# Performance Analysis of Selective Partial Update Normalized Least Mean Squares Algorithm over an Adaptive Incremental Network

M. Shams Esfand Abadi, A. R. Danaee, and M. Seifouri

*Abstract*—**Selective partial update (SPU) strategy in adaptive filter algorithms is used to reduce the computational complexity. In this paper we apply the SPU normalized least mean squares algorithm (SPU-NLMS) for distributed estimation problem in an incremental network. The distributed SPU-NLMS (dSPU-NLMS) has close convergence speed to dNLMS, low steady-state mean square error (MSE), and low computational complexity features. In addition, the mean-square performance analysis of dSPU-NLMS algorithm for each individual node is presented. The theoretical expressions for stability bounds, transient and steady-state performance analysis of dSPU-NLMS are introduced. The validity of the theoretical results and the good performance of dSPU-NLMS are demonstrated by several computer simulations.**

*Index Terms*—**Selective partial update, normalized least mean squares, distributed estimation, incremental network.**

## NOMENCLATURE

$\|.\|^2$ : Squared Euclidean norm of a vector.
$\|t\|_\Sigma^2$ : $\Sigma$-Weighted Euclidean norm of a column vector $t$ defined as $t^T \Sigma t$.
$vec(\mathbf{T})$ : Creates an $M^2 \times 1$ column vector $t$ through stacking the columns of the $M \times M$ matrix $\mathbf{T}$.
$vec(t)$ : Creates an $M \times M$ matrix $\mathbf{T}$ from the $M^2 \times 1$ column vector $t$.
$\mathbf{A} \otimes \mathbf{B}$ : Kronecker product of matrices $\mathbf{A}$ and $\mathbf{B}$.
$Tr(.)$ : Trace of a matrix.
$(.)^T$ : Transpose of a vector or a matrix.
$E\{.\}$ : Expectation operator.

## I. INTRODUCTION

ONE of the important features of sensor networks is the cooperative attempt of sensor nodes. Sensor nodes use their processing of the important features of sensor networks is the cooperative attempt of sensor nodes. Sensor nodes use their processing abilities to locally perform simple computations and transfer only the required and partially processed data. Some applications of sensor networks include environment monitoring, target localization, military surveillance, transportation, as well as medical research [1]-[6].

In contrast to classical centralized methods, distributed processing uses local computations at each node and communications among neighboring nodes to solve problems over the entire network, which is a practical use among a wide range of applications. Because of data statistical variations and network topology changes, the processing should be adaptive and requires two time scales in distributed networks. During the initial period of time, each node makes a special estimation and then through consensus iterations, the nodes incorporate further estimations to attain the desired estimate. In incremental learning algorithms over a distributed network, each node cooperates only with one adjacent node to utilize the spatial dimension, whilst doing local computations in the time dimension [7]. Within the last decade, different incremental adaptive strategies such as distributed least mean squares (dLMS), distributed normalized LMS (dNLMS), and distributed affine projection algorithm (dAPA) were introduced over distributed networks [7], [8]. Unfortunately, by increasing the filter length, the computational complexity of classical distributed adaptive algorithms will increase. Selective partial update (SPU) strategy was introduced in single adaptive filter to reduce the computational load. In SPU adaptive filter algorithms, the filter coefficients are partially updated at each iteration rather than the entire filter coefficients in ordinary adaptive filters. The SPU normalized least mean squares (SPU-NLMS) algorithm is one of the important family of SPU adaptive filter algorithms [9]-[12].

Therefore, to reduce the computational complexity of distributed adaptive algorithms, the SPU-NLMS algorithm is applied in an incremental adaptive network to establish the distributed SPU-NLMS (dSPU-NLMS) algorithm. In the following, the mean-square performance analysis of the spatial and temporal SPU-NLMS adaptive algorithm is presented and closed-form expressions for the transient and steady-state performances of each node are established. Moreover, the theoretical expressions for mean and mean-square stability of dSPU-NLMS are introduced. The analysis is based on spatial-temporal energy conservation relation which incorporates the space-time structure of the data.

What we propose in this paper can be summarized as follows:

- The establishment of dSPU-NLMS. In comparison with dNLMS algorithm, the convergence speed of dSPU-NLMS is close to dNLMS algorithm.

Manuscript received July 30, 2012; revised May 25, 2013.

The authors are with the Faculty of Electrical and Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran, (e-mails: mshams@srttu.edu, ar.danaee@srttu.edu, mahmood.seifouri@srttu.edu).

Publisher Item Identifier S 1682-0053(12)2033

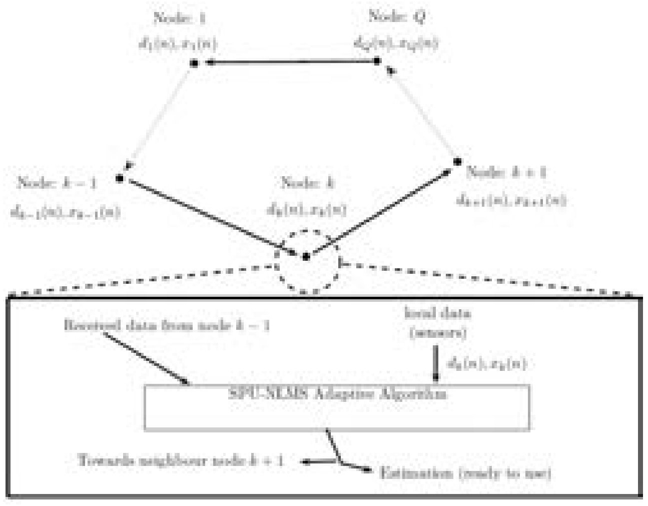

Fig. 1. Data processing in an incremental network through SPU-NLMS adaptive algorithm.

Furthermore, the computational complexity of dSPU-NLMS is lower than dNLMS due to SPU feature.

- Mean-square performance analysis of dSPU-NLMS. The theoretical expressions for transient and steady-state performance of the network are derived. The presented expressions can also be used to predict the performance of dLMS and dNLMS algorithms.
- Analysis of the mean and mean-square stability bounds for dSPU-NLMS algorithm.
- Performance analysis of the presented algorithm in an incremental network. The performance of dNLMS and dSPU-NLMS are compared and analyzed. The validity of the theoretical relations are also justified in both transient and steady-state for an incremental network.

This paper is organized as follows. In Section II, dSPU-NLMS algorithm is introduced. The computational complexity of dSPU-NLMS is discussed in Section III. In the next section, the mean-square performance analysis of each node is explained. Section V presents the stability bounds of step-size for dSPU-NLMS. Finally, the paper ends with a comprehensive set of simulations supporting the validity of the results.

## II. DSPU-NLMS Algorithm

The filter vector update equation for dNLMS algorithm in an incremental network over $Q$ nodes is given by

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \sum_{k=1}^{Q} \mu_k \frac{x_k(n)}{\|x_k(n)\|^2} e_k(n) \qquad (1)$$

where

$$e_k(n) = d_k(n) - x_k^T(n) h_{k-1}(n) \qquad (2)$$

is the error signal and $\mu_k$ is the step-size of each node. Also, $x_k(n) = [x_k(n), x_k(n-1), \ldots, x_k(n-(M+1))]^T$ is the $M \times 1$ data signal vector and $d_k(n) = x_k^T(n) h_t + v_k(n)$ where $v_k(n)$ is a temporally and spatially independent noise sequence with variance $\sigma_{v,k}^2$ at each node and $h_t$ is the true unknown system vector. Fig. 1 shows the learning algorithm in an incremental network for dSPU-NLMS adaptive algorithm. Based on the local data $(d_k(n), x_k(n))$, and $h_{k-1}(n)$ from its previous node $(k-1)$ in the cycle, the update for local estimation is performed at each time instant $n$. Finally, the local estimation $h_Q(n)$ is used as the global estimation for filter coefficients. This value is used as the initial local estimation $h_1(n+1)$ for the next time instant $n+1$. The local filter vector update equation for dNLMS adaptive algorithm in an incremental network is obtained by

$$h_k(n) = h_k(n+1) + \mu_k \frac{x_k(n)}{\|x_k(n)\|^2} e_k(n) \qquad (3)$$

In (3), all of the filter coefficients are updated at every iteration. In the following we introduce the dSPU-NLMS adaptive algorithm. Now partition the input signal vector and the vector of filter coefficients at each node into $B$ blocks each of length $L$ which are defined as

$$x_k(n) = [x_{1,k}^T(n), x_{2,k}^T(n), \ldots, x_{B,k}^T(n)]^T \qquad (4)$$

$$h_k(n) = [h_{1,k}^T(n), h_{2,k}^T(n), \ldots, h_{B,k}^T(n)]^T \qquad (5)$$

Suppose that we wish to update $S$ blocks of filter coefficients out of $B$ at every iteration. Let $\tau_S = \{i_1, i_2, \ldots, i_S\}$ denote $S$-subsets (subset with $S$ members) of the set $\{1, 2, \ldots, B\}$ and let $\Psi$ be the collection of all $S$-subsets, i.e. $\tau_S \in \Psi$. Selection of blocks can be carried out by considering the following constrained minimization problem

$$\min_{\tau_S \in \Psi} \min_{h_{\tau_S,k}(n)} \left\| h_{\tau_S,k}(n) - h_{\tau_S,k-1}(n) \right\|^2 \qquad (6)$$

subject to

$$h_k^T(n) x_k(n) = d_k(n) \qquad (7)$$

where the $SL \times 1$ vector $h_{\tau_S,k}(n)$ is defined by

$$h_{\tau_S,k}(n) = [h_{i_1,k}^T(n), h_{i_2,k}^T(n), \ldots, h_{i_S,k}^T(n)]^T \qquad (8)$$

This optimization problem can be solved by Lagrange multiplier approach. By minimizing the following cost function

$$\begin{aligned} J_{\tau_S,k}(n) &= \frac{1}{2} \left\| h_{\tau_S,k}(n) - h_{\tau_S,k-1}(n) \right\|^2 \\ &+ \lambda (d_k(n) - h_k^T(n) x_k(n)) \end{aligned} \qquad (9)$$

where $\lambda$ is a Lagrange multiplier, the update equation for dSPU-NLMS is introduced as

$$h_{\tau_S,k}(n) = h_{\tau_S,k-1}(n) + \frac{x_{\tau_S,k}(n)}{\left\| x_{\tau_S,k}(n) \right\|^2} e_k(n) \qquad (10)$$

where $x_{\tau_S,k}(n)$ is defined by

$$x_{\tau_S,k}(n) = [x_{i_1,k}^T(n), x_{i_2,k}^T(n), \ldots, x_{i_S,k}^T(n)]^T \qquad (11)$$

to determine which blocks to update, we need to find $S$ coefficient blocks with the minimum squared-Euclidean-norm update. The block selection problem can be formally written as

$$\begin{aligned} \tau_S &= \arg\min_{\tau_S \in \Psi} \left\| h_{\tau_S,k}(n) - h_{\tau_S,k-1}(n) \right\|^2 \\ &= \arg\min_{\tau_S \in \Psi} \left\| \frac{x_{\tau_S,k}(n) e_k(n)}{\left\| x_{\tau_S,k}(n) \right\|^2} \right\|^2 \\ &= \arg\max_{\tau_S \in \Psi} \left\| x_{\tau_S,k}(n) \right\|^2 = \arg\max_{\tau_S \in \Psi} \sum_{i \in \tau_S} \left\| x_{i,k}(n) \right\|^2 \end{aligned} \qquad (12)$$

The optimum $\tau_S$ satisfying (12) is found simply by ranking the regressor vector blocks according to their squared-Euclidean-norms and choosing the $S$ largest blocks

$$\left\|x_{i_1,k}(n)\right\|^2 \geq \left\|x_{i_2,k}(n)\right\|^2 \geq \ldots \geq \left\|x_{i_S,k}(n)\right\|^2 \geq \left\|x_{i,k}(n)\right\|^2 \quad , \quad \forall i \in \{1,2,\ldots,B\} \tag{13}$$

finally, the local update equation for dSPU-NLMS is given by

$$h_{\tau_S,k}(n) = h_{\tau_S,k-1}(n) + \mu_k \frac{x_{\tau_S,k}(n)}{\left\|x_{\tau_S,k}(n)\right\|^2} e_k(n) \tag{14}$$

where the indexes of $\tau_S$ are obtained by

$$\tau_S = \{i : \left\|x_{i,k}(n)\right\|^2 \text{ is one of the } S \text{ largest among } \left\|x_{1,k}(n)\right\|^2, \left\|x_{2,k}(n)\right\|^2, \ldots, \left\|x_{B,k}(n)\right\|^2\} \tag{15}$$

equation (14) can be represented in the form of full update equation as

$$h_k(n) = h_{k-1}(n) + \mu_k \frac{\mathbf{A}_k(n)x_k(n)}{\left\|\mathbf{A}_k(n)x_k(n)\right\|^2} e_k(n) \tag{16}$$

where the matrix $\mathbf{A}_k(n)$ is the $M \times M$ diagonal matrix with the **1** and **0** blocks each of length $L$ on the diagonal and the positions of **1**'s on the diagonal determine which coefficients should be updated in each iteration. The positions of **1** blocks on the diagonal of $\mathbf{A}_k(n)$ matrix for each iteration in dSPU-NLMS are determined by the indexes of $\tau_S$. Fig. 2 summarizes the pseudo-code implementation of dSPU-NLMS adaptive algorithm.

## III. Computational Complexity

Table I shows the number of multiplications, divisions, and comparisons of dNLMS and dSPU-NLMS per iteration per node. In dNLMS, the number of multiplications and division are $2M+2$ and 1 respectively. For every input sample, the computational complexity of dSPU-NLMS is $M+SL+2$ multiplications, one division, and $O(B)+B\log_2(S)$ comparisons when using the heapsort algorithm [10], [13]. As we can see, the number of multiplications in dSPU-NLMS is lower than dNLMS algorithm.

## IV. Performance Analysis

The performance analysis approach of dSPU-NLMS is based on the spatial-temporal energy conservation relation in [8] which was originally derived for single adaptive filter in [14]. In the mean-square performance analysis of dSPU-NLMS, we need to study the space-time evolution of the $E\{\left\|\tilde{h}_k(n)\right\|^2_{\Sigma_k}\}$ at each node, where $\Sigma_k$ is any Hermitian and positive-definite matrix[1], and $\tilde{h}_k(n)$ is the weight-error vector which is defined as $\tilde{h}_k(n) = h_t - h_k(n)$. The weight-error vector update equation for dSPU-NLMS adaptive algorithm can be obtained by

$$\tilde{h}_k(n) = \tilde{h}_{k-1}(n) - \mu_k \mathbf{A}_k(n)x_k(n)\mathbf{W}_k(n)e_k(n) \tag{17}$$

1. When $\Sigma_k = I$, the Mean Square Deviation (MSD) and when $\Sigma_k = \mathbf{R}_k$, where $\mathbf{R}_k = E\{x_k(n)x_k^T(n)\}$ is the autocorrelation matrix of the input signal, the Excess Mean Square Error (EMSE) expressions are established.

```
For each time instant n repeat:
    $k = 1,2,\ldots,Q$
    $h_0(n) = w(n-1)$
    $h_k(n) = h_{k-1}(n) + \mu_k \frac{\mathbf{A}_k(n)x_k(n)}{\|\mathbf{A}_k(n)x_k(n)\|^2} e_k(n)$
End
$w(n) = h_Q(n)$
```

Fig. 2: Pseudo-code implementation of dSPU-NLMS.

TABLE I
COMPUTATIONAL COMPLEXITY OF DNLMS AND DSPU-NLMS ADAPTIVE ALGORITHM PER ITERATION PER NODE

| Algorithm | DNLMS | DSPU-NLMS |
|---|---|---|
| Multiplications | $2M+2$ | $M+SL+2$ |
| Divisions | 1 | 1 |
| Comparisons | – | $O(B)+B\log_2 S$ |

where

$$\mathbf{W}_k(n) = 1 / \left\|\mathbf{A}_k(n)x_k(n)\right\|^2 \tag{18}$$

the error signal $e_k(n)$ can be represented as

$$e_k(n) = x_k^T(n)\tilde{h}_{k-1}(n) + v_k(n) \tag{19}$$

by substituting (19) in (17), and taking $\Sigma_k$-weighted norm from both sides of (17), we obtain

$$\left\|\tilde{h}_k(n)\right\|^2_{\Sigma_k} = \left\|\tilde{h}_{k-1}(n)\right\|^2_{\Sigma'_k} + \mu_k^2 v_k^2(n)\mathbf{Y}_k(n) + \{\text{Cross terms involving one instance of } v_k(n)\} \tag{20}$$

where

$$\mathbf{\Sigma}'_k = \mathbf{\Sigma}_k - \mu_k \mathbf{\Sigma}_k \mathbf{Z}_k(n) - \mu_k \mathbf{Z}_k^T(n)\mathbf{\Sigma}_k + \mu_k^2 \mathbf{Z}_k^T(n)\mathbf{\Sigma}_k \mathbf{Z}_k(n) \tag{21}$$

with $\mathbf{Y}_k(n) = \mathbf{W}_k^T(n)x_k^T(n)\mathbf{A}_k^T(n)\mathbf{\Sigma}_k\mathbf{A}_k(n)x_k(n)\mathbf{W}_k(n)$ and $\mathbf{Z}_k(n) = \mathbf{A}_k(n)x_k(n)\mathbf{W}_k(n)x_k^T(n)$. By taking the expectation from both sides of (20)

$$E\{\left\|\tilde{h}_k(n)\right\|^2_{\Sigma_k}\} = E\{\left\|\tilde{h}_{k-1}(n)\right\|^2_{\Sigma'_k}\} + \mu_k^2 E\{v_k^2(n)\mathbf{Y}_k(n)\} \tag{22}$$

we obtain the space-time evolution of the weight-error variance. To simplify (22), we need to use the following independence assumptions [8], [14]:

1) The matrix sequence $x_k(n)$ is independent of $x_k(n-1)$. This assumption guarantees that $\tilde{h}_{k-1}(n)$ is independent of both $\Sigma'_k$ and $x_k(n)$.
2) $\tilde{h}_{k-1}(n)$ is independent of $\mathbf{Z}_k(n)$.

Using these independence assumptions, the matrix $\Sigma'_k$ can be replaced by

$$\mathbf{\Sigma}'_k = \mathbf{\Sigma}_k - \mu_k \mathbf{\Sigma}_k E\{\mathbf{Z}_k(n)\} - \mu_k E\{\mathbf{Z}_k^T(n)\}\mathbf{\Sigma}_k + \mu_k^2 E\{\mathbf{Z}_k^T(n)\mathbf{\Sigma}_k\mathbf{Z}_k(n)\} \tag{23}$$

since $E\{v_k^2(n)\} = \sigma_{v,k}^2 \mathbf{I}$, (22) can be stated as

$$E\{\left\|\tilde{h}_k(n)\right\|^2_{\Sigma_k}\} = E\{\left\|\tilde{h}_{k-1}(n)\right\|^2_{\Sigma'_k}\} + \mu_k^2 \sigma_{v,k}^2 Tr(E\{\mathbf{Y}_k(n)\}) \tag{24}$$

by applying the $vec(.)$ operator on both sides of (23), and using $vec(\mathbf{P\Sigma Q}) = (\mathbf{Q}^T \otimes \mathbf{P})vec(\mathbf{\Sigma})$, we obtain

$$\sigma'_k = \sigma_k - \mu_k(E\{\mathbf{Z}_k^T(n)\} \otimes \mathbf{I}).\sigma_k - \mu_k(\mathbf{I} \otimes E\{\mathbf{Z}_k^T(n)\}).\sigma_k + \mu_k^2(E\{\mathbf{Z}_k^T(n)\} \otimes E\{\mathbf{Z}_k^T(n)\}).\sigma_k \tag{25}$$

where $\sigma'_k = vec(\mathbf{\Sigma}'_k)$ and $\sigma_k = vec(\mathbf{\Sigma}_k)$. With definition of the $M^2 \times M^2$ matrix $\mathbf{G}_k$ as

$$\begin{aligned}\mathbf{G}_k = \mathbf{I} - \mu_k(E\{\mathbf{Z}_k^T(n)\} \otimes \mathbf{I}) - \mu_k(\mathbf{I} \otimes E\{\mathbf{Z}_k^T(n)\}) \\ + \mu_k^2(E(\{\mathbf{Z}_k^T(n)\} \otimes \{\mathbf{Z}_k^T(n)\}))\end{aligned} \tag{26}$$

equation (25) can be represented as

$$\sigma'_k = \mathbf{G}_k \sigma_k \tag{27}$$

also, by defining $\gamma_k$ through

$$\gamma_k = vec(E\{\mathbf{A}_k(n)x_k(n)\mathbf{W}_k(n)\mathbf{W}_k^T(n)x_k^T(n)\mathbf{A}_k^T(n)\}) \tag{28}$$

the second term in right hand side of (24) is given by

$$Tr(E\{\mathbf{Y}_k(n)\}) = \gamma_k^T.\sigma_k \tag{29}$$

with the above considerations, the recursion of (24) can now be stated as

$$E\{\|\tilde{h}_k(n)\|^2_{\sigma_k}\} = E\{\|\tilde{h}_{k-1}(n)\|^2_{\mathbf{G}_k.\sigma_k}\} + \mu_k^2\sigma_{v,k}^2\gamma_k^T\sigma_k \tag{30}$$

this equation involves spatially local information from two nodes. But we need to find the energy flow through the nodes. Using the same approach in [8], and adopting the linear relation as $\sigma_{k-1} = \mathbf{G}_k.\sigma_k$, the following equation is introduced by

$$E\{\|\tilde{h}_k(n)\|^2_{\sigma_k}\} = E\{\|\tilde{h}_{k-1}(n)\|^2_{\sigma_{k-1}}\} + g_k\sigma_k \tag{31}$$

where $g_k = \mu_k^2\sigma_{v,k}^2\gamma_k^T$. Now by applying the iterating
procedure in (31), we can find the relation $E\{\|\tilde{h}_{k-1}(n)\|^2_{\sigma_{k-1}}\}$
at node $k$ as

$$\begin{aligned}E\{\|\tilde{h}_{k-1}(n)\|^2_{\sigma_{k-1}}\} = E\{\|\tilde{h}_{k-1}(-1)\|^2_{(\mathbf{\Pi}_{k-1,1})^n\sigma_{k-1}}\} \\ + \alpha_{k-1}(\mathbf{I} + \ldots + (\mathbf{\Pi}_{k-1,1})^n)\sigma_{k-1}\end{aligned} \tag{32}$$

where

$$\mathbf{\Pi}_{k-1,l} = \mathbf{G}_{k+l-1}\mathbf{G}_{k+l}\ldots\mathbf{G}_J\mathbf{G}_1\ldots\mathbf{G}_{k-1}\ ,\ l = 1,2,\ldots,Q \tag{33}$$

and

$$\alpha_{k-1} = g_k\mathbf{\Pi}_{k-1,2} + g_{k+1}\mathbf{\Pi}_{k-1,3} + \ldots + g_{k-2}\mathbf{\Pi}_{k-1,J} + g_{k-1} \tag{34}$$

from this recursion, we will be able to evaluate the transient performance of dSPU-NLMS. When $\sigma_{k-1} = vec(\mathbf{I})$, the theoretical transient performance of the MSD at node $k$ is established. For $\sigma_{k-1} = vec(\mathbf{R}_k)$, the theoretical EMSE curve is obtained. Also, when $n$ goes to infinity, we obtain

$$E\{\|\tilde{h}_k(\infty)\|^2_{\sigma_k}\} = E\{\|\tilde{h}_{k-1}(\infty)\|^2_{\mathbf{G}_k.\sigma_k}\} + \mu_k^2\sigma_{v,k}^2\gamma_k^T\sigma_k \tag{35}$$

now, by iterating (35) for an incremental network, and using the definitions for $\mathbf{\Pi}_{k-1,1}$, and $\alpha_{k-1}$, the following expression for steady state behavior of dSPU-NLMS is established

$$E\{\|\tilde{h}_{k-1}(\infty)\|^2_{(\mathbf{I}-\mathbf{\Pi}_{k-1,1})\sigma_{k-1}}\} = \alpha_{k-1}\sigma_{k-1} \tag{36}$$

if $(\mathbf{I} - \mathbf{\Pi}_{k-1,1})\sigma_{k-1} = vec(\mathbf{I})$ and $(\mathbf{I} - \mathbf{\Pi}_{k-1,1})\sigma_{k-1} = vec(\mathbf{R}_k)$, the steady-state MSD, and EMSE expressions at node $k$ are established respectively. Doing so, we obtain the following results

$$\mathbf{MSD}_k = \alpha_{k-1}(\mathbf{I} - \mathbf{\Pi}_{k-1,1})^{-1}vec(\mathbf{I}) \tag{37}$$

$$\mathbf{EMSE}_k = \alpha_{k-1}(\mathbf{I} - \mathbf{\Pi}_{k-1,1})^{-1}vec(\mathbf{R}_k) \tag{38}$$

also from (19), we obtain that the steady-state MSE at node $k$ is given by

$$\mathbf{MSE}_k = \mathbf{EMSE}_k + \sigma_{v,k}^2 \tag{39}$$

It should be mentioned that the derived expressions can also be used for performance analysis of dLMS, and dNLMS algorithms. Selecting $\mathbf{A}_k(n) = \mathbf{I}$ leads to the dNLMS algorithm. Also, by choosing $\mathbf{A}_k(n)$ and $\mathbf{W}_k(n)$ equal to $\mathbf{I}$, the performance of dLMS can be analyzed.

## V. Mean and Mean-Square Stability of DSPU-NLMS

By substituting (19) in (17), and taking the expectation from both sides of (17), we obtain

$$E\{\tilde{h}_k(n)\} = [\mathbf{I} - \mu_k E\{\mathbf{Z}_k(n)\}]E\{\tilde{h}_{k-1}(n)\} \tag{40}$$

from (40), the convergence in the mean of dSPU-NLMS is guaranteed for any $\mu_k$ that satisfies

$$\mu_k < \frac{2}{\lambda_{\max}E\{\mathbf{Z}_k(n)\}} \tag{41}$$

The general recursion (30), is stable if the matrix $\mathbf{G}_k$ is stable. From (26), we know that $\mathbf{G}_k = \mathbf{I} - \mu_k\mathbf{M}_k + \mu_k^2\mathbf{N}_k$, where $\mathbf{M}_k = (E\{\mathbf{Z}_k^T(n)\} \otimes \mathbf{I}) - (\mathbf{I} \otimes E\{\mathbf{Z}_k^T(n)\})$ and $\mathbf{N}_k = E(\{\mathbf{Z}_k^T(n)\} \otimes \{\mathbf{Z}_k^T(n)\})$. Following the same approach in [14], the condition on $\mu_k$ to guarantee the convergence in the mean-square sense of dSPU-NLMS is obtained from (42)

$$0 < \mu_k < \min\{\frac{1}{\lambda_{\max}\mathbf{M}_k^{-1}\mathbf{N}_k}, \frac{1}{\lambda_{\max}(\mathbf{H}_k) \in \Re^+}\} \tag{42}$$

where

$$\mathbf{H}_k = \begin{bmatrix} \frac{1}{2}\mathbf{M}_k & -\frac{1}{2}\mathbf{N}_k \\ I & 0 \end{bmatrix} \tag{43}$$

## VI. Simulation Results

We justified the performance of dSPU-NLMS algorithm in an incremental network with $Q = 20$ nodes in a system identification set-up. The impulse response of the random unknown system has $M = 16$ taps. The correlated input signal, $x_k(n)$, at each node are generated by passing a white Gaussian noise process with unit variance through the a first order autoregressive model filter with the z-transfer
function $\sqrt{1-\alpha_k^2}\big/1-\alpha_k z^{-1}$ and $\alpha_k \in [0, 0.5)$. The noise sequence of each node is a white Gaussian process with variance $\sigma_{v,k}^2 \in (0, 0.1]$.

Fig. 3 shows the node profiles of $\sigma_{v,k}^2$ and $\alpha_k$. As we can see, node 10 has the highest correlated input. Therefore, we present the results for this node. In all simulations, the simulated MSE and MSD learning curves are obtained by ensemble averaging over 200 independent trials. Also, the steady-state MSE, MSD and EMSE values are obtained by averaging over 1000 steady-state samples from 100 independent realizations for each value of $\mu$ for a given algorithm. The step-sizes were chosen to get approximately the same steady-state MSE, and MSD in all

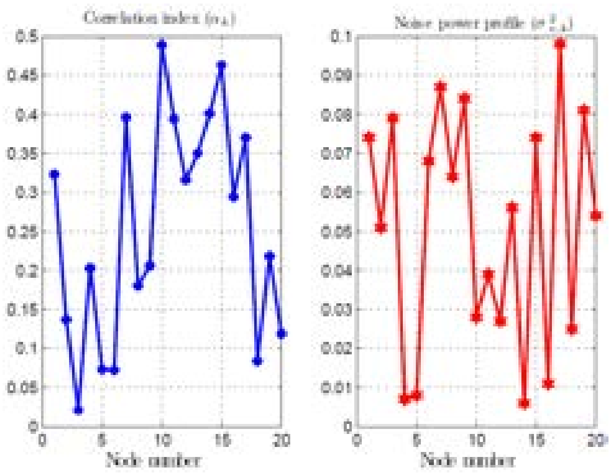

Fig. 3. Node profile: noise power and correlation index for the Gaussian data network.

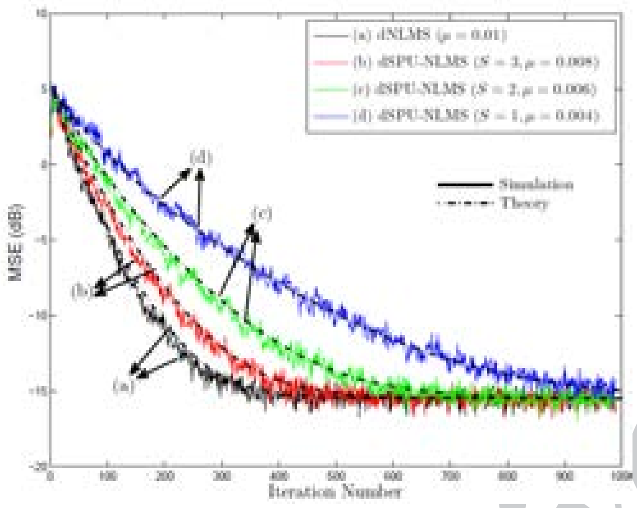

Fig. 4. The MSE learning curves of dNLMS and dSPU-NLMS adaptive algorithms for node 10.

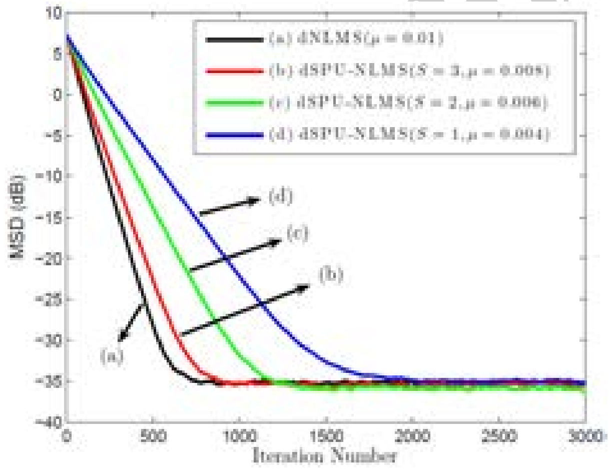

Fig. 5. The MSD learning curves of dNLMS and dSPU-NLMS adaptive algorithms for node 10.

learning curves. In dSPU-NLMS algorithm, the parameter $B$ was set to 4, and different values for $S$ (1, 2, and 3) have been chosen in simulations.

Fig. 4 presents the simulated and theoretical learning curves of dNLMS, and dSPU-NLMS with different values of $S$ for node 10. As we can see for large values of $S$, the performance of dSPU-NLMS will be close to the dNLMS. Also, good agreement between simulated and theoretical learning curves is observed. Fig. 5 compares the performance of dSPU-NLMS, and dNLMS for node 10 based on MSD curves. Again close performance to dNLMS can be seen for dSPU-NLMS especially for $S = 3$.

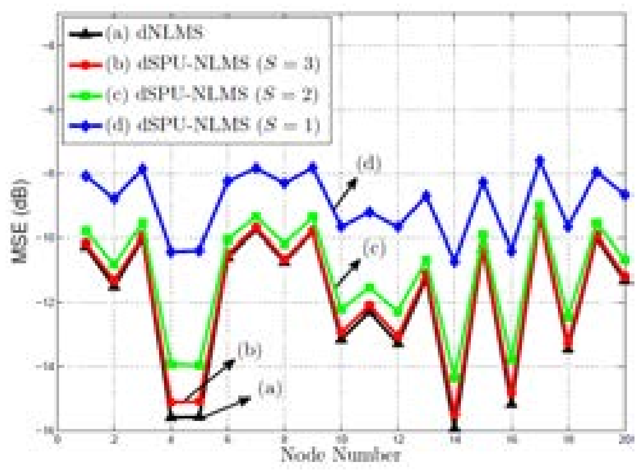

Fig. 6. Steady-state MSE for dNLMS, and dSPU-NLMS using $\mu_k = 0.5$.

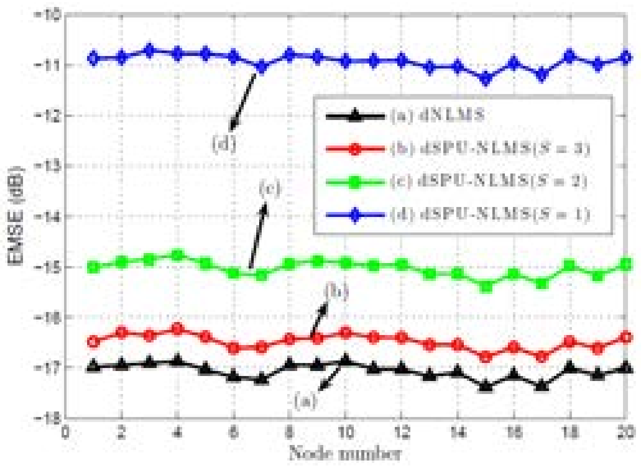

Fig. 7. Steady-state EMSE for dNLMS, and dSPU-NLMS using $\mu_k = 0.5$.

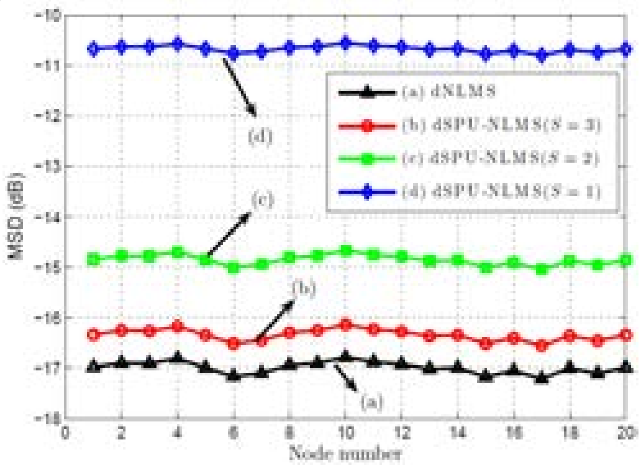

Fig. 8. Steady-state MSD for dNLMS, and dSPU-NLMS using $\mu_k = 0.5$.

Figs. 6-8 show the simulated steady-state MSE, MSD and EMSE values of dSPU-NLMS for each node with $\mu_k = 0.5$. The simulation results show that the steady-state values of dSPU-NLMS algorithm are close to steady-state values of dNLMS for $S = 2$ and 3. Figs. 9-11 show the simulated and theoretical steady-state MSE, MSD and EMSE values of dSPU-NLMS for each node with $\mu_k = 0.5$. The theoretical values are obtained from (37)-(39). The good agreement between simulated and theoretical values is observed.

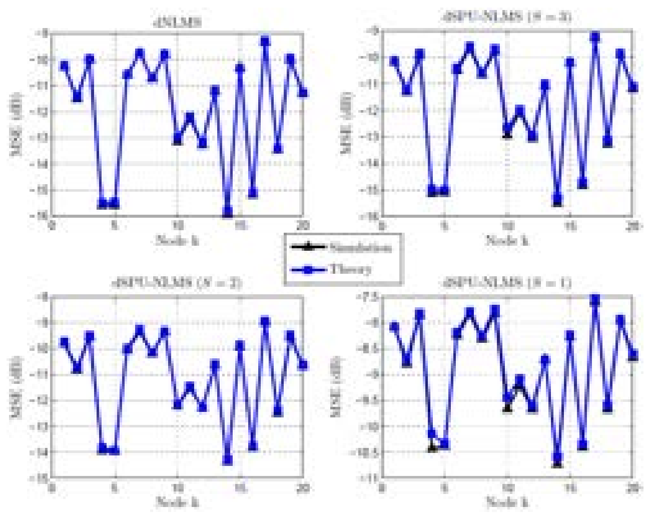

Fig. 9. Steady-state MSE for dNLMS, and dSPU-NLMS using $\mu_k = 0.5$ .

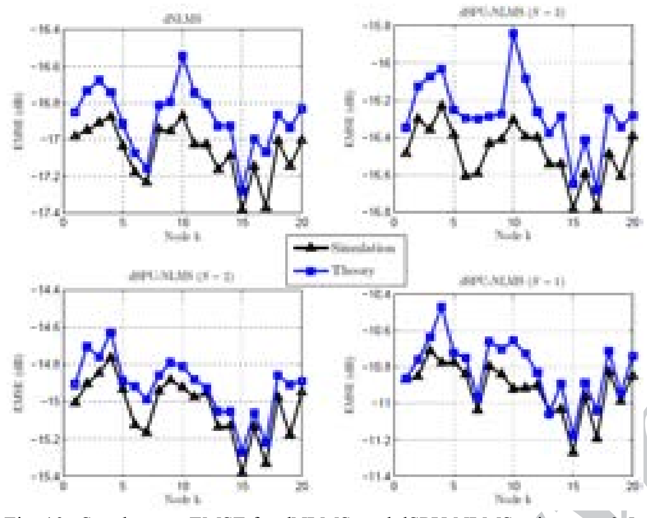

Fig. 10. Steady-state EMSE for dNLMS, and dSPU-NLMS using $\mu_k = 0.5$.

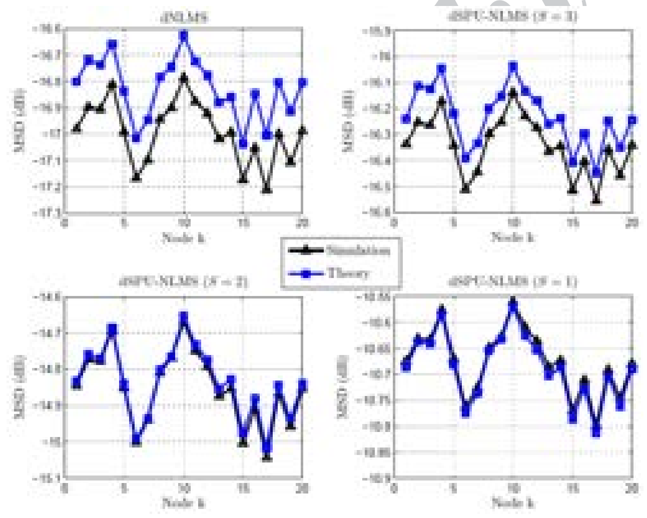

Fig. 11. Steady-state MSD for dNLMS, and dSPU-NLMS using $\mu_k = 0.5$.

Figs. 12-14 show the simulated steady-state MSE, MSD and EMSE of dSPU-NLMS as function of the step-size for node 10. The step-sizes change in the range [0.008,0.5] which are in the stability bound of dSPU-NlMS. Again the steady-state values of the dSPU-NLMS for different step-sizes are close to the dNLMS values. For $S = 2$ and 3, the better agreement can be seen. Also, Fig. 15 shows the simulated and theoretical steady-state MSE of dSPU-NLMS as function of the step-size for node 10. Again, the

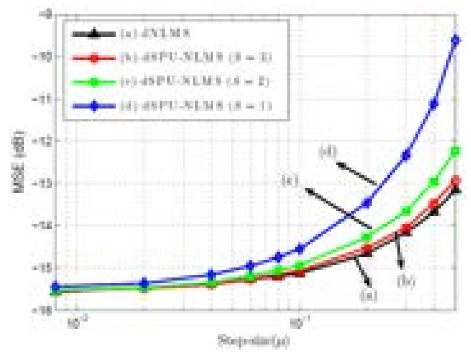

Fig. 12. Steady-state MSE curves of dNLMS and dSPU-NLMS at node 10 as a function of the step-size.

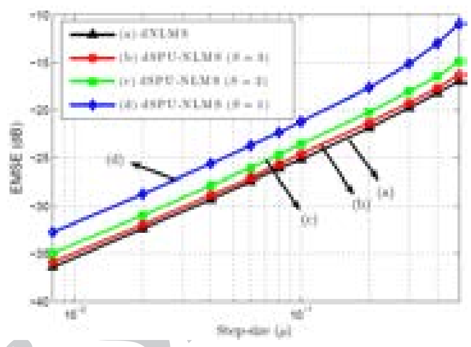

Fig. 13. Steady-state EMSE curves of dNLMS and dSPU-NLMS at node 10 as a function of the step-size.

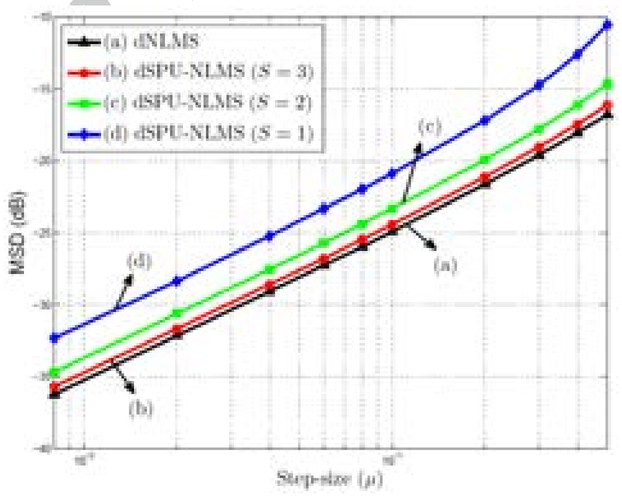

Fig. 14. Steady-state MSD curves of dNLMS and dSPU-NLMS at node 10 as a function of the step-size.

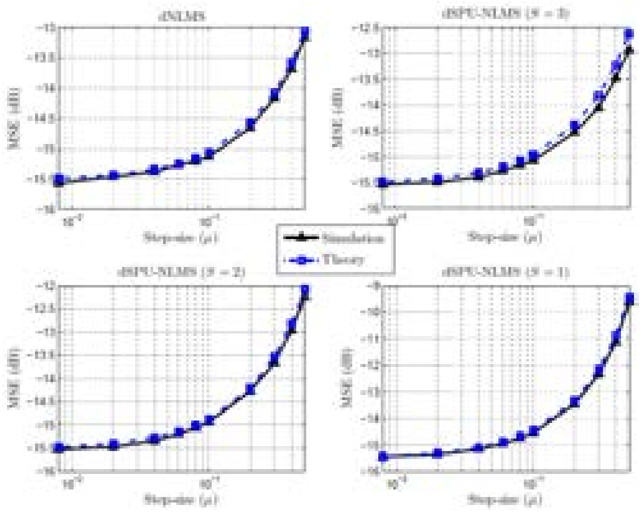

Fig. 15. Steady-state MSE curves of dNLMS and dSPU-NLMS at node 10 as a function of the step-size.

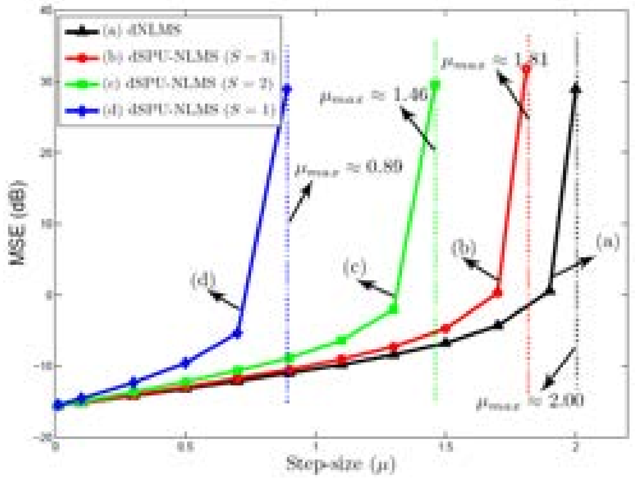

Fig. 16. Simulated MSE curves of dSPU-NLMS as a function of the step-size at node 10.

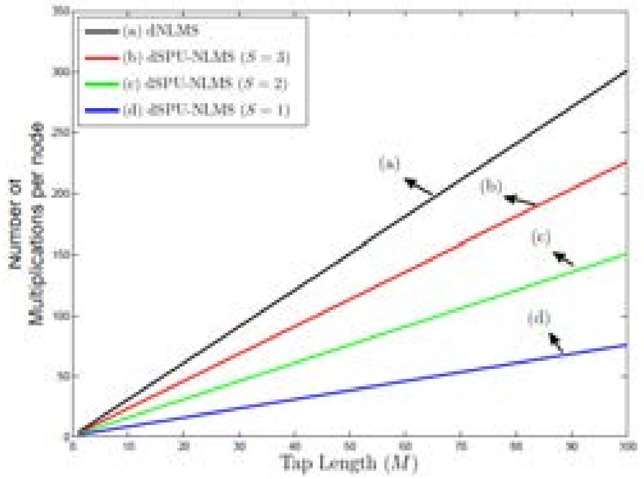

Fig. 17. Number of multiplications per node versus the filter length for dNLMS and dSPU-NLMS.

TABLE II
STABILITY BOUNDS OF THE STEP-SIZE FOR DNLMS AND DSPU-NLMS IN NODE 10

| Algorithm | $\frac{2}{\lambda_{\max} E\{\mathbf{Z}_{10}(n)\}}$ | $\frac{1}{\lambda_{\max} \mathbf{M}_{10}^{-1}\mathbf{N}_{10}}$ | $\frac{1}{\lambda_{\max}(\mathbf{H}_{10}) \in \Re^{+}}$ | $\mu_{\max}$ |
|---|---|---|---|---|
| dNLMS | 12.3179 | 2.0002 | 6.5639 | 2.0002 |
| dSPU-NLMS $(S=3)$ | 12.3207 | 1.8113 | 6.4507 | 1.8113 |
| dSPU-NLMS $(S=2)$ | 12.6387 | 1.4642 | 6.4828 | 1.4642 |
| dSPU-NLMS $(S=1)$ | 11.3935 | 0.8976 | 6.0036 | 0.8976 |

good agreement between simulated and theoretical learning curves is observed.

Table II shows the stability bounds of the step-size for dSPU-NLMS in node 10. The theoretical values have been obtained by (41) and (42). Different values for $S$ were selected. As we can see, by increasing $S$, the stability bound in dSPU-NLMS increases. Fig. 16 shows the simulated steady-state MSE values versus the step-size for dSPU-NLMS in node 10. The step-size changes from 0.001 to $\mu_{\max}$. The results show that the theoretical values from Table II are suitable estimations to predict $\mu_{\max}$.

Finally in Fig. 17, the number of multiplications per node versus the filter length $(M)$ for dNLMS and dSPUNLMS have been presented. As we can see, by increasing the filter length the number of multiplications will increase. This figure shows that dSPU-NLMS has lower computational load than dNLMS.

## VII. SUMMARY AND CONCLUSION

In this paper we applied SPU-NLMS algorithm for distributed estimation in an incremental network. The dSPU-NLMS had close performance to dNLMS algorithm. The computational complexity of this algorithm was lower than dNLMS due to partial update approach. We demonstrated the good performance of this algorithm in an incremental network for steady-state error and convergence speed through several simulation results.

## REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102-114, Aug. 2002.
[2] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed, "Detection, classification, and tracking of targets," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17-29, Mar. 2002.
[3] D. L. Duttweiler, "Proportionate normalized least-mean-squares adaptation in echo cancelers," *IEEE Trans. Speech., Audio Processing*, vol. 8, no. 5, pp. 508-518, Sep. 2000.
[4] H. Deng and M. Doroslovacki, "Improving convergence of the PNLMS algorithm for sparse impluse response identification," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 181-184, Mar. 2005.
[5] S. L. Gay, "An efficient, fast converging adaptive filter for network echo cancellation," in *Proc. Asilomar Conf. Signals, Systems, Comput.*, vol. 1, pp. 394-398, Pacific Grove, Calif, US, 1-4 Nov. 1998.
[6] P. A. Naylor, J. Cui, and M. Brookes, "Adaptive algorithms for sparse echo cancellation," *Signal Processing*, vol. 86, pp. 1182-1192, 2006.
[7] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Processing*, vol. 55, no. 8, pp. 4064-4077, Aug. 2007.
[8] L. Li, J. A. Chambers, C. G. Lopes, and A. H. Sayed, "Distributed estimation over an adaptive incremental network based on the affine projection algorithm," *IEEE Trans. Signal Processing*, vol. 58, no. 1, pp. 151-164, Jan. 2010.
[9] K. Dogancay, *Partial-Update Adaptive Signal Processing, Design Analysis, and Implementation*, Academic Press, 2009.
[10] K. Dogancay and O. Tanrikulu, "Adaptive filtering algorithms with selective partial updates," *IEEE Trans. Circuits, Syst. II: Analog and Digital Signal Processing*, vol. 48, no. 8, pp. 762-769, Aug. 2001.
[11] P. Mowlaee and M. H. Kahaei, "Convergence and tracking behavior of the new SPVSNLMS algorithm," in *Proc. 2nd Information and Communication Technologies,* vol. 1, pp. 1184-1185, Damascus, Syria, Apr. 2006.
[12] P. M. B. Mahale, "SVSPNLMS algorithm for acoustic echo cancellation," in *Proc. Information and Communication Technologies: from Theory to Applications,* 6 pp., Damascus, Syria, 7-11 Apr. 2008.
[13] D. E. Knuth, *Art of Computer Programming, Vol. 3: Sorting and Searching*, 2nd Ed. Reading, MA: Addison-Wesley, 1973.
[14] H. C. Shin and A. H. Sayed, "Mean-square performance of a family of affine projection algorithms," *IEEE Trans. Signal Processing*, vol. 52, no. 1, pp. 90-102, Jan. 2004.

**Mohammad Shams Esfand Abadi** was born in Tehran, Iran, on September 18, 1978. He received the B.S. degree in Electrical Engineering from Mazandaran University, Mazandaran, Iran and the M.S. degree in Electrical Engineering from Tarbiat Modares University, Tehran, Iran in 2000 and 2002, respectively, and the Ph.D. degree in Biomedical

Engineering from Tarbiat Modares University, Tehran, Iran in 2007. Since 2004 he has been with the Faculty of Electrical and Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran.

His research interests include digital filter theory and adaptive signal processing algorithms.

**Ali Reza Danaee** was born in Ramhormoz, Iran, in 1984. He received the B.Sc. degree in Electrical Engineering from the University of Kurdistan, Sanandaj, in 2008, and the M.Sc. degree in Electrical Engineering from the Shahid Rajaee Teacher Training University, Tehran, in 2013.

His research interests include signal processing, adaptive filters, and distributed networks.

**Mahmood Seifouri** received the B.Sc. (Hons) and Ph.D. degrees in Electrical and Electronic Engineering from the University of Wales College of Cardiff, UK, in 1985 and 1989, respectively.

After spending two years as a Lecturer at Brighton University, UK, Dr. Seifouri moved to Iran University of Science and Technology in 1991. After spending 9 years at IUST, in 2000, he was employed as a Senior Development Engineer at Bookham Technology, UK, where he was primarily involved in research & development projects in the field of electrical and electronic engineering. In September 2001, he joined Optinetrics, CA, USA, as a Senior Development Engineer and continued with his work there. Since 2006, Dr. Seifouri is with Shahid Rajaee Teacher Training University, Tehran, Iran, as an Assistant Professor at the Department of Electrical and Computer Engineering.