning methods for dynamic system identification," *IEEE Trans. on Neural Networks*, vol. 2, pp. 52 – 262, 1992.

[12] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. I, MIT Press, Cambridge, 1986b.

[13] Sjöberg, J., H. Hjalmerson and L. Ljung, "Neural Networks in System Identification," *Preprints 10th IFAC symposium on SYSID*, Copenhagen, Denmark. Vol.2, pp. 49-71, 1994.

[14] Söderström, T. and P .Stoica, *System Identification*, Prentice Hall, Englewood Cliffs, NJ. 1989.

[15] Valverde, Ricardo, "Dynamic systems identification using RBF neural networks, " *Universidad Carlos III de Madrid Technical report*, 1999.

[16] Widrow, B. and M. A. Lehr, "30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proc. IEEE*, vol. 78, no 9, pp. 1415-1442, 1990.

Wenle Zhang received his B.S. and M.S. from Shandong University of Science and Technology in 1983 and Shanghai University of Science and Technology in 1986, respectively. Then He worked as an instructor in the Department of Computer Engineering at Shanghai University of Science and Technology until 1990. He received his Ph.D. in Electrical Engineering from Ohio University in 2000. After received his Ph.D., Dr. Zhang worked for Lucent Technologies as a software engineer until March 2001. Prior to pursuing his Ph.D. at Ohio University, he was employed with Rockwell Automation, a famous industrial controls supplier, as a control engineer for more than 5 years. He has experience in PLC application, industrial control networking, and control software development.

Since spring of 2001, Dr. Zhang has been an assistant professor in the School of Electrical Engineering and Computer Science at Ohio University. Dr. Zhang's current research interests include system identification and control, neural networks and intelligent control, discrete event dynamic systems and manufacturing systems.