Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2006) 568–576
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Modified gradient algorithm for total least square filtering

Xiangyu Kong*, Chongzhao Han, Ruixuan Wei

School of Electronic and Information Engineering, Xi'an Jiaotong University, 710049 Xi'an, PR China

Received 28 April 2004; received in revised form 9 October 2005; accepted 31 October 2005
Available online 3 March 2006
Communicated by G.P. Zhang

Abstract

A neural approach for the parameter estimation of adaptive FIR filters for linear system identification is presented in the paper. It is based on a linear neuron with a modified gradient algorithm, capable of resolving the total least squares (TLS) problem present in this kind of estimation, where noisy errors affect not only the observation vector but also the data matrix. The learning rule is analyzed mathematically. The results of computer simulations are given to illustrate that the neural approach considerably outperforms the existing TLS methods when a larger learning factor is used or the signal-noise-ratio (SNR) is lower.
© 2006 Elsevier B.V. All rights reserved.

Keywords: FIR filters; Linear system identification; Total least square method

1. Introduction

The problem of *linear parameter estimation* gives rise to an overdetermined set of linear equations $\boldsymbol{Ax} \approx \boldsymbol{b}$ where $\boldsymbol{A}$ is the *data matrix* and $\boldsymbol{b}$ is the *observation vector*. In the least squares (LS) approach there is an underlying assumption that all errors are confined to the observation vector. This assumption is often unrealistic: the data matrix is not error-free because of sampling errors, human errors, modeling errors and instrument errors. The method of *total least squares* (TLS) is a technique devised to compensate for data errors. It was first introduced in [12], where it has been solved by using the *singular-value decomposition* (SVD), as pointed out in [11] and more fully in [10]. This method of fitting has a long history in the statistical literature, where the method is known as *orthogonal regression* or *errors-in-variables* (EIV) *regression*. These methods have been studied since 30 years ago [9]. A complete analysis of the TLS problems can be found in [22], where the algorithm of [12] is generalized to all cases in which it fails to produce a solution (*nongeneric* TLS).

The TLS problem can be solved by using direct and iterative methods. The direct methods compute directly the SVD. Since the multiplication number of the SVD for $N \times N$ matrix is $6N^3$, the application of TLS problems are limited in practice. Among the iterative methods, which are good for slowly varying set of equations, the most important use the inverse iteration, the ordinary and inverse Chebyshev iteration, the Rayleigh quotient iteration and the Lanczos methods (for a survey see [22]). The neural approaches can be considered iterative methods. There are two neural ways of solving TLS problem: (1) One is a linear neuron for the minor component analysis (MCA), which finds the eigenvector (MC) minimizing the Rayleigh quotient with a gradient learning, and a subsequent normalization ($\boldsymbol{W}_{L+1} = -1$) is needed. The most important MCA linear neurons are found in [4–6,17,19,23] and their performance have been studied in detail in [4]. (2) Another is linear neural network acting directly on the $\boldsymbol{W}_{L+1} = -1$ hyperplane (TLS NN). The existing TLS NNs are the Hopfield-like neural network of Luo [15,16], the principal limit of which is the fact that it is linked to the dimensions of the data matrix and cannot be used without structural changes for other TLS problems. Others are the linear neurons [1,2,7,8], which is correct enough for small gains and, above all, for weight norms much smaller than one. The TLS EXIN linear neuron [3] is a new neural approach, which is superior in performance and permits the use of such powerful algorithms as SCG and BFGS.

*Corresponding author.
E-mail address: xiangyukong01@163.com (X. Kong).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.10.005

Despite the large number of neural algorithms of solving the TLS problem proposed to date, there are few algorithms that posses the following desirable properties: (1) the neuron does not suffer from the sudden divergence; (2) the performance is good for larger learning factor or under higher noise environments. Considering the practical application as on-line system identification or fault diagnosis for higher noise environment, it is necessary to find an efficient neural approach with the above properties. The paper presents a TLS neural approach, which is applied to the parameter estimation of an adaptive FIR filters for system identification in presence of additive noise in both input and output signals.

The paper is organized as follows: In Section 2 the basic TLS formulation is introduced. Section 3 is devoted for the TLS linear neuron with modified gradient algorithm and the performance analysis. The simulation results are shown in Section 4. Finally, Section 5 is the conclusion.

2. The basic TLS formulation

The TLS method solves a set of m linear equations in $n \times 1$ unknowns $\boldsymbol{x}$ represented in matrix form by

$$\boldsymbol{Ax} \approx \boldsymbol{b}, \tag{1}$$

where $\boldsymbol{A}$ is the *data matrix* of elements a_{ij} and columns a_i, $\boldsymbol{b}$ is the *observation vector* of elements b_i and $m > n$ (*overdetermined system*).

The SVD of the $m \times (n+1)$ matrix $\boldsymbol{E} = [\boldsymbol{A};\boldsymbol{b}]$ is denoted by

$$[\boldsymbol{A};\boldsymbol{b}] = \boldsymbol{U} \sum \boldsymbol{V}^{\mathrm{T}} \tag{2}$$

where

$$\boldsymbol{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_m], \boldsymbol{u}_i \in \Re^m, \boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{I}_m, \tag{3}$$

$$\boldsymbol{V} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_{n+1}], \boldsymbol{v}_i \in \Re^{n+1}\, \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^{\mathrm{T}} = \boldsymbol{I}_{n+1}, \tag{4}$$

$$\sum = diag(\sigma_1, \dots, \sigma_{n+1}) \in \Re^{m \times (n+1)}, \sigma_1 \geqslant \dots \geqslant \sigma_{n+1} \geqslant 0. \tag{5}$$

The TLS problem is defined as *basic* if (1) it is unidimensional, (2) it is solvable and (3) it has a unique solution.

Given the overdetermined set (1), the TLS problem searches for

$$\min_{[\hat{\boldsymbol{A}};\hat{\boldsymbol{b}}] \in \Re^{m \times (n+1)}} \left\| [\boldsymbol{A};\boldsymbol{b}] - [\hat{\boldsymbol{A}};\hat{\boldsymbol{b}}] \right\|_F \quad \text{s.t.} \quad \hat{\boldsymbol{b}} \in \boldsymbol{R}(\hat{\boldsymbol{A}}), \tag{6}$$

where $\|\cdots\|_F$ is the Frobenius norm and $\boldsymbol{R}(\hat{\boldsymbol{A}})$ is the column space of $\hat{\boldsymbol{A}}$. Once a minimizing $[\hat{\boldsymbol{A}};\hat{\boldsymbol{b}}]$ is found, then any $\hat{\boldsymbol{x}}$ satisfying $\hat{\boldsymbol{A}}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ is called a TLS solution, and the TLS solution [3]

$$\mathbf{x}_{\mathrm{TLS}} = -\frac{1}{v_{n+1,n+1}}[v_{1,n+1}, \dots, v_{n,n+1}]^{\mathrm{T}} \tag{7}$$

exists and is the unique solution to $\hat{\boldsymbol{A}}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$, where $\boldsymbol{v}_{n+1}$ is the right singular vector corresponding to the smallest singular value of $\boldsymbol{E}$ and $v_{n+1,n+1}$ is the last component of v_{n+1}.

3. The TLS linear neuron with modified gradient algorithm

Consider the adaptive filter when the input and output observation data are both corrupted by noise, and write the input and output observation series at k as $\{[\tilde{x}(k), \tilde{d}(k)] | k = 1, 2, \dots, N\}$, $\tilde{x}(k) = x(k) + n_i(k)$, $\tilde{d}(k) = d(k) + n_o(k)$, where $n_i(k)$ and $n_o(k)$ are the additive noise. Write the filter weight vector as $\boldsymbol{H}(k) = [h_1, \dots, h_n]^{\mathrm{T}}$ and the input vector at k as $\tilde{\boldsymbol{X}}(k)$, then the filter output at k can be written as $y(k) = \tilde{\boldsymbol{X}}^{\mathrm{T}}(k)\boldsymbol{H}(k)$ and output error as $\varepsilon(k) = y(k) - \tilde{d}(k)$. Write the augmented input vector as $\boldsymbol{Z}(k) = [\tilde{\boldsymbol{X}}^{\mathrm{T}}(k),\ \tilde{d}(k)]^{\mathrm{T}}$ and the augmented weighted vector as $\boldsymbol{W}(k) = [\boldsymbol{H}^{\mathrm{T}}(k),\ -1]^{\mathrm{T}}$, then the output error can be written as $\varepsilon(k) = \boldsymbol{Z}^{\mathrm{T}}(k)\boldsymbol{W}(k)$.

3.1. *The derivation of the modified gradient algorithm*

Set the Rayleigh quotient of the augmented weighted vector $\boldsymbol{W}(k)$ as the TLS cost function

$$J(k) = r(\boldsymbol{W}(k), \boldsymbol{R}) = \frac{\boldsymbol{W}(k)^{\mathrm{T}}\boldsymbol{R}\boldsymbol{W}(k)}{\boldsymbol{W}(k)^{\mathrm{T}}\boldsymbol{W}(k)} = \frac{E[\varepsilon(k)^2]}{\|\boldsymbol{W}(k)\|_2^2}, \tag{8}$$

where $\boldsymbol{R} = E\{\boldsymbol{Z}(k)\boldsymbol{Z}(k)^{\mathrm{T}}\}$ is the autocorrelation matrix of the augmented input vector.

Evaluation of the gradient of (8) yields

$$\nabla J_{\boldsymbol{W}}(k) = \frac{\|\boldsymbol{W}(k)\|_2^2(2\boldsymbol{R}\boldsymbol{W}(k)) - 2\boldsymbol{W}(k)(\boldsymbol{W}(k)^{\mathrm{T}}\boldsymbol{R}\boldsymbol{W}(k))}{\left[\|\boldsymbol{W}(k)\|_2^2\right]^2}. \tag{9}$$

To develop adaptive algorithm, we evaluate the steepest descent of the cost function gradient to give

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) - \alpha(k)\frac{\|\boldsymbol{W}(k)\|_2^2\boldsymbol{R}\boldsymbol{W}(k) - (\boldsymbol{W}(k)^{\mathrm{T}}\boldsymbol{R}\boldsymbol{W}(k))\boldsymbol{W}(k)}{\|\boldsymbol{W}(k)\|_2^4}, \tag{10}$$

where $\alpha(k)$ is the learning rate, which controls the stability and rate of convergence of the algorithm.

The instantaneous approximation algorithm applied to (10) yields

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) - \alpha(k)\varepsilon(k)\frac{\boldsymbol{Z}(k)\|\boldsymbol{W}(k)\|_2^2 - \varepsilon(k)\boldsymbol{W}(k)}{\|\boldsymbol{W}(k)\|_2^4}. \tag{11}$$

Eq. (11) is the MCA EXIN algorithm for the TLS problem [4]. According to the property of the Rayleigh quotient [4], it can be known that the solution of TLS problem of formula (8) is the normalized eigenvector associated with the smallest eigenvalue of the input autocorrelation matrix $\boldsymbol{R}$.

We can use the *stochastic learning laws* [4,23] to analyze the algorithm (11). Algorithm (11) can be written as

$$\begin{aligned}
&\boldsymbol{W}(k+1)\\
&= \boldsymbol{W}(k) - \frac{\alpha(k)\varepsilon(k)}{\|\boldsymbol{W}(k)\|_2^2}\left(\boldsymbol{Z}(k) - \frac{\varepsilon(k)\boldsymbol{W}(k)}{\|\boldsymbol{W}(k)\|_2^2}\right)\\
&= \boldsymbol{W}(k) - \frac{\alpha(k)}{\|\boldsymbol{W}(k)\|_2^2}\left(\boldsymbol{Z}(k)\boldsymbol{Z}^{\mathrm{T}}(k)\boldsymbol{W}(k) - \frac{\varepsilon^2(k)\boldsymbol{W}(k)}{\|\boldsymbol{W}(k)\|_2^2}\right)\\
&= \boldsymbol{W}(k) - \frac{\alpha(k)}{\|\boldsymbol{W}(k)\|_2^2}\left(\left(\boldsymbol{Z}(k)\boldsymbol{Z}^{\mathrm{T}}(k) - \frac{\varepsilon^2(k)}{\|\boldsymbol{W}(k)\|_2^2}\boldsymbol{I}\right)\boldsymbol{W}(k)\right).
\end{aligned} \tag{12}$$

According to the orthogonal property of the *Raleigh quotient*, i.e., *the weight increment at each iteration is orthogonal to the weight direction* [4], and from the fact that the weight increment of aqua (12) is gradient flows of the *Raleigh quotient*, the formula below comes into existence

$$\boldsymbol{W}^{\mathrm{T}}(k)\left\{\boldsymbol{Z}(k)\varepsilon(k) - \frac{\varepsilon^2(k)\boldsymbol{W}(k)}{\|\boldsymbol{W}(k)\|_2^2}\right\} = 0. \tag{13}$$

Then the squared modulus of the weight vector at instant k+1 is given by

$$\begin{aligned}
&\|\boldsymbol{W}(k+1)\|_2^2\\
&= \|\boldsymbol{W}(k)\|_2^2 + \frac{\alpha^2(k)\varepsilon^2(k)}{\|\boldsymbol{W}(k)\|_2^4}\left\|\boldsymbol{Z}(k) - \frac{\varepsilon(k)\boldsymbol{W}(k)}{\|\boldsymbol{W}(k)\|_2^2}\right\|_2^2\\
&= \|\boldsymbol{W}(k)\|_2^2 + \frac{\alpha^2(k)\varepsilon^2(k)}{\|\boldsymbol{W}(k)\|_2^4}\left\|\|\boldsymbol{Z}(k)\|_2^2 - \frac{\varepsilon^2(k)}{\|\boldsymbol{W}(k)\|_2^2}\right\|\\
&= \|\boldsymbol{W}(k)\|_2^2 + \frac{\alpha^2(k)\varepsilon^2(k)}{\|\boldsymbol{W}(k)\|_2^4}\left\|\|\boldsymbol{Z}(k)\|_2^2 - \|\boldsymbol{Z}(k)\|_2^2\cos^2\theta_{zw}\right\|\\
&= \|\boldsymbol{W}(k)\|_2^2 + \frac{\alpha^2(k)\varepsilon^2(k)}{\|\boldsymbol{W}(k)\|_2^4}\|\boldsymbol{Z}(k)\|_2^2\sin^2\theta_{zw}\\
&= \|\boldsymbol{W}(k)\|_2^2 + \frac{\alpha^2(k)}{4}\|\boldsymbol{W}(k)\|_2^{-2}\|\boldsymbol{Z}(k)\|_2^4\sin^2 2\theta_{zw}
\end{aligned} \tag{14}$$

where θ_{zw} is the angle between the directions of the augmented observation vector $\boldsymbol{Z}(k)$ and the augmented weight vector $\boldsymbol{W}(k)$. From the formula (14), it can be known that $\|\boldsymbol{W}(k+1)\|_2^2 > \|\boldsymbol{W}(k)\|_2^2$, and because $sin^2 2\theta_{zw}$ is a positive function with four peaks in the interval $(-\pi,\pi]$, the squared modulus of the augmented weight vector always increases with the oscillatory behavior. Commonly in the TLS problem, the initial value of the squared modulus is bigger than or equal to one, then $\|\boldsymbol{W}(k)\|_2 > 1$ is true.

We know that $\boldsymbol{Z}(k)$ has the input and output noise, and from the above anaysis, $\|\boldsymbol{W}(k)\|_2^2$ (which is bigger than 1) always increases with the oscillatory behavior, so $\|\boldsymbol{W}(k)\|_2^2\boldsymbol{Z}(k)$ in Eq. (11) can enlarge these noise, which has a negative effect on the behavior of the algorithm. In order to improve the performance of the adaptive algorithm so as to be suitable for larger learning factor and higher noise environments, by substituting the term $\|\boldsymbol{W}(k)\|_2^2\boldsymbol{Z}(k)$ in formula (11) with $\boldsymbol{Z}(k)$, the modified gradient algorithm for TLS problem can be built as follows

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) - \alpha(k)\varepsilon(k)\frac{\boldsymbol{Z}(k) - \varepsilon(k)\boldsymbol{W}(k)}{\|\boldsymbol{W}(k)\|_2^4}. \tag{15}$$

This is our modified gradient algorithm for TLS problem and its properties are analyzed below. The presented TLS neuron is a linear unit with n inputs, n weights, one output and one training error, and the presented algorithm is a modified gradient algorithm, which can be used for the NN weights estimation where the input and output observation data are both corrupted by noise.

3.2. The analysis of the convergence performance

According to the stochastic approximation theory [13,14] and further explanations given by Oja and Karhunen [20], it can be show that if some conditions are satisfied [17], then the asymptotic limit of the above discrete learning algorithm can be solved by applying the corresponding continuous time differential equations

$$\frac{\mathrm{d}\boldsymbol{W}(t)}{\mathrm{d}t} = -\frac{\varepsilon(t)[\boldsymbol{Z}(t) - \varepsilon(t)\boldsymbol{W}(t)]}{\|\boldsymbol{W}(t)\|_2^4}. \tag{16}$$

Assume $\boldsymbol{Z}(t)$ is stationary, and $\boldsymbol{Z}(t)$ is not correlated with $\boldsymbol{W}(t)$, and taking the expected value with respect to two sides of Eq. (16), then Eq. (16) can be approximated by the following differential equation:

$$\frac{\mathrm{d}\boldsymbol{W}(t)}{\mathrm{d}t} = \frac{-\boldsymbol{R}\boldsymbol{W}(t) + \boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{R}\boldsymbol{W}(t)\boldsymbol{W}(t)}{\|\boldsymbol{W}(t)\|_2^4} \tag{17}$$

For the sake of convenience, the expection of the $\mathrm{d}\boldsymbol{W}(t)/\mathrm{d}t$ is still written as $\mathrm{d}\boldsymbol{W}(t)/\mathrm{d}t$ in Eq. (17), and the latter equations such as Eq. (19), etc. is treated similarly. The asymptotic property of (16) approximates that of (17), and the asymptotic property of (17) can be insured by the following theorem.

Theorem. *Let $\boldsymbol{R}$ be a semipositive definite matrix, λ_{L+1} and $\mathbf{c}_{L+1}$ be respectively its smallest eigenvalue and corresponding normalized eigenvector with the non zero last component. If the initial augmented vector $\boldsymbol{W}(0)$ satisfies $\boldsymbol{W}^{\mathrm{T}}(0)\mathbf{c}_{\mathrm{L+1}}\neq 0$ and λ_{L+1} is single, then $\lim_{t\to\infty}\boldsymbol{W}(t) = \lim_{t\to\infty}f_{L+1}(t)\boldsymbol{c}_{L+1}$, where $f_{L+1}(t)$ is a scalar function, i.e. $\boldsymbol{W}(t)$ tends to be in the direction of $\mathbf{c}_{L+1}$ asymptotically as $t\to\infty$.*

Proof. Denote $L+1$ eigenvalues of $\boldsymbol{R}$ by $\lambda_1, \lambda_2, \ldots, \lambda_{L+1}$, where λ_{L+1} is the smallest eigenvalue, and denote a set of corresponding normalized eigenvectors by $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{L+1}$. So $\boldsymbol{R}$ and $\boldsymbol{W}(t)$ can be written as

$$\boldsymbol{R} = \sum_{i=1}^{L+1}\lambda_i\boldsymbol{c}_i\boldsymbol{c}_i^{\mathrm{T}},\ \boldsymbol{W}(t) = \sum_{i=1}^{L+1}f_i(t)\boldsymbol{c}_i. \tag{18}$$

And then it can be obtained

$$\frac{\mathrm{d}\boldsymbol{W}(t)}{\mathrm{d}t} = \sum_{i=1}^{L+1}\frac{\mathrm{d}f_i(t)}{\mathrm{d}t}\boldsymbol{c}_i = \frac{-\sum_{i=1}^{L+1}(\lambda_i f_i(t)\boldsymbol{c}_i) + \boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{R}\boldsymbol{W}(t)\sum_{i=1}^{L+1}(f_i(t)\boldsymbol{c}_i)}{\|\boldsymbol{W}(t)\|_2^4} = \sum_{i=1}^{L+1}\left[(-\lambda_i + \boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{R}\boldsymbol{W}(t))f_i(t)\boldsymbol{c}_i\right]\|\boldsymbol{W}(t)\|_2^{-4}. \quad (19)$$

Then, we have

$$\frac{\mathrm{d}f_i(t)}{\mathrm{d}t} = \left[-\lambda_i + \boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{R}\boldsymbol{W}(t)\right]f_i(t)\|\boldsymbol{W}(t)\|_2^{-4} \quad \forall i = 1,2,\ldots,L+1. \quad (20)$$

From (18) we have $f_i(t) = \boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{c}_i i = 1,2,...,L+1$. Since $f_{L+1}(0) = \mathbf{W}^{\mathrm{T}}(0)\mathbf{c}_{L+1} \neq 0$, $f_{L+1}(t) \neq 0$ for all t and we may define

$$\gamma_i(t) = \frac{f_i(t)}{f_{L+1}(t)} \quad (i = 1,\ldots,L). \quad (21)$$

And then the following differential equation can be obtained:

$$\frac{\mathrm{d}\gamma_i(t)}{\mathrm{d}t} = \frac{f_{L+1}(t)\frac{\mathrm{d}f_i(t)}{\mathrm{d}t} - f_i(t)\frac{\mathrm{d}f_{L+1}(t)}{\mathrm{d}t}}{f_{L+1}^2(t)} \overset{(20)}{=\!=} \frac{[\lambda_{L+1} - \lambda_i]f_i(t)f_{L+1}(t)}{f_{L+1}^2(t)\|\mathbf{W}(t)\|_2^4} = \frac{[\lambda_{L+1} - \lambda_i]\gamma_i(t)}{\|\mathbf{W}(t)\|_2^4}, \quad (22)$$

whose solution on $[0,\infty]$ is

$$\gamma_i(t) = \exp((\lambda_{L+1} - \lambda_i)\int_0^t \|\boldsymbol{W}(\tau)\|_2^{-4}d\tau) \quad \forall i = 1,2,\ldots,L. \quad (23)$$

If $\lambda_i > \lambda_{L+1}$ (that is, the smallest eigenvalue is single but not multiple), then $\gamma_i(t)$ tends to zero as $t \to \infty$ $(\forall i = 1,2,\ldots,L)$, consequently $\lim_{t\to\infty} f_i(t) = 0$ $(\forall i = 1,2,\ldots,L)$.

So our conclusion can be obtained

$$\lim_{t\to\infty}\boldsymbol{W}(t) = \lim_{t\to\infty}\left[\sum_{i=1}^{L+1} f_i(t)\boldsymbol{c}_i\right] = \lim_{t\to\infty} f_{L+1}(t)\boldsymbol{c}_{L+1}. \quad (24)$$

This completes the proof of the theorem. □

3.3. Dynamic stability and learning rate

Theorem in the previous section only deals with the asymptotic solutions of the learning rules. It says nothing about finite-time behavior, especially, about the speed of learning, which is strongly affected by the specific choice of the gain $\alpha(k)$. The learning rate $\alpha(k)$ should be quite small, otherwise the learning will become unstable and diverge. This may bring some problems: (a) a small learning rate gives a low learning speed; (b) it is difficult to find a good learning rate to prevent learning divergence; (c) the degree of fluctuation and thus the solution's accuracy are both affected by the choice of the learning rate.

Then, the study of the stochastic discrete learning laws of the presented neurons w.r.t. $\alpha(k)$ becomes an analysis of their dynamics.

Define

$$r' = \frac{|\boldsymbol{W}^{\mathrm{T}}(k+1)\boldsymbol{Z}(k)|^2}{\|\boldsymbol{W}(k+1)\|_2^2}, \quad (25)$$

$$r = \frac{|\boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{Z}(k)|^2}{\|\boldsymbol{W}(k)\|_2^2},$$

$$\rho(\alpha) = \frac{r'}{r} \geqslant 0, \quad p = \|\boldsymbol{W}(k)\|_2^2, \quad u = y^2(k). \quad (26)$$

The two scalars r' and r represent, respectively, the squared perpendicular distance from the input $\boldsymbol{Z}(t)$ to the data fitting hyperplane whose normal is given by the weight and passes through the origin, after and before the weight increment. Recalling the definition of MC, it should hold $r' \leqslant r$. If this inequality is not valid, this means that the learning law increases the estimation error due to disturbances caused by noisy data. When this disturbance is too large, it will make $\boldsymbol{W}(t)$ deviate drastically from the normal learning, which may result in divergence or fluctuations (implying an increased learning time).

Proposition. *If*

$$\alpha > \alpha_b = \frac{2\|\boldsymbol{W}(k)\|_2^4}{\|\boldsymbol{Z}(k)\|_2^2(1 - 2\|\boldsymbol{W}(k)\|_2^2\cos^2\theta_{zw})} \quad \text{and} \quad |\cos\theta_{ZW}| < \frac{1}{\sqrt{2}\|\boldsymbol{W}(k)\|_2}.$$

It holds that $r' > r$, which implies divergence, where θ_{zw} is the angle between the directions of $\boldsymbol{Z}(t)$ and $\boldsymbol{W}(t)$.

Proof. By Eq. (15), we have

$$\boldsymbol{W}^{\mathrm{T}}(k+1)\boldsymbol{Z}(k) = \varepsilon(k) - \frac{\alpha(k)\varepsilon(k)}{\|\boldsymbol{W}(t)\|_2^4}\left(\|\boldsymbol{Z}(k)\|_2^2 - \varepsilon^2(k)\right) = \varepsilon(k)\left(1 - \frac{\alpha(k)\left(\|\boldsymbol{Z}(k)\|_2^2 - \varepsilon^2(k)\right)}{\|\boldsymbol{W}(k)\|_2^4}\right), \quad (27)$$

$$\|\boldsymbol{W}(k+1)\|_2^2 = \|\boldsymbol{W}(k)\|_2^2 - \frac{2\alpha(k)\varepsilon^2(k)}{\|\boldsymbol{W}(k)\|_2^4}(1 - \|\boldsymbol{W}(k)\|_2^2) + \frac{\alpha^2(k)\varepsilon^2(k)}{\|\boldsymbol{W}(k)\|_2^8}(\|\boldsymbol{Z}(k)\|_2^2 - 2\varepsilon^2(k) + \varepsilon^2(k)\|\boldsymbol{W}(k)\|_2^2). \quad (28)$$

Therefore

$$\rho(\alpha) = \frac{r'}{r} = \frac{(\boldsymbol{W}^T(k+1)\boldsymbol{Z}(k))^2}{\left\|\boldsymbol{W}(k+1)\right\|_2^2}\frac{\left\|\boldsymbol{W}(k)\right\|_2^2}{\varepsilon^2(k)}$$
$$= \frac{\left\|\boldsymbol{W}(k)\right\|_2^2(1 - \frac{2\alpha(k)}{\left\|\boldsymbol{W}(k)\right\|_2^4}(\left\|\boldsymbol{Z}(k)\right\|_2^2 - \varepsilon^2(k)) + \frac{\alpha^2(k)}{\left\|\boldsymbol{W}(k)\right\|_2^8}(\left\|\boldsymbol{Z}(k)\right\|_2^2 - \varepsilon^2(k))^2)}{\left\|\boldsymbol{W}(k)\right\|_2^2 - \frac{2\alpha(k)\varepsilon^2(k)}{\left\|\boldsymbol{W}(k)\right\|_2^4}(1 - \left\|\boldsymbol{W}(k)\right\|_2^2) + \frac{\alpha^2(k)\varepsilon^2(k)}{\left\|\boldsymbol{W}(k)\right\|_2^8}(\left\|\boldsymbol{Z}(k)\right\|_2^2 - 2\varepsilon^2(k) + \varepsilon^2(k)\left\|\boldsymbol{W}(k)\right\|_2^2)} \tag{29}$$

Noticing that $\varepsilon(k) = \boldsymbol{W}^{\mathrm{T}}(k)\boldsymbol{Z}(k) = \left\|\boldsymbol{W}(k)\right\|\left\|\boldsymbol{Z}(k)\right\|\cos\theta_{zw}$, we see that $\rho(\alpha) > 1$ is equivalent to

$$\frac{\alpha(k)}{\left\|\boldsymbol{W}(k)\right\|_2^2}\left(\frac{\alpha(k)\left\|\boldsymbol{Z}(k)\right\|_2^4}{\left\|\boldsymbol{W}(k)\right\|_2^4}\sin^2\theta_{zw}(1 - 2\left\|\boldsymbol{W}(k)\right\|_2^2\cos^2\theta_{zw}) - 2\left\|\boldsymbol{Z}(k)\right\|_2^2\sin^2\theta_{zw}\right) > 0. \tag{30}$$

Since $\alpha(k)$ takes values in the interval (0, 1), we see that this inequality holds only when

$$\alpha > \alpha_b = \frac{2\left\|\boldsymbol{W}(k)\right\|_2^4}{\left\|\boldsymbol{Z}(k)\right\|_2^2(1 - 2\left\|\boldsymbol{W}(k)\right\|_2^2\cos^2\theta_{zw})} \tag{31}$$

and

$$|\cos\theta_{\boldsymbol{ZW}}| < \frac{1}{\sqrt{2}\left\|\boldsymbol{W}(k)\right\|_2} \tag{32}$$

This completes the proof of the proposition. □

Considering the case $0 < \alpha_b \leqslant \gamma < 1$ (Condition (32) can imply the case $0 < \alpha_b$), it holds

$$\cos^2\theta_{ZW} \le \frac{1}{2p}(1 - \frac{2p}{\left\|\boldsymbol{Z}(k)\right\|^2\gamma}) = \Upsilon, \tag{33}$$

which is more restrictive than (31). Define $\sigma = \arccos\sqrt{\Upsilon}$, and from Eq. (33), it can be seen that $-\sigma < \theta_{ZW} < +\sigma$ or $(180^\circ - \sigma) < \theta_{ZW} < (180^\circ + \sigma)$ is the stability domain and an increasing weight modulus increases the stability. Also the decrease of γ has a similar positive effect.

3.4. The rule of the squared weight modulus

In TLS or MCA linear neurons, the norm of the weight of some neurons diverge in a finite time, i.e. $\left\|\mathbf{W}(t)\right\|_2^2 \to \infty$ when $t \to t_\infty$, where t_∞ is a finite time. The divergence is called as sudden divergence, which causing problems in the choice of a stopping criterion. Then, does the sudden divergence phenomenon happen in a finite time for the presented neuron approach? In this part, we will study this in detail.

3.4.1. The increasing rule of the squared weight modulus

By Eq. (15), we have

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + \alpha(k)\varepsilon(k)\frac{\varepsilon(k)\boldsymbol{W}(k) - \boldsymbol{Z}(k)}{\left\|\boldsymbol{W}(k)\right\|_2^4}. \tag{34}$$

The squared modulus of the weight vector at instant $k+1$ is then given by

$$\begin{aligned}\left\|\boldsymbol{W}(k+1)\right\|_2^2 &= \left\|\boldsymbol{W}(k)\right\|_2^2 + \frac{2\alpha(k)\varepsilon(k)\boldsymbol{W}^{\mathrm{T}}(k)}{\left\|\boldsymbol{W}(k)\right\|_2^4}(\varepsilon(k)\boldsymbol{W}(k) - \boldsymbol{Z}(k)) + O(\alpha^2(k))\\ &= \left\|\boldsymbol{W}(k)\right\|_2^2 + \frac{2\alpha(k)\varepsilon^2(k)}{\left\|\boldsymbol{W}(k)\right\|_2^4}(\left\|\boldsymbol{W}(k)\right\|_2^2 - 1) + O(\alpha^2(k)).\end{aligned} \tag{35}$$

From the above equation, we can see that the increment in the squared modulus is always decreasing or increasing according to the initial conditions. Considering that it always holds $\left\|\boldsymbol{W}(0)\right\|_2 \geqslant 1$, the increment in the squared modulus is always increasing.

From Eq. (15), it can be seen that our algorithm has the quantity $\left\|\boldsymbol{W}(k)\right\|_2^4$ in the denominator of the weight increment. It can be considered an iterative algorithm whose global learning rate is given by $\alpha(k)/\left\|\boldsymbol{W}(k)\right\|_2^4$. From this point of view, the global learning rate will become smaller and smaller with the increasing $\left\|\boldsymbol{W}(k)\right\|_2^4$, which can improve the transient and the accuracy in the solution. However, the quantity $\left\|\boldsymbol{W}(k)\right\|_2^4$ in the denominator also makes the global learning rate $\alpha(k)/\left\|\boldsymbol{W}(k)\right\|_2^4$ smaller in the initial period, which leads to the rate of the convergence lower than other TLS algorithms.

3.4.2. The sudden divergence of the squared weight modulus

From Eq. (17), it holds

$$\begin{aligned}\frac{\mathrm{d}\left\|\boldsymbol{W}(t)\right\|_2^2}{\mathrm{d}t} &= \frac{2(\boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{RW}(t)\boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{W}(t) - \boldsymbol{W}(t)^{\mathrm{T}}\boldsymbol{RW}(t))}{\left\|\boldsymbol{W}(t)\right\|_2^4}\\ &= \frac{2[\boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{RW}(t)]}{\left\|\boldsymbol{W}(t)\right\|_2^4}(\left\|\boldsymbol{W}(t)\right\|_2^2 - 1).\end{aligned} \tag{36}$$

Assuming that the MC direction has been approached and according to the Remark 6[4], the above equation can be approximated as

$$\frac{\mathrm{d}p}{\mathrm{d}t} = \frac{2\lambda_{L+1}}{p}(p-1). \tag{37}$$

where $p = \|\boldsymbol{W}(t)\|_2^2$, λ_{L+1} is the smallest eigenvalue of the autocorrelation matrix $\boldsymbol{R}$. Define the instant of time in which the MC direction is approached as to t_0 and the corresponding value of the squared weight modulus as p_0. The solution of Eq. (37) is given by

$$p = p_0 \quad \text{if} \quad p_0 = 1,$$
$$p + \ln(p-1) = 2\lambda_{L+1}(t - t_0) + p_0 + \ln(p_0 - 1) \quad \text{if} \quad p_0 \neq 1. \tag{38}$$

From Eq. (38), it can be seen that though the value of the squared weight modulus is always increasing according to the initial conditions, the sudden divergence does not happen in a finite time.

Noisy input data implies a high λ_{L+1}. From the Eq. (38), the value of the squared weight modulus will increase with the λ_{L+1}, then the global learning rate $\alpha(k)/\|\boldsymbol{W}(k)\|_2^4$ is smaller than other TLS algorithms, which is positive for the anti-noise behavior of the algorithm.

4. Simulation

In this section, we present some simulation results for demonstrating the performance of the proposed TLS algorithm. Three examples are given in this paper. All the learning curves below are obtained by averaging 30 independent experiments. Assume that the SNR of the input signal is equal to the rate of the output signal. The additive noise is independent zero-mean white noise. The input signal is zero-mean white Gaussian noise with variance 1.

Example 1. In the simulation, four algorithms are used for a linear system identification, i.e. the MCA algorithm [23] (MCA), the total least mean squares algorithm [6] (TLMS), the MCA EXIN algorithm [4] (MCA EXIN) and the presented neural algorithm (MTLS). The linear system is given as $\boldsymbol{H} = [-0.3,\ -0.9,\ 0.8,\ -0.7,\ 0.6]^{\mathrm{T}}$. Their convergence performances are compared under different extent noise environment and by using different learning factor.

Define the learning error of the weight vector as $Error(k) = 20 \log\left[\left\|\boldsymbol{H} - \hat{\boldsymbol{H}}(k)\right\|^2\right]$, where $\hat{\boldsymbol{H}}(k)$ is the estimated weight vector. Figs. 1–3 show, respectively, the learning curves for SNR = 20, 10 and 5 dB with different learning factors.

These learning curves indicate that present algorithm has the best accuracy and the best convergence performance when a larger learning factor is used or SNR is lower in linear system. However, from these learning curves (Figs. 1(a), 2(a) and 3(a)) it can be seen that the present algorithm has the lower convergence speed than other congener TLS algorithms when all the algorithms are convergent, which indicates the tracking performance of the presented algorithm is inferior.

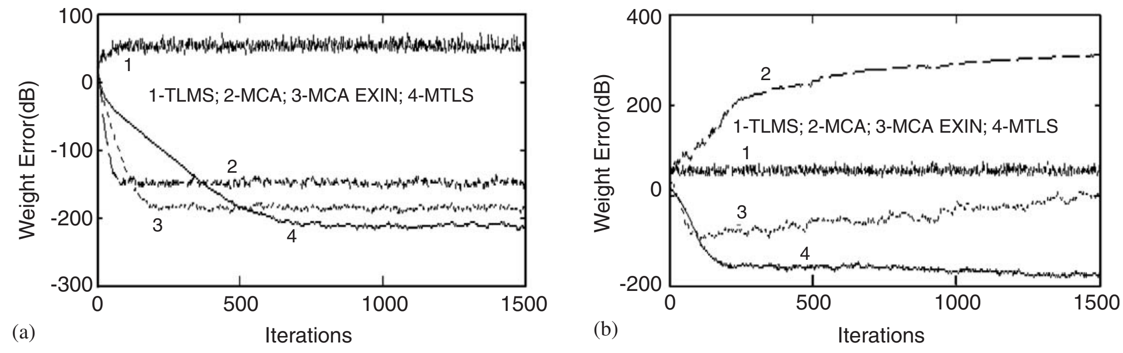


Fig. 1. Learning curve at: (a) $\mu = 0.1$, 20 dB (b) $\mu = 0.5$, 20 dB.

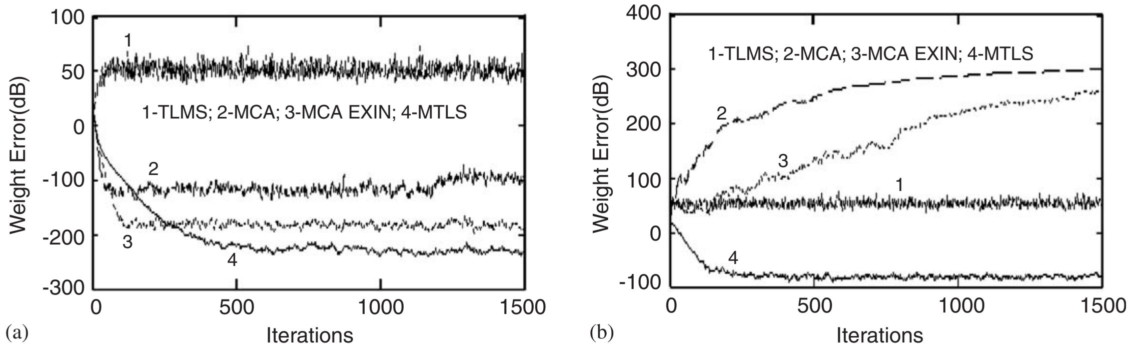
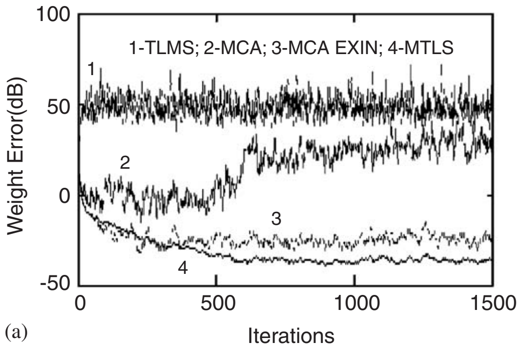


Fig. 2. Learning curve at (a) $\mu = 0.1$, 10 dB (b) $\mu = 0.5$, 10 dB.



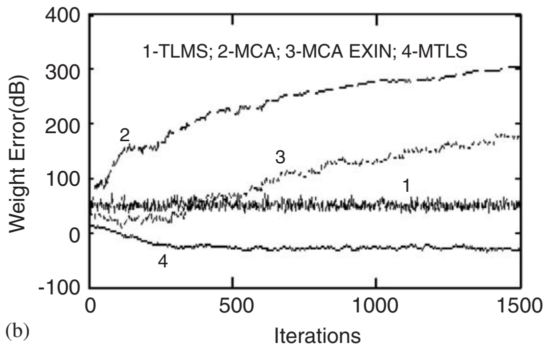
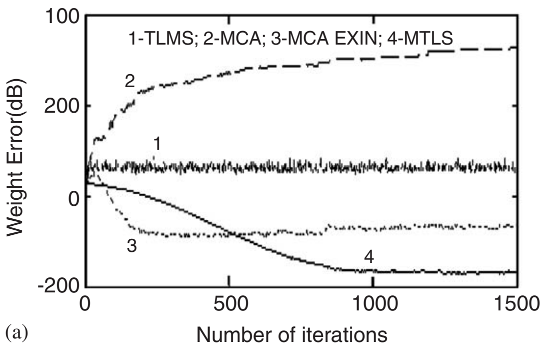


Fig. 3. Learning curve at: (a) $\mu = 0.1$, 5 dB; (b) $\mu = 0.3$, 5 dB.



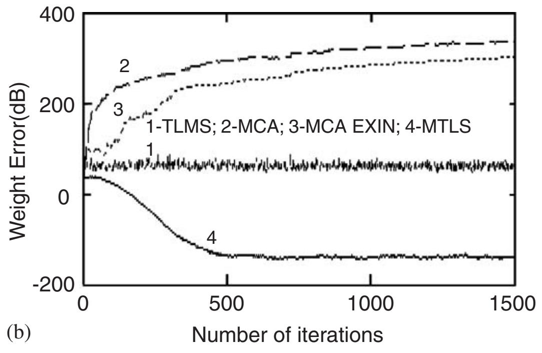


Fig. 4. Learning curve at: (a) $\mu = 0.3$, 20 dB; (b) $\mu = 0.8$, 20 dB.

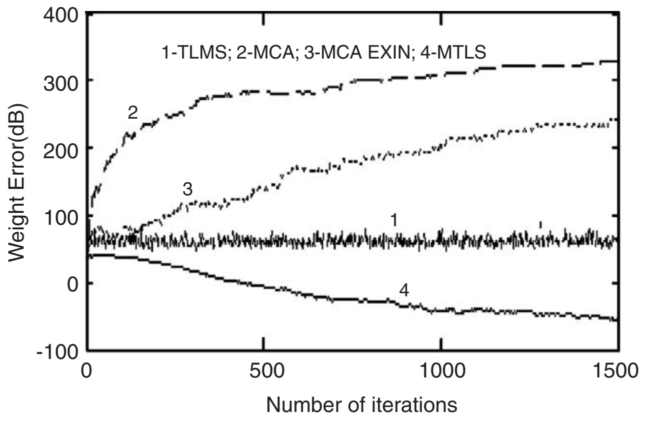


Fig. 5. Learning curve at $\mu = 0.4$, 10 dB.

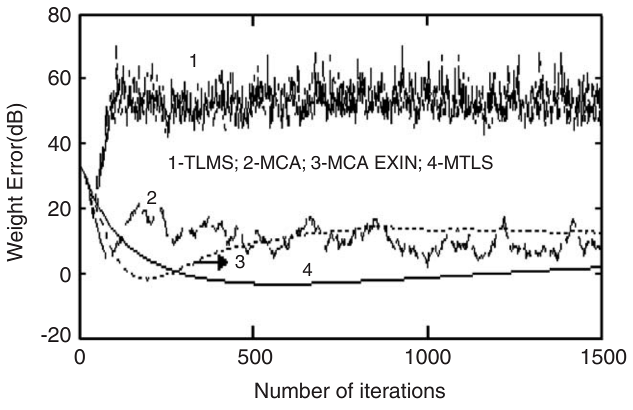


Fig. 6. Learning curve at $\mu = 0.005$, 5 dB.

Example 2. In the simulation, four algorithms in Example 1 are applied to adaptive FIR filters with larger number of coefficients. Let the parameter values of the "unknown" FIR system be given as $\boldsymbol{H} = [-0.3, -0.9, 0.8, -0.7, 0.6, 0.2, -0.5, 1.0, -0.7, 0.9, -0.4]^{\mathrm{T}}$. Figs. 4–6 show, respectively, the learning curves of the fours algorithms for SNR = 20, 10 and 5 dB with different learning factors.

These learning curves indicate that, for FIR filters with higher numbers of coefficients, the presented algorithm similarly has the best accuracy and the best convergence performance when a larger learning factor is used or SNR is lower in linear system. Fig. 6 indicates that, compared with the other congener TLS algorithms, the presented algorithm has the best stability for the environments with higher noise.

Example 3. By describing the Volterra filter as a pseudo-linear one, the presented algorithm can be used for the identification of nonlinear system based on the Volterra series. The algorithm can be given as

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) - \alpha(k)\varepsilon(k)\frac{\boldsymbol{Z}(k) - \varepsilon(k)\boldsymbol{W}(k)}{\|\boldsymbol{W}(k)\|_2^4}, \tag{39}$$

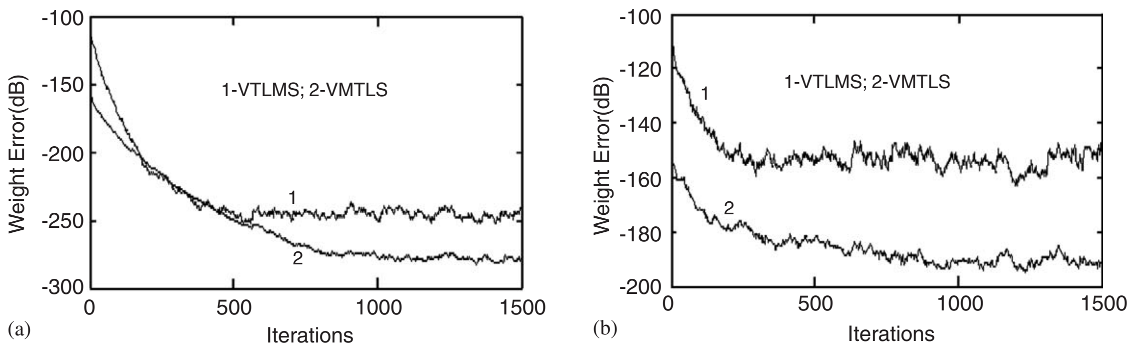


Fig. 7. Learning curve at: (a) $\mu = 0.05$, 30 dB (b) $\mu = 0.05$, 20 dB.

where $\boldsymbol{W} = [\boldsymbol{H}_V^{\mathrm{T}},\ -1]^{\mathrm{T}}$ is the Volterra augmented kernel vector and $\boldsymbol{H}_V$ is the Volterra kernel vector. $\boldsymbol{Z}(k)$ is called as the Volterra augmented observation vector, and $\varepsilon(k) = \boldsymbol{Z}^{\mathrm{T}}(k)\boldsymbol{W}(k)$ is Volterra output error. A detailed treatment for Volterra system may be found in [18,21].

In the simulation, four algorithms in Example 1 are used for a Volterra nonlinear system. The nonlinear system is given by $y(k) = 0.62u(k)-0.25u(k-1)+0.64u(k-2)+0.8^2 u^2(k)-1.48u(k-1)u(k-2)$. Their convergence performances are compared under different extent noise environment and by using different learning factor. Because the TLMS and MCA algorithms do not converge in the simulation, their learning curves are not drawn in Fig. 7.

These learning curves indicate that the MCA EXIN algorithm and the presented algorithm can be applied to Volterra filters, and that other algorithms cannot. Furthermore, from Fig. 7 it can be seen that the presented algorithm has the better convergence performance and accuracy than the MCA EXIN algorithm.

5. Conclusion

A TLS neuron with a modified gradient algorithm is presented for the parameter estimation of adaptive FIR filters for system identification where noisy errors affect not only the observation vector but also the data matrix. With respect to other TLS algorithms, it is superior in noise-robustness performance and accuracy when a larger learning factor is used or SNR is lower. All these properties are proved mathematically and simulated numerically.

Acknowledgements

The Paper is supported by national key fundamental research and development program (973) of PR China, No.2001CB309403, and National Natural Science Foundation of China, No. 60304004.

Reference

[1] A. Cichocki, R. Unbehauen, Neural Networks for Optimization and Signal Processing, Wiley, New York, 1993.
[2] A. Cichocki, R. Unbehauen, Simplified neural networks for solving linear least squares and total least squares problems in real time, IEEE Trans. Neural Networks 5 (6) (1994) 910–923.
[3] G. Cirrincione, M. Cirrincione, Linear system identification using the TLS EXIN neuron, Neurocomputing 28 (1) (1999) 53–74.
[4] G. Cirrincione, M. Cirrincione, The MCA EXIN neuron for the minor component analysis, IEEE Trans. Neural Networks 13 (1) (2002) 160–187.
[5] B.E. Dunne, G.A. Williamson, Stable simplified gradient algorithms for total least squares filtering, in: Proceedings of 32nd Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, 2000, pp. 1762–1766.
[6] D.Z. Feng, Z. Bao, L.C. Jiao, Total least mean squares algorithm, IEEE Trans. Signal Process. 46 (8) (1998) 2122–2130.
[7] K.Q. Gao, M.O. Ahmad, M.N. Swamy, Learning algorithm for total least-squares adaptive signal processing, Electron. Lett. 28 (4) (1992) 430–432.
[8] K.Q. Gao, M.O. Ahmad, M.N. Swamy, A constrained anti-Hebbian learning algorithm for total least-squares estimation with applications to adaptive FIR and IIR filtering, IEEE Trans. Circuits Syst. II: Analog Digital Signal Process. 41 (11) (1994) 718–729.
[9] L.J. Glser, Estimation in a multivariate “errors in variables” regression model: large sample results, Ann. Stat. 9 (1981) 24–44.
[10] G.H. Golub, Some modified eigenvalue problems, SIAM Rev. 15 (1973) 318–344.
[11] G.H. Golub, C. Reinsch, Singular value decomposition and least squares solutions, Numer. Math. 14 (1970) 403–420.
[12] G.H. Golub, C.F. Van Loan, An analysis of the total least squares problem, SIAM J. Numer. Anal. 17 (6) (1980) 883–893.
[13] H. Kushner, D. Clark, Stochastic Approximation Methods for Constrained and Unconstrained Systems, Springer, New York, 1978.
[14] L. Ljung, Analysis of recursive stochastic algorithms, IEEE Trans. Automat. Control 22 (1977) 551–575.
[15] F. Luo, Y. Li, C. He, Neural network approach to the TLS linear prediction frequency estimation problem, Neurocomputing 11 (1996) 31–42.
[16] F. Luo, R. Unbehauen, Applied Neural Networks for Signal Processing, Cambridge University Press, Cambridge, 1997.
[17] F. Luo, R. Unbehauen, A. Cichocki, A minor component analysis algorithm, Neural Networks 10 (2) (1997) 291–297.
[18] V.J. Mathews, Adaptive polynomial filters[J], IEEE Signal Process. Mag. 8 (3) (1991) 10–26.
[19] E. Oja, Principal components, minor components and linear neural networks, Neural Networks 5 (1992) 927–935.
[20] E. Oja, J. Karhunen, On stochastic approximation of the eigenvectors and eigenvalues of the expection of a random matrix, J. Math. Anal. Appl. 106 (1985) 69–84.
[21] M. Schetzen, The Volterra and Wiener Theories of Nonlinear Systems, Malabar, Krieger, FL, 1989.

[22] S. Van Huffel, J. Vandewalle, The total least squares problems: computational aspects and analysis, Frontiers in Applied Mathematics, SIAM, Philadelphia, 1991.

[23] L. Xu, E. Oja, C. Suen, Modified Hebbian learning for curve and surface fitting, Neural Networks 5 (3) (1992) 441–457.

Xiangyu Kong born on May 18, 1968 in Shanxi province, PR China. He received his Bachelor degree from Beijing institute of technology, Beijing, PR China in 1990, and Master degree from the Xi'an Research institute of high technology Xi'an, PR China in 2000. In 2004 he was received Ph.D. degree in control science and engineering from Xi'an jiaotong University Xi'an, PR China. His current research interests are adaptive signal processing, system identification, nonlinear system modeling and its application.

Chongzhao Han born on February 13, 1943 in Shaanxi Province, PR China. He received his B.E. degree from Department of Electrical Engineering, Xi'an Jiaotong University (XJTU), Xi'an, PR China in 1968, and Master degree from Graduate School, Academic Sinica, Beijing, PR China in 1981.

During 1968–1978, he served as an engineer in Northwest Optical Firm in Xi'an. And since 1981, he has been a member of faculty of XJTU, and he was a full professor of XJTU in 1990, and was the Vice Director of Department of Information and Control Engineering of XJTU during 1991–1994, and was the Vice Dean of the School of Electronic and Information Engineering of XJTU during 1996–2001. Now, he is the Director of the Department of Automation, XJTU.

His research interests are in stochastic system analysis, estimation theory, nonlinear analysis and information fusion. He published seven books and more than 140 journal papers.

Ruixuan Wei born in Qishan county, Shannxi province, China, in 1968. He received the Bachelor and the Master of Technology degree from Air Force Engeering College, in 1990 and 1993, and the Ph.D. degree in control science and engineering from Xi'an jiaotong University, Xi'an, PR China, in 2001. Now he is a professor from Air Force Engineering University. His current interests cover adaptive signal processing, nonlinear Volterra system modeling and its application.