



## A revisit to block and recursive least squares for parameter estimation

Jin Jiang <sup>a,\*</sup>, Youmin Zhang <sup>b,1</sup>

<sup>a</sup> *Department of Electrical and Computer Engineering, The University of Western Ontario, London, Ontario, Canada N6A 5B9*

<sup>b</sup> *Department of Computer Science and Engineering, Aalborg University Esbjerg, Niels Bohrs Vej 8, 6700 Esbjerg, Denmark*

Received 17 January 2003; received in revised form 25 January 2004; accepted 20 May 2004

---

### Abstract

In this paper, the classical least squares (LS) and recursive least squares (RLS) for parameter estimation have been re-examined in the light of the present day computing capabilities. It has been demonstrated that for linear time-invariant systems, the performance of blockwise least squares (BLS) is always superior to that of RLS. In the context of parameter estimation for dynamic systems, the current computational capability of personal computers are more than adequate for BLS. However, for time-varying systems with abrupt parameter changes, standard blockwise LS may no longer be suitable due to its inefficiency in discarding “old” data. To deal with this limitation, a novel sliding window blockwise least squares approach with automatically adjustable window length triggered by a change detection scheme is proposed. Two types of sliding windows, rectangular and exponential, have been investigated. The performance of the proposed algorithm has been illustrated by comparing with the standard RLS and an exponentially weighted RLS (EWRLS) using two examples. The simulation results have conclusively shown that: (1) BLS has better performance than RLS; (2) the proposed variable-length sliding window blockwise least squares (VLSWBLS) algorithm can outperform RLS with forgetting factors; (3) the scheme has both good tracking ability for abrupt parameter changes and can ensure the high accuracy of parameter estimate at the steady-state; and (4) the computational burden of

---

\* Corresponding author. Tel.: +1 519 661 2111x88320; fax: +1 519 850 2436.

E-mail addresses: [jjiang@uwo.ca](mailto:jjiang@uwo.ca) (J. Jiang), [ymzhang@cs.auc.auc.dk](mailto:ymzhang@cs.auc.auc.dk) (Y. Zhang).

<sup>1</sup> Tel.: +45 7912 7741; fax: +45 7912 7710.

VLSWBLS is completely manageable with the current computer technology. Even though the idea presented here is straightforward, it has significant implications to virtually all areas of application where RLS schemes are used.

© 2004 Elsevier Ltd. All rights reserved.

*Keywords:* Blockwise least squares (BLS); Recursive least squares (RLS); Sliding window blockwise least squares (SWBLS); Variable-length window; Change detection

---

## 1. Introduction

Since Karl Gauss proposed the technique of least squares around 1795 to predict the motion of planets and comets using telescopic measurements, least squares (LS) and its many variants have been extensively applied to solving estimation problems in many fields of application [6,7,9,11–13]. The classical formula of least squares is in *batch* or *block* form, meaning that all measurements are collected first and then processed simultaneously. Such a formula poses major computational problems since the computational complexity is in the order of  $O(N^3)$  which grows continuously with the number of data collected, where  $N$  is the number of parameters to be estimated. Because of its non-iterative nature, block/batch LS algorithms are often regarded to be less desirable for on-line applications.

To increase the efficiency of LS algorithms, a recursive variant, known as recursive least squares (RLS), has been derived. The computational complexity of RLS is in the order of  $O(N^2)$ . Although RLS algorithm is theoretically equivalent to a block LS, it suffers from the following shortcomings: (1) numerical instability due to round-off errors caused by its recursive operations in a finite-word length environment; (2) slow tracking capability for time-varying parameters; and (3) sensitive to the initial conditions of the algorithm.

To improve the numerical properties, several numerically more robust variants based on QR decomposition, U-D factorization and singular value decomposition (SVD) implementations of RLS algorithms have been developed [2,9,11]. To improve the tracking capability of RLS, techniques based on variable/time-varying forgetting factors [5,11,15], covariance matrix re-setting/modification [6,8,14], and sliding window [3,4,10,13] have been developed.

It is true that the computational complexity of an estimation algorithm was the main concern ten to twenty years ago, but it is no more! To date, benefiting from rapid advance in computer technology, the computational complexity of LS becomes much less an issue. One may choose to view the advancement in computer technology being relative to the complexity of the problem that we are able to solve with RLS. In the context of dynamic system identification, the fact of matter is that for any practical system with more than two dozen of unknown parameters, the requirement on the system input to be persistently exciting is so high that most of the LS-based schemes will no longer provide accurate parameter estimate anyway. For systems with fewer parameters, the computational power of existing personal computers will be far more adequate for block LS. It is under this circumstance that the research reported in this paper has been carried out.

By simply extending the block least squares to blockwise least squares (BLS), the classical LS algorithm can be easily extended to on-line applications. Leaving the computational issues aside,

this paper has shown that BLS leads to superior performance to standard RLS in time-invariant parameter estimation problems.

To improve the tracking capability of BLS for estimating parameters in time-varying systems, certain techniques to effectively discard “old” data is necessary. In view of this, a new sliding-window blockwise least squares (SWBLS) algorithm is developed in this paper. Furthermore, to obtain parameter estimates with both good parameter tracking and high accuracy at the steady-state for systems with abruptly changed parameters, an automatically adjustable window length sliding window BLS technique (abbreviated as VLSWBLS) is also proposed. The window length adjustment is triggered by a parameter change detection scheme. In addition, the added advantages of the developed algorithm are: (1) simpler to implement and (2) no need to specify the initial conditions. The use of forgetting factor within the sliding window has also been investigated to further increase the tracking capability. The paper has shown that the VLSWBLS can significantly outperform the existing RLS algorithms with forgetting factors, even for time-varying systems.

The performance of the newly developed algorithms has been evaluated in comparison with that of the existing ones for estimating parameters of time-invariant and time-varying systems, and the superior performance has been obtained in all cases.

It is important to emphasize also that the results presented in this paper have tremendous implications. The paper may prompt those who are relying on RLS now to re-think about other options in their own field of applications.

## 2. Review of block and recursive least squares

Let us consider a discrete dynamic system described by

$$\begin{aligned}
 y(k) &= \boldsymbol{\phi}^T(k-1)\boldsymbol{\theta} \\
 z(k) &= y(k) + v(k) \\
 \boldsymbol{\phi}^T(k-1) &= [-z(k-1), \dots, -z(k-n), u(k-1), \dots, u(k-m)] \\
 \boldsymbol{\theta}^T &= [a_1, \dots, a_n, b_1, \dots, b_m]
 \end{aligned} \tag{1}$$

where  $u(k)$  and  $y(k)$  are the system input and output, respectively.  $z(k)$  is measured system output and  $v(k)$  is a zero-mean white Gaussian noise term which counts for measurement noise and modelling uncertainties.  $\boldsymbol{\phi}^T(k-1)$  and  $\boldsymbol{\theta}$  are the information vector and the unknown parameter vector, respectively. The parameters in  $\boldsymbol{\theta}$  can either be constant or subject to infrequent jumps. The number of parameters to be estimated is denoted as  $N = n + m$ .

### 2.1. Block and blockwise least squares

To estimate the parameter vector,  $\boldsymbol{\theta}$ , using all available measurements up to the current instant,  $j = 1, 2, \dots, k$ , one can first define the following cost function:

$$J(\boldsymbol{\theta}, k) = \sum_{j=1}^k [z(j) - \boldsymbol{\phi}^T(j-1)\boldsymbol{\theta}]^2 \tag{2}$$

Differentiating  $J(\theta, k)$  with respect to  $\theta$  and setting the derivative to zero will lead to the well-known LS solution [6,11]:

$$\hat{\theta} = \left[ \sum_{j=1}^k \phi(j-1)\phi^T(j-1) \right]^{-1} \left[ \sum_{j=1}^k \phi^T(j-1)z(j) \right] \quad (3)$$

where  $\hat{\theta}$  is known as the least squares estimate of  $\theta$ .

Since  $\hat{\theta}$  is obtained using the sample data up to the time instant  $k$ , it is often denoted as  $\hat{\theta}(k)$ . An alternative compact matrix-vector form can be written as follows:

$$\hat{\theta}(k) = [\Phi_k^T \Phi_k]^{-1} [\Phi_k^T \mathbf{z}_k] \quad (4)$$

where

$$\begin{aligned} \mathbf{z}_k &= [z(1) \ z(2) \ \dots \ z(k)]_{k \times 1}^T \\ \Phi_k &= \begin{bmatrix} \phi^T(0) \\ \phi^T(1) \\ \vdots \\ \phi^T(k-1) \end{bmatrix} = \begin{bmatrix} -z(0) & \dots & -z(1-n) & u(0) & \dots & u(1-m) \\ -z(1) & \dots & -z(2-n) & u(1) & \dots & u(2-m) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -z(k-1) & \dots & -z(k-n) & u(k-1) & \dots & u(k-m) \end{bmatrix}_{k \times N} \end{aligned}$$

If this formulation of LS is used for on-line parameter estimation, one has to re-calculate the Eq. (4) each time when a new measurement becomes available. This style of implementation is referred to as BLS in this paper. To circumvent the problem of re-calculating Eq. (4), a recursive version of the above scheme has been derived.

## 2.2. Recursive least squares

The basic idea behind a RLS algorithm is to compute the parameter update at time instant  $k$  by adding a correction term to the previous parameter estimate  $\hat{\theta}(k-1)$  once the new information becomes available, rather than re-calculating Eq. (4) with the entire measurements. Such reformulation has reduced the computational requirement significantly, making the RLS extremely attractive in the last three decades for on-line parameter estimation applications.

Derivation of RLS algorithms can be found in many textbooks and papers (see e.g. [6,11]). For the sake of easy reference, a typical RLS algorithm is presented here:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)\phi(k-1)\varepsilon(k) \quad (5)$$

$$\varepsilon(k) = z(k) - \phi^T(k-1)\hat{\theta}(k-1) \quad (6)$$

$$P(k) = P(k-1) - \frac{P(k-1)\phi(k-1)\phi^T(k-1)P(k-1)}{1 + \phi^T(k-1)P(k-1)\phi(k-1)} \quad (7)$$

where  $\varepsilon(k)$  is the one-step ahead prediction error. Thus, the amount of correction is proportional to the prediction error.

It can be seen that due to its *recursive* nature, the complexity of the RLS has been reduced considerably from  $O(N^3)$  in BLS to  $O(N^2)$  in each estimate update. For more details about the block and recursive least squares algorithms, readers are referred to, for example, [6,11–13].

### 3. A variable-length sliding window blockwise least squares algorithm

#### 3.1. Sliding window blockwise least squares formulation

Both of the above-discussed BLS and RLS algorithms can lead to the convergence of the parameter estimate to their true value for time-invariant systems. However, when LS is applied to non-stationary environments, e.g. system identification and parameter estimation in the presence of unknown parameter changes, forgetting factor or finite length sliding window techniques have been widely used in association with the RLS. Such techniques can also be incorporated in BLS.

In addition to enhancing the tracking capability of the BLS or RLS algorithm, the sliding window technique also has an inherent advantage to keep the computational complexity to a fixed level by replacing the growing dimension in  $\Phi_k$  with a fixed and lower dimension matrix. To further improve the tracking capability of the algorithm and to exploit the advantages of both sliding window and forgetting factor techniques, an exponential weighting technique can be used within the sliding window. Hence, a modified cost function of Eq. (2) can be described as follows:

$$J_L(\theta, k) = \sum_{j=k-L+1}^k \lambda^{k-j} [z(j) - \phi^T(j-1)\theta]^2 \tag{8}$$

where  $L$  denotes the length of the sliding window.  $\lambda (0 < \lambda \leq 1)$  is the forgetting factor to exponentially discard the “old” data within the length of the window. The LS problem based on parameter estimation of  $\theta$  is then to minimize the above cost function with the most recent data of size  $L$ .

Following the same derivation as in the BLS, a sliding exponentially weighted window blockwise least squares (SEWBLS) algorithm can be obtained as follows:

$$\hat{\theta}_L(k) = \left[ \sum_{j=k-L+1}^k \lambda^{k-j} \phi(j-1)\phi^T(j-1) \right]^{-1} \left[ \sum_{j=k-L+1}^k \lambda^{k-j} \phi^T(j-1)z(j) \right] \tag{9}$$

or, in a compact matrix-vector form as

$$\hat{\theta}_L(k) = [(\Phi_{k-L+1}^k)^T A_{k-L+1}^k \Phi_{k-L+1}^k]^{-1} [(\Phi_{k-L+1}^k)^T A_{k-L+1}^k \mathbf{z}_{k-L+1}^k] \tag{10}$$

where  $\Phi_{k-L+1}^k$  is similarly defined as in Eq. (4), and  $A_{k-L+1}^k$  is a  $L \times L$  diagonal matrix with diagonal elements being the forgetting factors,  $\lambda^{L-1}, \lambda^{L-2}, \dots, \lambda^0$ .

The performance of the above algorithm will depend on the length of the sliding window. For time-invariant systems, the longer the window length, the higher the estimation accuracy. However, for a system with abrupt parameter changes, to achieve fast tracking of the changed

parameters, the window length should be adjusted accordingly so that the out-of-date information from the past measurements can be discarded effectively to assign a relatively heavier weight on the latest measurement.

### 3.2. Automatic window length adjustment

To automatically initiate the window length adjustment, a change detection mechanism and a window length adjustment strategy must be developed.

#### 3.2.1. Change detection mechanism

There are many ways that one can detect parameter changes, an easy way to implement change detector can be just based on the prediction error of the system within the sliding-window:

$$\mathbf{e}(k) = \mathbf{z}_{k-L+1}^k - \hat{\mathbf{z}}_{k-L+1}^k = \mathbf{z}_{k-L+1}^k - \Phi_{k-L+1}^k \hat{\boldsymbol{\theta}}_L(k) \quad (11)$$

where  $\mathbf{z}_{k-L+1}^k = [z(k-L+1) \ z(k-L) \ \dots \ z(k)]^T$  and  $\hat{\mathbf{z}}_{k-L+1}^k$  is its estimated counterpart.

A decision to decrease the window length at time  $k = k_D$  can be made if the following averaged sliding-window detection index:

$$d(k) = \frac{1}{M} \sum_{i=k-M+1}^k \mathbf{e}^T(i)\mathbf{e}(i) \quad (12)$$

exceeds a pre-set threshold  $\rho$

$$\begin{array}{l} H_1 \\ d(k) > \rho \\ H_0 \end{array} \quad (13)$$

where  $H_0 = \{\text{No change in the system parameters}\}$ ,  $H_1 = \{\text{Parameter changes have occurred in the system}\}$ .  $M$  is the window length for calculating the detection index. If the change is detected at time  $k_D$ ,  $k_D$  will be referred to as the *detection time*.

It should be noted that Eq. (12) is only a very simple scheme for change detection. There are many other more sophisticated change detection techniques available. Detailed discussion will be beyond the scope of this paper. Interested readers are referred to [1,7] for more information.

#### 3.2.2. Length adjustable sliding window

To improve the tracking ability of BLS, a new window length adjustment strategy has been developed. The essence of the proposed approach lies in its ability to quickly shrink the window length, should a change in the system parameters be detected. By using a shorter window length, it allows the algorithm to track the parameter changes quickly. From the onset of this window shrinking, the window length will be progressively expanding and returning to the original width to maintain the steady-state performance of the algorithm. Such a fast shrinking and gradual expanding window makes this approach suitable for a wide range of applications.

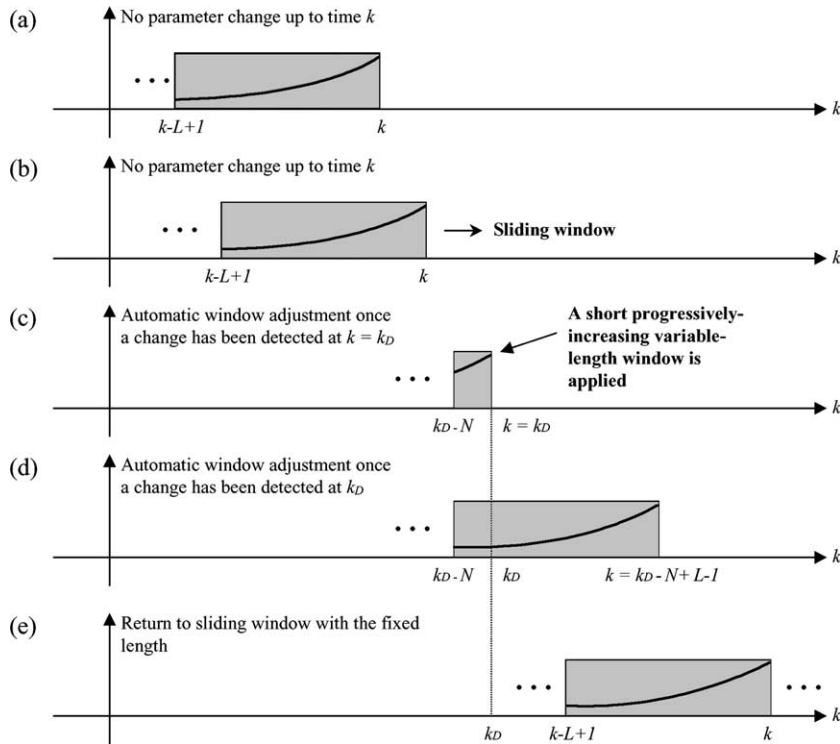


Fig. 1. Demonstration for automatic adjustment of the window length.

To illustrate this concept clearly, consider five snapshots shown in Fig. 1. If there are no detected changes, a sliding window of length  $L$  is used. This situation is shown in Fig. 1(a) and (b). Suppose some parameters of the system have undergone changes, and they are detected at  $k_D$ . At this point, the window length is reduced drastically to discount the measurements in the subsequent LS calculations. This case is illustrated in Fig. 1(c). As new data becomes available, the window size will increase step-by-step until it again reaches the full length  $L$ . Thereafter, the original sliding window scheme resumes as shown in Fig. 1(d) and (e).

As we know, the window size for a LS calculation has to be greater than the number of parameters to be estimated. In this case, the minimum window length will be  $N + 1$ . Furthermore, at the instant of the window shrinking,  $\Phi_{k-L+1}^k$  may still contain some data prior to the parameter change. One way to further improving the performance of this scheme is to use an exponential weighting within this shrunk window. However, our experience indicates that, because of the relatively short window length at the initial stage of shrinking, the effect of such weighting is not significant.

#### 4. Illustrative examples

To illustrate the concepts presented in this paper, two systems with respective time-invariant and abrupt change parameters have been used in this section.

The purpose of the first example is to compare the performance of a RLS algorithm and a simple BLS. The second example is to demonstrate the capability of the proposed VLSWBL algorithm in comparison with RLS-type algorithms in the presence of parameter changes.

#### 4.1. Example 1: parameter estimation of a time-invariant system

A second order system from [11] is adopted here. The system is described by the following difference equation:

$$\begin{aligned} y(k) &= -a_1y(k-1) - a_2y(k-2) + b_1u(k-1) + b_2u(k-2) \\ z(k) &= y(k) + v(k) \end{aligned} \quad (14)$$

where

$$a_1 = -1.5, \quad a_2 = 0.7; \quad b_1 = 1.0, \quad b_2 = 0.5$$

and  $y(k)$  is the output of the system,  $z(k)$  the measured output, and  $u(k)$  the input.  $v(k)$  is zero-mean white Gaussian noise sequence with variance  $\sigma_v^2 = 0.01$ . The external input  $u(k)$  is also a zero-mean white Gaussian sequence with unit variance  $\sigma_u^2 = 1.0$ , independent of  $v(k)$ .

In an ideal situation, if there were no noise and a perfect initial parameter vector were available for RLS, both RLS and BLS algorithms would have produced a perfect parameter estimation. However, in reality, there are always some measurement noises presented in the system. Furthermore, the initial settings for the estimated parameter vector and covariance matrix cannot be specified accurately. All these facts have contributed negatively on the performance of RLS. It can be seen in Table 1 and Fig. 2 that, compared with the results of RLS, much higher estimation accuracy has been obtained by using the BLS. Note that all curves in Figs. 2, 3 and 5 represent the norm of the parameter estimation error defined by

$$e_\theta(k) = \frac{1}{N_{\text{run}}} \sum_{j=1}^{N_{\text{run}}} \left\| \theta - \hat{\theta}_L^j(k) \right\|_2 \quad (15)$$

These results are drawn based on 100 independent runs, i.e.  $N_{\text{run}} = 100$  for 1000 measurement data points.

Table 1  
Comparison of estimation accuracy between RLS and BLS

Algorithms	Cases	$a_1$	$a_2$	$b_1$	$b_2$	AE	STD
RLS	Noise free	-1.4980	0.6974	0.9990	0.5009	$1.5706 \times 10^{-2}$	$1.9375 \times 10^{-1}$
	With noise	-1.4978	0.6972	0.9980	0.5021	$1.6655 \times 10^{-2}$	$1.9369 \times 10^{-1}$
	$\hat{\theta}_0 = \mathbf{0}$	-1.4986	0.6984	0.9985	0.5018	$9.1101 \times 10^{-3}$	$9.7834 \times 10^{-2}$
	Larger $P_0$	-1.4998	0.6998	0.9990	0.5012	$1.3638 \times 10^{-2}$	$1.8703 \times 10^{-1}$
BLS	Noise free	-1.5000	0.7000	1.0000	0.5000	$8.9727 \times 10^{-15}$	$7.2497 \times 10^{-15}$
	With noise	-1.4999	0.6999	1.0001	0.5000	$1.9741 \times 10^{-3}$	$4.0577 \times 10^{-3}$
True parameters		-1.5	0.7	1.0	0.5		

AE, average error of  $e_\theta(k)$ ,  $k = 1000$ ; STD, standard deviation.

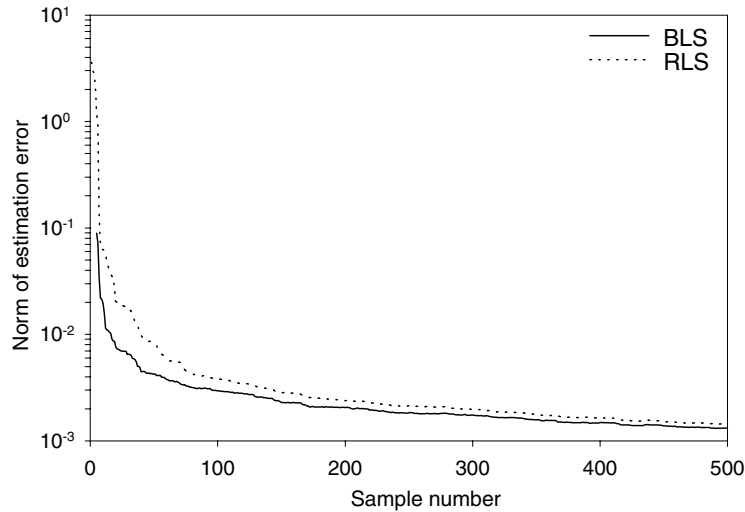


Fig. 2. Trajectories of averaged estimation error.

For the RLS algorithm in Table 1, the first row represents the case with no measurement noise and randomly chosen initial parameter vector  $\hat{\theta}_0$  and covariance matrix  $P_0 = 10I$ , while the second row shows the result with the addition of measurement noise of  $\sigma_v^2 = 0.01$  to the 1st case (this corresponds to a signal-noise-ratio (SNR) of 20 dB for the simulated example). To demonstrate the performance of the RLS with different initial parameter vectors and initial covariance matrices, additional cases are analyzed. The 3rd row presents a case which uses the same initial covariance matrix as in the second case but with  $\hat{\theta}_0 = \mathbf{0}$ , while the 4th row uses the same  $\hat{\theta}_0$  as in 2nd case but with a larger  $P_0 = 10^3I$ . For the BLS algorithm, the two cases shown in the table are under the same conditions as in the first two cases in RLS, respectively. It can be seen that RLS is very sensitive to the initial conditions. However, there is no initial condition issue for BLS. The results in Table 1 clearly indicate that BLS outperforms RLS in every case in terms of estimation accuracy.

To evaluate the sensitivity of the RLS and BLS algorithms to different SNR levels, the system in Eq. (14) has been simulated at different noise levels. The results are shown in Table 2. The input to the system is still a zero-mean white Gaussian sequence with a fixed unit variance of  $\sigma_u^2 = 1.0$ , while the variance of the noise term  $v(k)$  has been adjusted to count for the different levels of

Table 2  
Comparison of estimation accuracy at different SNRs

SNR (dB)	RLS		BLS	
	AE	STD	AE	STD
30.0	$1.5711 \times 10^{-2}$	$1.9375 \times 10^{-1}$	$1.9710 \times 10^{-4}$	$4.0528 \times 10^{-4}$
20.0	$1.6655 \times 10^{-2}$	$1.9369 \times 10^{-1}$	$1.9741 \times 10^{-3}$	$4.0577 \times 10^{-3}$
10.0	$3.8487 \times 10^{-2}$	$1.9383 \times 10^{-1}$	$2.7867 \times 10^{-2}$	$9.6821 \times 10^{-2}$
5.0	$1.7060 \times 10^{-1}$	$1.9387 \times 10^{-1}$	$1.6165 \times 10^{-1}$	$1.3146 \times 10^{-1}$
1.0	$6.7118 \times 10^{-1}$	$1.8474 \times 10^{-1}$	$6.6664 \times 10^{-1}$	$2.1505 \times 10^{-1}$

the measurement noise. It can be observed that for different levels of noise intensity, the BLS can still achieve much better estimation accuracy.

It is also interesting to compare the required CPU time per run (CTPR) for BLS and RLS algorithms running on a relatively older and a relatively newer computer. This figure of merit is measured in terms of the MATLAB function `cputime`. Again, the results are obtained by an average of 100 independent runs using 1000 data points. The CTPRs using the older computer with CPU 200 MHz and RAM 64 KB and the newer computer with CPU 1.7 GHz and RAM 256 KB are listed in Table 3.

It can be seen that using the newer computer, the required CTPR of the BLS is even significantly lower than that of the RLS in the older computer. This observation re-affirms our view on replacing RLS by BLS for achieving better performance.

#### 4.2. Example 2: systems with abrupt parameter changes

In this example, the performance of BLS and RLS for systems with abruptly changed parameters is examined. The system used in Example 1 has been modified by introducing parameter changes at  $k = 500$ . These changes are described by

$$\begin{cases} a_1 = -1.5, & a_2 = 0.7; & b_1 = 1.0, & b_2 = 0.5; & 0 < k < 500 \\ a_1 = -1.2, & a_2 = 0.5; & b_1 = 0.7, & b_2 = 0.3; & 500 \leq k < 1000 \end{cases} \quad (16)$$

##### 4.2.1. Comparison between the proposed and the RLS-type algorithms

To compare the estimation accuracy and the tracking performance, the norm of the estimation error using the variable-length sliding rectangular window blockwise least squares (VLSRWBLs), the exponentially weighted recursive least squares (EWRLS) and the RLS are shown in Fig. 3. In the EWRLS, a constant forgetting factor of  $\lambda = 0.95$  is used for improving the tracking performance. It can be seen that significantly better tracking performance has been achieved by the VLSRWBLs algorithm. The estimate converges quickly to the new parameter values and the proposed scheme also results in a much smaller steady-state estimation error in comparison to the other two RLS algorithms. Compared with the estimation error prior to the change occurrence, almost the same level of the estimation accuracy has been obtained for post-changed parameters. However, RLS without forgetting factor simply could not track the changed parameters. Although the EWRLS can still track the new parameters, the rate of convergence is significantly slower and the steady-state estimation error is relatively larger.

To demonstrate the time history of the parameter estimate, Fig. 4 shows the trajectories of the estimated parameters using these three different algorithms. It can be seen clearly that the proposed approach leads to excellent estimation both in pre- and post-change periods. Excellent

Table 3  
CPU time per run required by RLS and BLS

Algorithms	Older computer	Newer computer
RLS	<u>1.3222</u>	0.1483
BLS	3.6901	<u>0.3356</u>

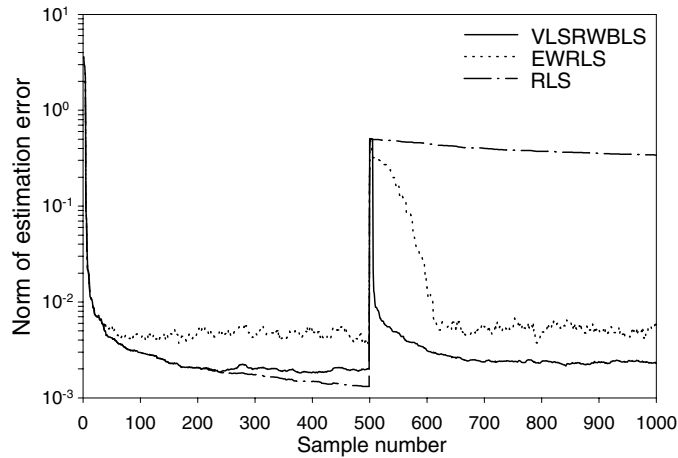


Fig. 3. Comparison between the proposed and RLS-based algorithms.

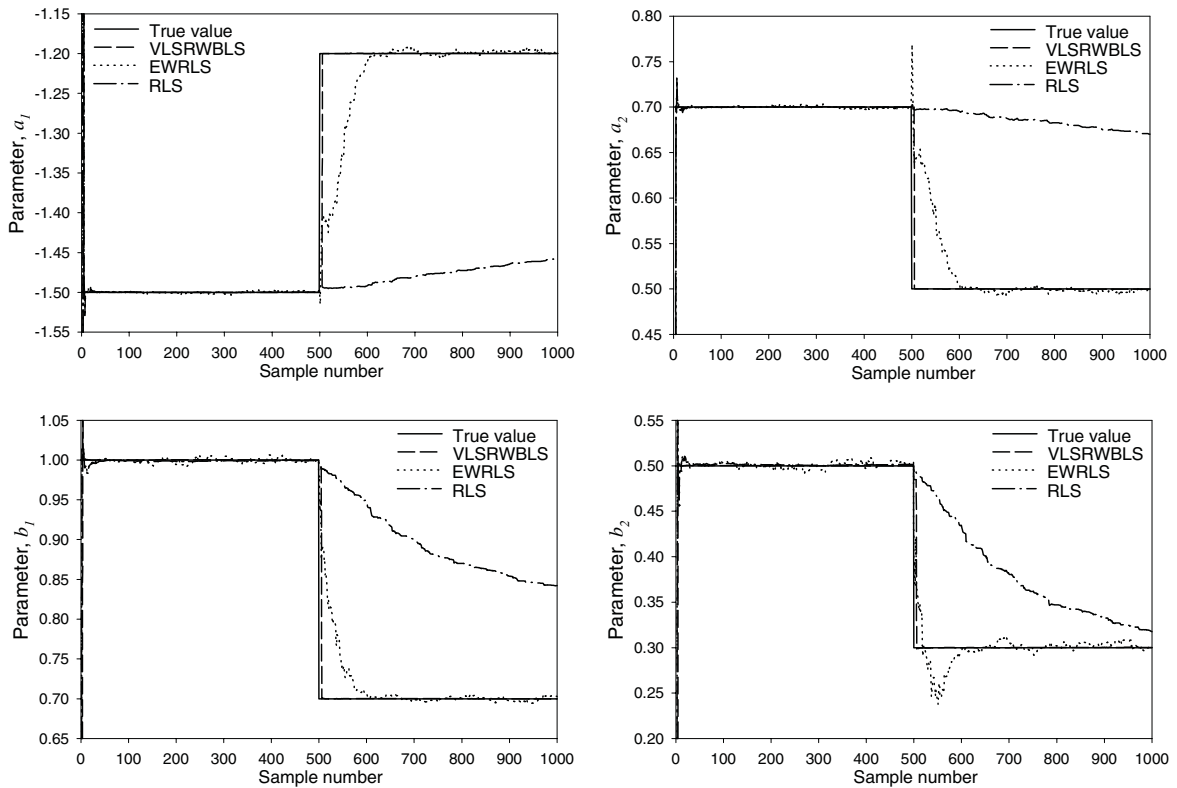


Fig. 4. Parameter estimates via different algorithms.

tracking performance has been achieved for all four parameters. Unfortunately, the RLS scheme cannot track the changed parameters. The EWRLS can provide somewhat acceptable results. The

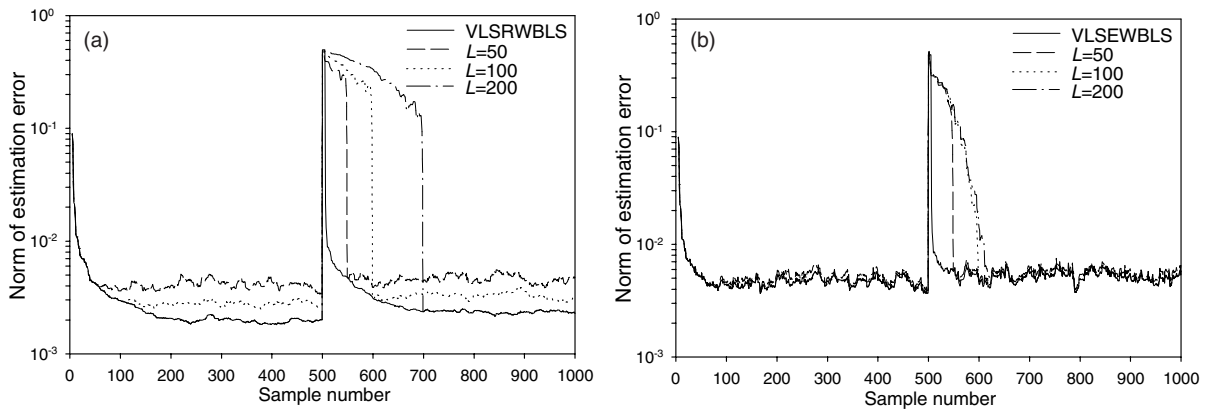


Fig. 5. Comparison between different window shapes and lengths: (a) rectangular window and (b) exponential window.

superiority of the proposed algorithm lies mainly in the effective adjustment of the sliding window length which leads to inherently faster convergence rate.

#### 4.2.2. Performance for fixed- and variable-length windows

To evaluate the effect of window length on the performance of the fixed-length sliding window algorithm and to demonstrate the superiority of the variable-length window, the estimation errors with different fixed window lengths of  $L = 50, 100, 200$  and the one using proposed variable-length sliding window algorithm have been shown in Fig. 5. For comparison purposes, results using rectangular and exponential weighting are shown in Fig. 5(a) and (b), respectively.

For sliding window with different fixed-lengths, it can be seen that the longer the window, the higher the estimation accuracy. However, a longer window leads to larger delay to obtain good post-change parameter estimation. It is clearly observable that, by using a fixed window length, one could not achieve good performance in both the steady-state and the transient periods. However, by using the proposed variable-length sliding window approach, significant improvement in both the transient and the steady-state periods has been obtained using a rectangular window (VLSRWBL, Fig. 5(a)) and an exponential window (VLSEWBL, Fig. 5(b)).

In VLSRWBL, since there is no data prior to the change is included in the sliding window after the change has been detected, equal weights are given to all measurements. Therefore, the same accuracy as the one using a longer fixed-length window of  $L = 200$  has been achieved. Furthermore, due to the automatic adjustment of the window length, much faster convergence and smaller estimation error has been obtained as shown in Fig. 5(a). By comparing the performance between the rectangular and the exponential windows, it can be seen that excellent performance during the transient period has also been obtained by the exponential window. However, since the use of forgetting factor,  $\lambda = 0.95$ , within the sliding window, relatively larger estimation errors at both steady-state periods, before and after the parameter change, are also observed.

The proposed algorithms can also be extended to systems with gradual/incipient parameter changes. However, results for such cases are not shown here due to space reason. Extension of the developed algorithms to numerically more stable implementation, such as QR decomposition,

U-D factorization or SVD, is relatively straightforward, which will not be elaborated further in this paper.

## 5. Conclusions

With today's readily available computing power, one could not help but to think and re-think some of the existing computationally more efficient, numerically less perfect algorithms. In this paper, the standard least squares (LS) parameter estimation approach has been revisited in view of its better accuracy and simpler implementation as compared to the standard recursive least squares (RLS) algorithm. To address the weak tracking ability of BLS, an effective sliding window *blockwise* least squares approach with an adjustable window length has been proposed for extending the LS approach to parameter estimation of systems with abrupt parameter changes. The proposed approach outperforms the RLS-type algorithms significantly and possesses both excellent tracking and steady-state performance for systems with abrupt parameter changes. Simulation results have also shown that the variable length sliding rectangular window algorithm provides the best performance among tracking ability, steady-state estimation accuracy and computational complexity.

The results presented in this paper is significant in the sense that one may need to re-think about the following issue: Should we use *blockwise* or *recursive* least squares? In view of its excellent performance, it is our view that the blockwise LS approaches will play even more important role in signal processing, communication, control and other engineering applications in a very near future in the situations where RLS algorithms have been predominant up to now.

## Acknowledgment

The research was partially supported by the Natural Sciences and Engineering Research Council of Canada.

## References

- [1] Basseville M, Nikiforov I. Detection of abrupt changes: theory and application. Englewood Cliffs, NJ: Prentice-Hall; 1993.
- [2] Baykal B, Constantinides AG. Sliding window adaptive fast QR and QR-lattice algorithms. IEEE Trans Signal Process 1998;46(11):2877–87.
- [3] Belge M, Miller EL. A sliding window RLS-like adaptive algorithm for filtering alpha-stable noise. IEEE Signal Process Lett 2000;7(4):86–9.
- [4] Choi BY, Bien Z. Sliding-windowed weighted recursive least-squares method for parameter estimation. Electron Lett 1989;25(20):1381–2.
- [5] Fortescue TR, Kershenbaum LS, Ydstie BE. Implementation of self-tuning regulators with variable forgetting factors. Automatica 1981;17(6):831–5.
- [6] Goodwin GC, Sin KS. Adaptive filtering, prediction and control. Englewood Cliffs, NJ: Prentice-Hall; 1984.
- [7] Gustafsson F. Adaptive filtering and change detection. West Sussex: Wiley; 2000.

- [8] Jiang J, Cook R. Fast parameter tracking RLS algorithm with high noise immunity. *Electron Lett* 1992;28(22):2042–5.
- [9] Lawson CL, Hanson RJ. Solving least squares problems. Englewood Cliffs, NJ: Prentice-Hall; 1974.
- [10] Liu H, He Z. A sliding-exponential window RLS adaptive algorithm: properties and applications. *Signal Process* 1995;45:357–68.
- [11] Ljung L, Soderstrom T. Theory and practice of recursive identification. Cambridge, MA: MIT Press; 1983.
- [12] Mendel JM. Lessons in estimation theory for signal processing, communications, and control. Englewood Cliffs, NJ: Prentice-Hall; 1995.
- [13] Niedzwiecki M. Identification of time-varying processes. Chichester: Wiley; 2000.
- [14] Park DJ, Jun BE. Self-perturbing recursive least squares algorithm with fast tracking capability. *Electron Lett* 1992;28(6):558–9.
- [15] Toplis B, Pasupathy S. Tracking improvements in fast RLS algorithms using a variable forgetting factor. *IEEE Trans Acoust Speech Signal Process* 1988;ASSP-36(2):206–27.



**Jin Jiang** obtained his Ph.D. degree in 1989 from the Department of Electrical Engineering, University of New Brunswick, Fredericton, New Brunswick, Canada. Currently, he is a Professor in the Department of Electrical and Computer Engineering at The University of Western Ontario, London, Ontario, Canada. His research interests are in the areas of fault-tolerant control of safety-critical systems, power system dynamics and controls, and advanced signal processing.



**Youmin Zhang** received his Ph.D. degree in 1995 from the Department of Automatic Control, Northwestern Polytechnical University, Xian, PR China. He is currently an Assistant Professor in the Department of Computer Science and Engineering at Aalborg University Esbjerg, Esbjerg, Denmark. His main research interests are in the areas of fault diagnosis and fault-tolerant (control) systems, dynamic systems modelling, identification and control, and signal processing.