Low computational complexity family of affine projection algorithms over adaptive distributed incremental networks

Mohammad Shams Esfand Abadi*, Ali-Reza Danaee[1]

Faculty of Electrical and Computer Engineering, Shahid Rajaee Teacher Training University, P.O. Box 16785-163, Tehran, Iran

ARTICLE INFO

Article history:
Received 26 January 2013
Accepted 5 July 2013

Keywords:
Affine projection algorithm
Adaptive distributed estimation
Incremental network
Selective partial update
Dynamic selection
Mean-square performance

ABSTRACT

This paper presents the problem of distributed estimation in an incremental network based on the family of affine projection (AP) adaptive algorithms. The distributed selective partial update normalized least mean squares (dSPU-NLMS), the distributed SPU-AP algorithm (dSPU-APA), the distributed selective regressor APA (dSR-APA), the distributed dynamic selection of APA (dDS-APA), dSPU-SR-APA and dSPU-DS-APA are introduced in a unified way. These algorithms have low computational complexity feature and close convergence speed to ordinary distributed adaptive algorithms. In dSPU-NLMS and dSPU-APA, the weight coefficients are partially updated at each node during the adaptation. In dSR-APA, the optimum number of input regressors is selected during the weight coefficients update. The dynamic selection of input regressors is used in dDS-APA. dSPU-SR-APA and dSPU-DS-APA combine SPU with SR and DS approaches. In these algorithms, the weight coefficients are partially updated and the input regressors are optimally/dynamically selected at every iteration for each node. In addition, a unified approach for mean-square performance analysis of each individual node is presented. This approach can be used to establish a performance analysis of classical distributed adaptive algorithms as well. The theoretical expressions for stability bounds, transient, and steady-state performance analysis of various distributed APAs are introduced. The validity of the theoretical results and the good performance of dAPAs are demonstrated by several computer simulations.

1. Introduction

One 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 [1,2].

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 [3]. 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 [3,4]. Unfortunately, in comparison with dLMS and dNLMS, the computational complexity of dAPA is high.

In some of applications such as adaptive distributed networks, a high computational load is needed to achieve an acceptable performance. Therefore the computational complexity is an important problem in these applications. Several single adaptive filter algorithms such as adaptive filter algorithms with selective partial updates have been proposed to reduce the computational complexity. These algorithms update only a subset of filter coefficients at each time iteration. The Max-NLMS [5], the N-Max NLMS [6,7] (N is the number of filter coefficients to update), and the variants of the selective partial update normalized least mean square (SPU-NLMS) [8,9] are important examples of this family of adaptive filter algorithms. The SPU approach was also successfully extended to APA in [8].

In the case of APA, other algorithms such as selective regressor APA (SR-APA) and dynamic selection of APA (DS-APA) were

* Corresponding author. Tel.: +98 21 22970003; fax: +98 21 22970003.
E-mail addresses: mshams@srttu.edu (M.S.E. Abadi), ar.danaee@srttu.edu (A.-R. Danaee).
[1] Tel.: +98 21 22970003; fax: +98 21 22970003.

proposed to reduce the computational complexity [10,11]. In SR-APA, the input regressors are optimally selected at each iteration. The dynamic selection of input regressors during the filter coefficients update were introduced in [11]. Also by combination of SPU and SR approaches, SPU-SR-APA was suggested in [12]. In comparison with APA, the introduced algorithms are good convergence speed, low steady-state error, and low computational complexity.

In the present study, we therefore focus on a new class of adaptive algorithms-based distributed learning mechanism in incremental networks. The distributed SPU-APA (dSPU-APA), the distributed SR-APA (dSR-APA) and the distributed DS-APA (dDS-APA) are introduced in a unified way. Also, dSPU-SR-APA, and dSPU-DS-APA are presented by combining SPU and DS approaches. These algorithms have good convergence speed, low steady-state error, and low computationally complexity features. In the following, a unified approach to performance analysis of the spatial and temporal family of affine projection algorithms 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 dAPAs are introduced. The analysis is based on spatial-temporal energy conservation relation which incorporates the space–time structure of the data. For the classical dAPA, this anslysis was presented in [4]. But the present analysis is the unified formalism for transient performance of various adaptive distributed networks. The general expressions for steady-state MSE and MSD are established,

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

- The establishment of the family of dAPAs. In comparison with classical dAPA in [4], the introduced dAPAs have close performance to dAPA with low computational complexity feature. In dSPU-APA, and dSPU-NLMS, the coefficients are partially updated at each node for every iteration. In dSR-APA, and dDS-APA, the number of recent regressors are optimally selected. By combination of SPU and DS approaches, the dSPU-SR-APA and dSPU-DS-APA are established. In these algorithms the coefficients are partially updated and the number of recent regressors are optimally selected at each node for every iteration.
- Mean-square performance analysis of the family of dAPAs in a unified way. The general theoretical expressions for transient and steady-state performance of the network are obtained. The presented expressions can also be used to predict the performance of dLMS, dNLMS, and dAPA.
- Analysis of the mean and mean-square stability bounds for the family of dAPAs.
- Demonstrating the presented algorithms in an incremental network. The performance of dNLMS, dAPA, dSPU-NLMS, dSPU-APA, dSR-APA, dDS-APA, dSPU-SR-APA, and dSPU-DS-APA are compared and analyzed. The validity of the theoretical relations is also justified in both transient and steady-state.

This paper is organized as follows. In Section 2, the generic adaptive weight coefficients update equation in distributed network is introduced. Section 3 describes the establishment of classical adaptive distributed networks based on the generic update equation. The family of dAPAs is presented in Section 4. In the next section, the mean-square performance analysis of each node is explained. Section 6 presents the stability bounds of step-size for various dAPAs. The computational complexity of dAPAs is discussed in Section 7. Finally, the paper ends with a comprehensive set of simulations supporting the validity of the results.

Throughout the paper, the following notations are used:

$\lvert . \rvert$	Norm of a scalar.
$\lVert . \rVert^2$	Squared Euclidean norm of a vector.
$\lVert \mathbf{t} \rVert^2_{\mathbf{\Sigma}}$	$\mathbf{\Sigma}$-Weighted Euclidean norm of a column vector $\mathbf{t}$ defined as $\mathbf{t}^T\mathbf{\Sigma}\mathbf{t}$.
vec($\mathbf{T}$)	Creates an $M^2 \times 1$ column vector $\mathbf{t}$ through stacking the columns of the $M \times M$ matrix $\mathbf{T}$.
vec($\mathbf{t}$)	Creates an $M \times M$ matrix $\mathbf{T}$ from the $M^2 \times 1$ column vector $\mathbf{t}$.
$\mathbf{A} \otimes \mathbf{B}$	Kronecker product of matrices $\mathbf{A}$ and $\mathbf{B}$.
Tr(.)	Trace of a matrix.
$(.)^{-1}$	Inverse of a square matrix.
$(.)^T$	Transpose of a vector or a matrix.
diag{ . }	Has the same meaning as the MATLAB operator with the same name: If its argument is a vector, a diagonal matrix with the diagonal elements given by the vector argument results. If the argument is a matrix, its diagonal is extracted into a resulting vector.
$E\{ \cdot \}$	Expectation operator.

2. The generic adaptive weight update equation

Due to special features of family of affine projection algorithms such as high convergence speed, low computational complexity and low steady state mean square error, we apply different APAs for a distributed estimation in an incremental network. The classical adaptive distributed algorithms, dSPU-NLMS, dSPU-APA, dSR-APA, dDS-APA, dSPU-SR-APA, and dSPU-DS-APA can be established based on a unified formalism. By selecting the proper matrices and parameters in this approach, various dAPAs are obtained. This new proposition will be useful to show the differences and similarities of dAPAs and to analyze the performance of dAPAs in a unified way.

The weight vector update equation for the family of dAPAs in an incremental network over J nodes is introduced as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \sum_{k=1}^{J} \mu_k \mathbf{C}_k(n)\mathbf{X}_k(n)\mathbf{Y}_k(n)\mathbf{e}_k(n), \tag{1}$$

where

$$\mathbf{e}_k(n) = \mathbf{d}_k(n) - \mathbf{X}_k^T(n)\mathbf{h}_{k-1}(n), \tag{2}$$

is the error vector, and μ_k is the step-size of each node. In (2), $\mathbf{X}_k(n) = [\mathbf{x}_k(n), \mathbf{x}_k(n-1), \ldots, \mathbf{x}_k(n-(P-1))]$ is the $M \times P$ block data signal matrix and $\mathbf{d}_k(n) = [d_k(n), d_k(n-1), \ldots, d_k(n-(P-1))]^T$ is the $P \times 1$ data vector where $\mathbf{x}_k(n) = [x_k(n), x_k(n-1), \ldots, x_k(n-M+1)]^T$. We assume the linear data model between the unknown system vector $\mathbf{h}_t$, $\mathbf{d}_k(n)$, and $\mathbf{X}_k(n)$ as $\mathbf{d}_k(n) = \mathbf{X}_k^T(n)\mathbf{h}_t + \mathbf{v}_k(n)$, where $\mathbf{v}_k(n)$ is a temporally and spatially independent noise sequence with variance $\sigma^2_{v,k}$ at each node. By selecting matrices $\mathbf{C}_k(n)$ and $\mathbf{Y}_k(n)$ from Table 1, the family of distributed affine projection adaptive algorithms such as dAPA, dSPU-APA, dSR-APA, dDS-APA, dSPU-SR-APA, and dSPU-DS-APA as well as classical adaptive distributed algorithms can be established. Fig. 1 shows the learning algorithm in an incremental network for the family of dAPAs [4]. Based on the local data ($\mathbf{d}_k(n)$, $\mathbf{X}_k(n)$), and $\mathbf{h}_{k-1}(n)$ 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 $\mathbf{h}_J(n)$ is used as the global estimation for $\mathbf{w}(n)$. This value is used as the initial local estimation $\mathbf{h}_1(n+1)$ for the next time instant $n+1$. The same as [4], the pseudo-code implementation of dAPAs has been described in Table 2.

3. Derivation of classical distributed adaptive algorithms

We can now make specific choices for the matrices $\mathbf{C}_k(n)$ and $\mathbf{Y}_k(n)$ as well as for the parameter P. Different adaptive distributed algorithms can be viewed as specific instantiations of the generic adaptive weight update equation. By setting the parameter P equal to 1, and substituting the matrices $\mathbf{C}_k(n)$ and $\mathbf{Y}_k(n)$ from Table 1 in the generic adaptive weight update equation, dLMS, dNLMS, dAPA,

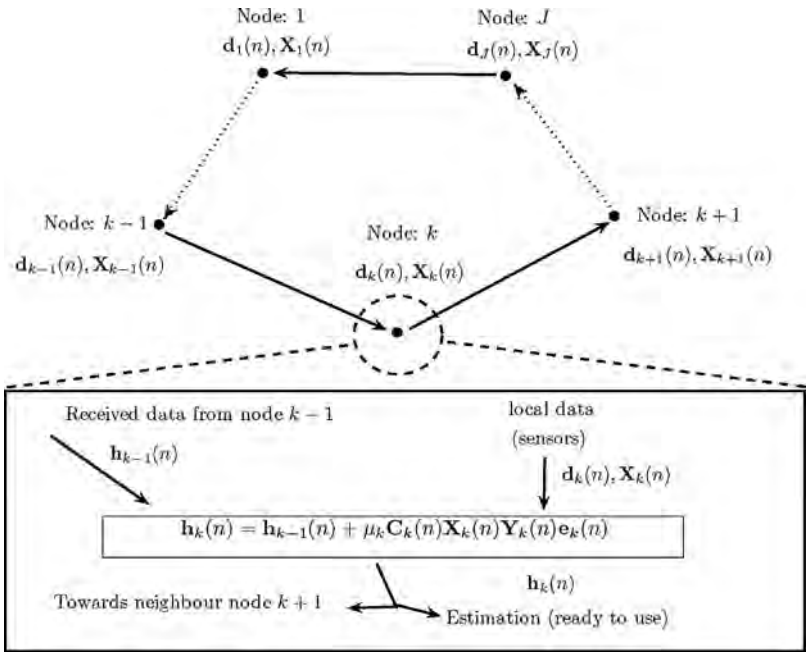

Fig. 1. Data processing of the family of dAPAs in an incremental network.

Table 1
Adaptive distributed algorithms.

Algorithm	P	$\mathbf{C}_k(n)$	$\mathbf{Y}_k(n)$
dLMS	$P=1$	$\mathbf{I}$	$\mathbf{I}$
dNLMS	$P=1$	$\mathbf{I}$	$\frac{1}{\|\mathbf{x}_k(n)\|^2}$
dAPA	$P\leq M$	$\mathbf{I}$	$(\mathbf{X}_k^T(n)\mathbf{X}_k(n))^{-1}$
dDR-LMS	$P\leq M$	$\mathbf{I}$	$\mathbf{I}$
dNDR-LMS	$P\leq M$	$\mathbf{I}$	$\mathrm{diag}\{1/\|\mathbf{x}_k(n)\|^2, \ldots, 1/\|\mathbf{x}_k(n-P+1)\|^2\}$
dRLS	$P=1$	$\left[\sum_{i=0}^{n}\lambda^{n-i}\mathbf{x}_k(i)\mathbf{x}_k^T(i)\right]^{-1}$	$\mathbf{I}$
dTDAA	$P=1$	$\mathbf{T}\cdot\left[\mathrm{diag}\left\{\mathrm{diag}\left\{\sum_{i=0}^{n}\lambda^{n-i}\mathbf{T}^T\mathbf{x}_k(i)\mathbf{x}_k^T(i)\mathbf{T}\right\}\right\}\right]^{-1}\cdot\mathbf{T}^T$	$\mathbf{I}$
dSPU-NLMS	$P=1$	$\mathbf{A}_k(n)$	$\frac{1}{\|\mathbf{A}_k(n)\mathbf{x}_k(n)\|^2}$
dSPU-APA	$P\leq M$	$\mathbf{A}_k(n)$	$(\mathbf{X}_k^T(n)\mathbf{A}_k(n)\mathbf{X}_k(n))^{-1}$
dSR-APA	$P\leq M$	$\mathbf{I}$	$\mathbf{B}_k(n)(\mathbf{B}_k^T(n)\mathbf{X}_k^T(n)\mathbf{X}_k(n)\mathbf{B}_k(n))^{-1}\mathbf{B}_k^T(n)$
dSPU-SR-APA	$P\leq M$	$\mathbf{A}_k(n)$	$\mathbf{B}_k(n)(\mathbf{B}_k^T(n)\mathbf{X}_k^T(n)\mathbf{A}_k(n)\mathbf{X}_k(n)\mathbf{B}_k(n))^{-1}\mathbf{B}_k^T(n)$

Table 2
Pseudo-code implementation of dAPAs.

For each time instant $n\geq 0$ repeat:
$k=1,\ldots,J$
$\mathbf{h}_0(n)=\mathbf{w}(n-1)$
$\mathbf{h}_k(n)=\mathbf{h}_{k-1}(n)+\mu_k\mathbf{C}_k(n)\mathbf{X}_k(n)\mathbf{Y}_k(n)\mathbf{e}_k(n)$
end
$\mathbf{w}(n)=\mathbf{h}_J(n)$

the distributed data-reusing LMS (dDR-LMS), and the distributed normalized DR-LMS (dNDR-LMS) [13] are established, respectively. This table shows that based on (1), the distributed recursive least squares (dRLS)[2] and the distributed transform domain adaptive algorithm (dTDAA)[3] [14] can also be established.

4. Derivation of the family of distributed adaptive algorithms with low computational complexity

In this section, based on the generic update equation, the family of low computational complexity of dAPAs is presented. These algorithms are dSPU-NLMS, dSPU-APA, dSR-APA, dDS-APA, dSPU-SR-APA, and dSPU-DS-APA which have been presented in Table 1.

4.1. Derivation of dSPU-NLMS

The single SPU-NLMS algorithm was presented in [8]. In this algorithm the filter coefficients are partially updated at every iteration. From (1), the generic weight coefficients update equation for $P=1$, $\mathbf{C}_k(n)=\mathbf{A}_k(n)$, and $\mathbf{Y}_k(n)=\mathbf{I}$, can be stated as

$$\mathbf{w}(n)=\mathbf{w}(n-1)+\sum_{k=1}^{J}\mu_k\mathbf{A}_k(n)\mathbf{x}_k(n)e_k(n). \tag{3}$$

In the adaptive distributed algorithms with selective partial updates, the $M\times 1$ vector of weight coefficients is partitioned into K blocks ($K=(M/L)$ and is an integer) each of length L, $\mathbf{w}(n)=[\mathbf{w}_1^T(n),\mathbf{w}_2^T(n),\ldots,\mathbf{w}_K^T(n)]^T$, and in each iteration a subset of these blocks (N blocks out of K blocks) is updated at each node. The $\mathbf{A}_k(n)$ matrix is the $M\times M$ diagonal matrix with the $\mathbf{1}$ and $\mathbf{0}$ blocks each of length L on the diagonal and the positions of 1's on the diagonal determine which coefficients should be updated in every iteration at each node.

[2] The parameter λ is the forgetting factor.
[3] The matrix $\mathbf{T}$ is an $M\times M$ orthogonal transform matrix.

By partitioning the regressor vector $\mathbf{x}_k(n)$ into K blocks each of length L as

$$\mathbf{x}_k(n) = [\mathbf{x}_{k,1}^T(n), \mathbf{x}_{k,2}^T(n), \ldots, \mathbf{x}_{k,K}^T(n)]^T, \tag{4}$$

the positions of **1** blocks (N blocks and $N \le K$) on the diagonal of $\mathbf{A}_k(n)$ matrix for each iteration in dSPU-NLMS adaptive algorithm is determined by the following procedure:

1 The $\|\mathbf{x}_{k,i}(n)\|^2$ values are sorted for $1 \le i \le K$.
2 The i values that determine the positions of **1** blocks correspond to N largest values of $\|\mathbf{x}_{k,i}(n)\|^2$.

4.2. Derivation of dSPU-APA

The selective partial update approach in single adaptive filter was successfully extended to APA in [8]. Following the same approach in [8], the weight vector update equation for dSPU-APA is introduced as

$$\mathbf{w}_F(n) = \mathbf{w}_F(n-1) + \sum_{k=1}^{J} \mu_k \mathbf{X}_{k,F}(n)(\mathbf{X}_{k,F}^T(n)\mathbf{X}_{k,F}(n))^{-1}\mathbf{e}_k(n), \tag{5}$$

where $F = \{j_1, j_2, \ldots, j_N\}$ denote the indexes of the N blocks out of K blocks that should be updated at every adaptation, $\mathbf{w}_F(n) = [\mathbf{w}_{j_1}^T(n), \mathbf{w}_{j_2}^T(n), \ldots, \mathbf{w}_{j_N}^T(n)]^T$, and

$$\mathbf{X}_{k,F}(n) = [\mathbf{X}_{k,j_1}^T(n), \mathbf{X}_{k,j_2}^T(n), \ldots, \mathbf{X}_{k,j_N}^T(n)]^T, \tag{6}$$

is the $NL \times P$ matrix and

$$\mathbf{X}_{k,i}(n) = [\mathbf{x}_{k,i}(n), \mathbf{x}_{k,i}(n-1), \ldots, \mathbf{x}_{k,i}(n-(P-1))] \tag{7}$$

is the $L \times P$ matrix. The indexes of F are obtained by the following procedure:

1 Compute the following values for $1 \le i \le K$

$$\mathrm{Tr}(\mathbf{X}_{k,i}^T(n)\mathbf{X}_{k,i}(n)). \tag{8}$$

2 The indexes of F correspond to N largest values of (8).

In dSPU-APA, the weight coefficients are partially updated at every iteration for each node. The dSPU partial rank algorithm (dSPU-PRA) can be established when the adaptation of the weight coefficients is performed only once every P iterations. Eq. (5) can be represented in the form of full update equation as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \sum_{k=1}^{J} \mu_k \mathbf{A}_k(n)\mathbf{X}_k(n)(\mathbf{X}_k^T(n)\mathbf{A}_k(n)\mathbf{X}_k(n))^{-1}\mathbf{e}_k(n), \tag{9}$$

where the $\mathbf{A}_k(n)$ matrix 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 (N blocks and $N \le K$) on the diagonal of $\mathbf{A}_k(n)$ matrix for each iteration in dSPU-APA are determined by the indexes of F.

4.3. Derivation of dSR-APA

In [10], another low computational complexity version of APA was introduced. In SR-APA, the number of input regressors are selected at each iteration. This approach can be extended to adaptive distributed networks where the number of input regressors at each node are optimally selected during the adaptation. Suppose $\mathbf{X}_{k,G}(n)$ is the $M \times Q$ matrix of the input signal and $\mathbf{d}_{k,G}(n)$ is the $Q \times 1$ vector of the desired signal which are defined as

$$\mathbf{X}_{k,G}(n) = [\mathbf{x}_k(n-i_1), \mathbf{x}_k(n-i_2), \ldots, \mathbf{x}_k(n-i_Q)], \tag{10}$$

$$\mathbf{d}_{k,G}(n) = [d_k(n-i_1), d_k(n-i_2), \ldots, d_k(n-i_Q)]^T, \tag{11}$$

where $G = \{i_1, i_2, \ldots, i_Q\}$ denote the Q subset (subset with Q member) of the set $\{0, 1, \ldots, P-1\}$. From [10], the weight coefficients update equation for dSR-APA can be stated as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \sum_{k=1}^{J} \mu_k \mathbf{X}_{k,G}(n)(\mathbf{X}_{k,G}^T(n)\mathbf{X}_{k,G}(n))^{-1}\mathbf{e}_{k,G}(n), \tag{12}$$

where

$$\mathbf{e}_{k,G}(n) = \mathbf{d}_{k,G}(n) - \mathbf{X}_{k,G}^T(n)\mathbf{h}_{k-1}(n). \tag{13}$$

The indexes of G are obtained by the following procedure:

1 Compute the following values for $0 \le i \le P-1$

$$\frac{e_k^2(n-i)}{\|\mathbf{x}_k(n-i)\|^2}, \tag{14}$$

where $\mathbf{e}_k(n) = [e_k(n), e_k(n-1), \ldots, e_k(n-(P-1))]^T$.
2 The indexes of G correspond to Q largest values of (14).

Eq. (12) can also be represented as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \sum_{k=1}^{J} \mu_k \mathbf{X}_k(n)\mathbf{B}_k(n)(\mathbf{B}_k^T(n)\mathbf{X}_k^T(n)\mathbf{X}_k(n)\mathbf{B}_k(n))^{-1} \times \mathbf{B}_k^T(n)\mathbf{e}_k(n), \tag{15}$$

where $\mathbf{B}_k(n) = [\mathbf{1}_{i_1}, \mathbf{1}_{i_2}, \ldots, \mathbf{1}_{i_Q}]$ is the $P \times Q$ matrix and $\mathbf{1}_{i_q} = [0, \ldots, 0, 1, 0, \ldots, 0]^T$ is the $P \times 1$ vector with the element 1 in the position i_q.

4.4. Derivation of dDS-APA

In [11], the affine projection algorithm with dynamic selection of input vectors was presented. In this algorithm, the optimum selection of the input vectors is derived by the largest decrease of the mean-square deviation. Let $G_{Q(n)} = \{i_1, i_2, \ldots, i_{Q(n)}\}$ denote a subset with $Q(n)$ members of the set $\{0, 1, \ldots, P-1\}$, where $Q(n)$ is defined as the number of the selected input vectors at iteration n. Following the same strategy in [11], the weight vector update equation of dDS-APA for $Q(n) \neq 0$ is introduced as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \sum_{k=1}^{J} \mu_k \mathbf{X}_{k,G_{Q(n)}}(n)(\mathbf{X}_{k,G_{Q(n)}}^T(n)\mathbf{X}_{k,G_{Q(n)}}(n))^{-1} \times \mathbf{e}_{k,G_{Q(n)}}(n). \tag{16}$$

For $Q(n) = 0$, the weight coefficients do not change. The indexes of $G_{Q(n)}$ correspond to $Q(n)$ members of $e_k(n-i)$ that satisfy the following condition:

$$e_k^2(n-i) > \frac{2\sigma_{v,k}^2}{2-\mu_k}, \tag{17}$$

where $\sigma_{v,k}^2$ is variance of measurement noise at each node. Eq. (16) can be written as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \sum_{k=1}^{J} \mu_k \mathbf{X}_k(n)\mathbf{B}_k(n)(\mathbf{B}_k^T(n)\mathbf{X}_k^T(n)\mathbf{X}_k(n)\mathbf{B}_k(n))^{-1} \times \mathbf{B}_k^T(n)\mathbf{e}_k(n), \tag{18}$$

where now $\mathbf{B}_k(n)=[\mathbf{1}_{i_1},\mathbf{1}_{i_2},\ldots,\mathbf{1}_{i_{Q(n)}}]$ is the $P\times Q(n)$ matrix and $\mathbf{1}_{i_q}=[0,\ldots,0,1,0,\ldots,0]^T$ is the $P\times 1$ vector with the element 1 in the position i_q.

4.5. Derivation of dSPU-SR-APA and dSPU-DS-APA

By combining the SPU and SR approaches, dSPU-SR-APA can be established. Defining $SL\times Q$ input signal matrix through

$$\mathbf{X}_{k,FG}(n)=\begin{pmatrix}\mathbf{x}_{k,j_1}(n-i_1) & \ldots & \mathbf{x}_{k,j_1}(n-i_Q)\\ \mathbf{x}_{k,j_2}(n-i_1) & \ldots & \mathbf{x}_{k,j_2}(n-i_Q)\\ \vdots & \ddots & \vdots\\ \mathbf{x}_{k,j_S}(n-i_1) & \ldots & \mathbf{x}_{k,j_S}(n-i_Q)\end{pmatrix} \tag{19}$$

the update equation for the dSPU-SR-APA is given by

$$\mathbf{w}_F(n)=\mathbf{w}_F(n-1)+\sum_{k=1}^{J}\mu_k\mathbf{X}_{k,FG}(n)(\mathbf{X}_{k,FG}^T(n)\mathbf{X}_{k,FG}(n))^{-1}\mathbf{e}_{k,G}(n) \tag{20}$$

This weight update equation can also be stated as

$$\mathbf{w}(n)=\mathbf{w}(n-1)+\sum_{k=1}^{J}\mu_k\mathbf{A}_k(n)\mathbf{X}_k(n)\mathbf{B}_k(n)\times(\mathbf{B}_k^T(n)\mathbf{X}_k^T(n)\mathbf{A}_k(n)\mathbf{X}_k(n)\mathbf{B}_k(n))^{-1}\mathbf{B}_k^T(n)\mathbf{e}_k(n). \tag{21}$$

The matrices $\mathbf{A}_k(n)$ and $\mathbf{B}_k(n)$ which are introduced in (21) are used for selecting the blocks of weight coefficients to update and the input regressors at every iteration for each node, respectively. It is clear that from (21), the dSPU-DS-APA can also be established when the matrix $\mathbf{B}_k(n)$ is defined according to dDS-APA.

5. Performance analysis of the family of dAPAs

The performance analysis approach of the dAPAs is based on the spatial-temporal energy conservation relation in [4] which was originally derived for single adaptive filter in [15]. In the mean-square performance analysis of dAPAs, we need to study the space–time evolution of the $E\{\|\tilde{\mathbf{h}}_k(n)\|^2_{\mathbf{\Sigma}_k}\}$ at each node, where $\mathbf{\Sigma}_k$ is any Hermitian and positive-definite matrix[4], and $\tilde{\mathbf{h}}_k(n)$ is the weight-error vector which is defined as $\tilde{\mathbf{h}}_k(n)=\mathbf{h}_t-\mathbf{h}_k(n)$. The weight-error vector update equation for the family of dAPAs is introduced as

$$\tilde{\mathbf{h}}_k(n)=\tilde{\mathbf{h}}_{k-1}(n)-\mu_k\mathbf{C}_k(n)\mathbf{X}_k(n)\mathbf{Y}_k(n)\mathbf{e}_k(n). \tag{22}$$

From the linear data model, the error vector $\mathbf{e}_k(n)$ can be represented as

$$\mathbf{e}_k(n)=\mathbf{X}_k^T(n)\tilde{\mathbf{h}}_{k-1}(n)+\mathbf{v}_k(n). \tag{23}$$

By substituting (23) in (22), and taking the $\mathbf{\Sigma}_k$-weighted norm from both sides of (22), we obtain

$$\|\tilde{\mathbf{h}}_k(n)\|^2_{\mathbf{\Sigma}_k}=\|\tilde{\mathbf{h}}_{k-1}(n)\|^2_{\mathbf{\Sigma}'_k}+\mu_k^2\mathbf{v}_k^T(n)\mathbf{U}_k(n)\mathbf{v}_k(n)+\{\text{Cross terms involving one instance of }\mathbf{v}_k(n)\}, \tag{24}$$

where

$$\mathbf{\Sigma}'_k=\mathbf{\Sigma}_k-\mu_k\mathbf{\Sigma}_k\mathbf{Z}_k(n)-\mu\mathbf{Z}_k^T(n)\mathbf{\Sigma}_k+\mu_k^2\mathbf{Z}_k^T(n)\mathbf{\Sigma}_k\mathbf{Z}_k(n) \tag{25}$$

with $\mathbf{Z}_k(n)=\mathbf{C}_k(n)\mathbf{X}_k(n)\mathbf{Y}_k(n)\mathbf{X}_k^T(n)$, and $\mathbf{U}_k(n)=\mathbf{Y}_k^T(n)\mathbf{X}_k^T(n)\mathbf{C}_k^T(n)\mathbf{\Sigma}_k\mathbf{C}_k(n)\mathbf{X}_k(n)\mathbf{Y}_k(n)$. By taking the expectation from both sides of (24)

$$E\{\|\tilde{\mathbf{h}}_k(n)\|^2_{\mathbf{\Sigma}_k}\}=E\{\|\tilde{\mathbf{h}}_{k-1}(n)\|^2_{\mathbf{\Sigma}'_k}\}+\mu_k^2E\{\mathbf{v}_k^T(n)\mathbf{U}_k(n)\mathbf{v}_k(n)\}, \tag{26}$$

we obtain the space–time evolution of the weight-error variance. To simplify (26), we need to use the following independence assumptions [4,15]:

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

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

$$\mathbf{\Sigma}'_k=\mathbf{\Sigma}_k-\mu_k\mathbf{\Sigma}_kE\{\mathbf{Z}_k(n)\}-\mu_kE\{\mathbf{Z}_k^T(n)\}\mathbf{\Sigma}_k+\mu_k^2E\{\mathbf{Z}_k^T(n)\mathbf{\Sigma}_k\mathbf{Z}_k(n)\}. \tag{27}$$

Since $E\{\mathbf{v}_k(n)\mathbf{v}_k^T(n)\}=\sigma^2_{v,k}\mathbf{I}$, Eq. (26) can be stated as

$$E\{\|\tilde{\mathbf{h}}_k(n)\|^2_{\mathbf{\Sigma}_k}\}=E\{\|\tilde{\mathbf{h}}_{k-1}(n)\|^2_{\mathbf{\Sigma}'_k}\}+\mu_k^2\sigma^2_{v,k}\mathrm{Tr}(E\{\mathbf{U}_k(n)\}). \tag{28}$$

By applying the vec(.) operator on both sides of (27), and using $\mathrm{vec}(\mathbf{P}\mathbf{\Sigma}\mathbf{Q})=(\mathbf{Q}^T\otimes\mathbf{P})\mathrm{vec}(\mathbf{\Sigma})$, we obtain

$$\boldsymbol{\sigma}'_k=\boldsymbol{\sigma}_k-\mu_k(E\{\mathbf{Z}_k^T(n)\}\otimes\mathbf{I})\cdot\boldsymbol{\sigma}_k-\mu_k(\mathbf{I}\otimes E\{\mathbf{Z}_k^T(n)\})\cdot\boldsymbol{\sigma}_k+\mu_k^2(E\{\mathbf{Z}_k^T(n)\otimes\mathbf{Z}_k^T(n)\})\cdot\boldsymbol{\sigma}_k, \tag{29}$$

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

$$\mathbf{G}_k=\mathbf{I}-\mu_k(E\{\mathbf{Z}_k^T(n)\}\otimes\mathbf{I})-\mu_k(\mathbf{I}\otimes E\{\mathbf{Z}_k^T(n)\})+\mu_k^2(E\{\mathbf{Z}_k^T(n)\otimes\mathbf{Z}_k^T(n)\}), \tag{30}$$

Eq. (29) can be represented as

$$\boldsymbol{\sigma}'_k=\mathbf{G}_k\cdot\boldsymbol{\sigma}_k. \tag{31}$$

Also, by defining $\boldsymbol{\gamma}_k$ through

$$\boldsymbol{\gamma}_k=\mathrm{vec}(E\{\mathbf{C}_k(n)\mathbf{X}_k(n)\mathbf{Y}_k(n)\mathbf{Y}_k^T(n)\mathbf{X}_k^T(n)\mathbf{C}_k^T(n)\}), \tag{32}$$

the second term in right hand side of (28) is given by

$$\mathrm{Tr}(E\{\mathbf{U}_k(n)\})=\boldsymbol{\gamma}_k^T\cdot\boldsymbol{\sigma}_k. \tag{33}$$

With the above considerations, the recursion of (28) can now be stated as

$$E\{\|\tilde{\mathbf{h}}_k(n)\|^2_{\boldsymbol{\sigma}_k}\}=E\{\|\tilde{\mathbf{h}}_{k-1}(n)\|^2_{\mathbf{G}_k\boldsymbol{\sigma}_k}\}+\mu_k^2\sigma^2_{v,k}\boldsymbol{\gamma}_k^T\boldsymbol{\sigma}_k. \tag{34}$$

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 [4], and adopting the linear relation as $\boldsymbol{\sigma}_{k-1}=\mathbf{G}_k\boldsymbol{\sigma}_k$, the following equation is introduced by

$$E\{\|\tilde{\mathbf{h}}_k(n)\|^2_{\boldsymbol{\sigma}_k}\}=E\{\|\tilde{\mathbf{h}}_{k-1}(n)\|^2_{\boldsymbol{\sigma}_{k-1}}\}+g_k\boldsymbol{\sigma}_k, \tag{35}$$

where $g_k=\mu_k^2\sigma^2_{v,k}\boldsymbol{\gamma}_k^T$. Now by applying the iterating procedure in (35), we can find the relation $E\{\|\tilde{\mathbf{h}}_{k-1}(n)\|^2_{\boldsymbol{\sigma}_{k-1}}\}$ at node k as

$$E\{\|\tilde{\mathbf{h}}_{k-1}(n)\|^2_{\boldsymbol{\sigma}_{k-1}}\}=E\{\|\tilde{\mathbf{h}}_{k-1}(-1)\|^2_{(\mathit{\Pi}_{k-1,1})^n\boldsymbol{\sigma}_{k-1}}\}+a_{k-1}(\mathbf{I}+\cdots+(\mathit{\Pi}_{k-1,1})^{n-1})\boldsymbol{\sigma}_{k-1} \tag{36}$$

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

where $\Pi_{k-1,l} = \mathbf{G}_{k+l-1}\mathbf{G}_{k+l}\ldots\mathbf{G}_J\mathbf{G}_1\ldots\mathbf{G}_{k-1}$, $l = 1, 2, \ldots, J$, and $a_{k-1} = g_k\Pi_{k-1,2} + g_{k+1}\Pi_{k-1,3} + \cdots + g_{k-2}\Pi_{k-1,J} + g_{k-1}$. From this recursion, we will be able to evaluate the transient performance of the family of dAPAs. When $\boldsymbol{\sigma}_{k-1} = \text{vec}(\mathbf{I})$, the theoretical transient performance of the MSD of node k is established. For $\boldsymbol{\sigma}_{k-1} = \text{vec}(\mathbf{R}_k)$, the theoretical EMSE curve is obtained. Also, when n goes to infinity, we obtain

$$E\{\|\tilde{\mathbf{h}}_k(\infty)\|^2_{\boldsymbol{\sigma}_k}\} = E\{\|\tilde{\mathbf{h}}_{k-1}(\infty)\|^2_{\mathbf{G}_k\boldsymbol{\sigma}_k}\} + \mu_k^2\sigma_{v,k}^2\boldsymbol{\gamma}_k^T\boldsymbol{\sigma}_k. \tag{37}$$

Now, by iterating (37) for an incremental network, and using the definitions for $\Pi_{k-1,1}$, and a_{k-1}, the following expression for steady state behavior of dAPAs is established

$$E\{\|\tilde{\mathbf{h}}_{k-1}(\infty)\|^2_{(\mathbf{I}-\Pi_{k-1,1})\boldsymbol{\sigma}_{k-1}}\} = a_{k-1}\boldsymbol{\sigma}_{k-1}. \tag{38}$$

If $(\mathbf{I}-\Pi_{k-1,1})\boldsymbol{\sigma}_{k-1} = \text{vec}(\mathbf{I})$ and $(\mathbf{I}-\Pi_{k-1,1})\boldsymbol{\sigma}_{k-1} = \text{vec}(\mathbf{R}_k)$, the steady-state MSD, and EMSE expressions at node k are established, respectively. Doing this, the final results are

$$\text{MSD}_k = a_{k-1}(\mathbf{I}-\Pi_{k-1,1})^{-1}\text{vec}(\mathbf{I}), \tag{39}$$

$$\text{EMSE}_k = a_{k-1}(\mathbf{I}-\Pi_{k-1,1})^{-1}\text{vec}(\mathbf{R}_k). \tag{40}$$

Also from (23), we obtain that the steady-state MSE at node k is given by

$$\text{MSE}_k = \text{EMSE}_k + \sigma_{v,k}^2. \tag{41}$$

Based on Section 3 and from Table 1, we highlight again that the presented theoretical expressions can be used to predict the performance of classical distributed adaptive algorithms such as dLMS, dNLMS, dDR-LMS, dNDR-LMS, dAPA, dRLS, and dTDAA.

6. Mean and mean-square stability of the family of dAPAs

By substituting (23) in (22), and taking the expectation from both sides, we obtain

$$E\{\tilde{\mathbf{h}}_k(n)\} = [\mathbf{I} - \mu_k E\{\mathbf{Z}_k(n)\}]E\{\tilde{\mathbf{h}}_{k-1}(n)\}. \tag{42}$$

From (42), the convergence in the mean of dAPAs is guaranteed for any μ_k that satisfies

$$\mu_k < \frac{2}{\lambda_{\max}(E\{\mathbf{Z}_k(n)\})}. \tag{43}$$

The general recursion (Eq. (34)), is stable if the matrix $\mathbf{G}_k$ is stable. From (30), 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 [15], the condition on μ_k to guarantee the convergence in the mean-square sense of dAPAs is obtained from (44),

$$0 < \mu_k < \min\left\{\frac{1}{\lambda_{\max}(\mathbf{M}_k^{-1}\mathbf{N}_k)}, \frac{1}{\lambda_{\max}(\mathbf{H}_k)\in\Re^+}\right\}, \tag{44}$$

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

7. Computational complexity

Table 3 shows the number of multiplications and comparisons of various dAPAs for each node at every iteration. The computational complexity of dNLMS and dAPA is obtained from [16,4]. For dAPA, the complexity depends on P (the number of recent regressors) and M. The dSPU-NLMS and dSPU-APA need $3NL+1$ and $(P^2+2P)NL + P^3 + P^2 + 1$ multiplications. Especially for large values of M, the number of multiplications reduces significantly. Selecting the taps of the weight vector to be updated needs some computational costs. But it is less than the computational savings achieved in using partial update of the weight vector. Both algorithms need $K\log_2 N + O(K)$ comparisons when using the heapsort algorithm [12,17].

In the case of dSR-APA, the number of multiplications is $(Q^2+2Q)M + Q^3 + Q^2 + (P-Q)M + P + 1$. This algorithm needs also $P\log_2 Q + O(P)$ comparisons. The number of multiplications and comparisons in dDS-APA are $(Q^2(n)+2Q(n))M + Q^3(n) + Q^2(n)$ and P, respectively [11]. By combining SPU with DS and SR approach, the number of multiplications will be reduced. The dSPU-SR-APA needs $(Q^2+2Q)NL + Q^3 + Q^2 + (P-Q)M + P + 1$ multiplications and $K\log_2 N + O(K) + P\log_2 Q + O(P)$ comparisons [12]. These values reduce to $(Q^2(n)+2Q(n))NL + Q^3(n) + Q^2(n)$ multiplications and $K\log_2 N + O(K) + P$ comparisons in dSPU-DS-APA. In the following section, several simulations to evaluate the computational complexity of dAPAs are presented.

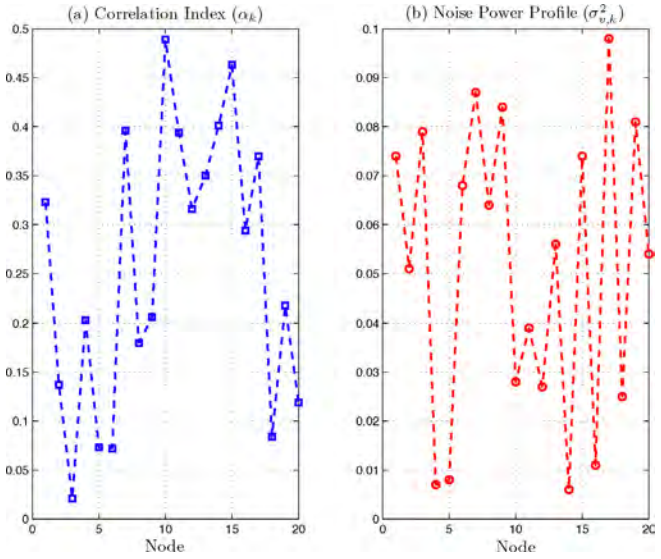

Fig. 2. Node profile. (a) Correlation index for the Gaussian data network. (b) Noise power for the Gaussian data network.

8. Simulation results

We justified the performance of dAPAs in an incremental network with $J = 20$ 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 a first order autoregressive model filter with the z-transfer function $\sqrt{1-\alpha_k^2}/(1-\alpha_k z^{-1})$ 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. 2 shows the node profiles of $\sigma_{v,k}^2$ and α_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 learning curves are obtained by ensemble averaging over 200 independent trials. Also, the steady-state MSE values are obtained by averaging over 1000 steady-state samples from 100 independent realizations for each value of μ for a given algorithm. The step-sizes were chosen to get approximately the same steady-state MSE in all learning curves.

8.1. Simulation results for stability bounds

In the first experiment, the stability bounds of different dAPAs were demonstrated based on (43) and (44). Table 4 shows the

Table 3
Comparison of the estimated complexity per iteration per node for various algorithms for the case of real valued data.

Algorithm	Multiplications	Comparisons
dNLMS	$3M+1$	–
dSPU-NLMS	$3NL+1$	$K\log_2 N+O(K)$
dAPA	$(P^2+2P)M+P^3+P^2$	–
dSPU-APA	$(P^2+2P)NL+P^3+P^2+1$	$K\log_2 N+O(K)$
dSR-APA	$(Q^2+2Q)M+Q^3+Q^2+(P-Q)M+P+1$	$P\log_2 Q+O(P)$
dDS-APA	$(Q^2(n)+2Q(n))M+Q^3(n)+Q^2(n)$	P
dSPU-SR-APA	$(Q^2+2Q)NL+Q^3+Q^2+(P-Q)M+P+1$	$K\log_2 N+O(K)+P\log_2 Q+O(P)$
dSPU-DS-APA	$(Q^2(n)+2Q(n))NL+Q^3(n)+Q^2(n)$	$K\log_2 N+O(K)+P$

Table 4
Stability bounds of the step-size for various dAPAs 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	12.31	2.00	6.56	2.00
dSPU-NLMS ($N=3$)	12.32	1.81	6.45	1.81
dSPU-NLMS ($N=2$)	12.63	1.46	6.48	1.46
dSPU-NLMS ($N=1$)	11.39	0.89	6.00	0.89
dAPA	6.70	2.00	3.91	2.00
dSPU-APA ($N=3$)	6.74	1.62	3.93	1.62
dSPU-APA ($N=2$)	6.70	0.99	3.76	0.99
dSPU-APA ($N=1$)	5.81	0.04	0.58	0.04
dSR-APA ($Q=3$)	8.05	2.00	4.60	2.00
dSR-APA ($Q=2$)	10.56	2.00	5.93	2.00
dSR-APA ($Q=1$)	18.51	2.00	9.33	2.00
dDS-APA	13.01	2.00	7.66	2.00
dSPU-SR-APA ($N=3$, $Q=3$)	8.02	1.67	4.60	1.67
dSPU-SR-APA ($N=3$, $Q=2$)	10.82	1.70	5.97	1.70
dSPU-SR-APA ($N=2$, $Q=3$)	10.43	1.16	5.85	1.16
dSPU-SR-APA ($N=2$, $Q=2$)	10.62	1.22	5.85	1.22
dSPU-DS-APA ($N=3$)	11.40	1.68	6.94	1.68
dSPU-DS-APA ($N=2$)	11.35	1.16	6.74	1.16

values of μ_{max} in dAPAs for various parameters at node 10. For dAPAs, the parameter P was set to 4. In dSPU-NLMS and dSPU-APA, the number of blocks (K) was set to 4 and different values for N were chosen. Also, the parameter Q was set to 1, 2, and 3 in dSR-APA. This table shows that for dSPU-NLMS, by increasing the parameter N, the stability bounds will be increased. This fact can be seen for dSPU-APA. By changing the parameter Q, the stability bounds of dSR-APA will not change. Furthermore, the stability bounds of dAPA and dDS-APA will be the same. In this table, we have also presented the stability bounds for dSPU-SR-APA and dSPU-DS-APA. In these algorithms, by increasing the parameter N, the parameter μ_{max} will be increased. To justify the theoretical values of Table 4, we presented the simulated steady-state MSE values versus the step-size for different dAPAs in Figs. 3–6. The step-sizes change from 0.008 to μ_{max}. Fig. 3 shows the results for dSPU-NLMS algorithm in node 10. Different values for N were chosen. This figure shows that the theoretical values from Table 4 are good estimation for μ_{max}. Fig. 4 presents the steady-state MSE values versus the step-size for dSPU-APA and dSPU-DS-APA. This figure shows that dSPU-APA and dSPU-DS-APA have close stability bounds for both $N=2$ and 3. Fig. 5 compares the stability bounds of dAPA, dDS-APA, and dSR-APA with $Q=2$, and 3 in node 10. This figure shows that the stability bounds of these algorithms are the same and the results from Table 4 are good estimation for μ_{max}. The steady-state MSE versus the step-size for dSPU-SR-APA with different values of Q and N were presented in Fig. 6. By increasing the parameter N, the stability bounds will be increased and good estimations for μ_{max} are observed from Table 4.

8.2. Simulation results for transient performance

Figs. 7–12 present the simulated and theoretical MSE learning curves for the family of dAPAs. The number of blocks (K) was set to 4. Fig. 7 shows the learning curves for dSPU-NLMS in node 10. This figure describes that by increasing the parameter N, the performance of dSPU-NLMS will be close to dNLMS especially for $N=3$. Furthermore, the computational complexity of dSPU-NLMS is lower than dNLMS due to SPU approach. Fig. 8 compares the performance of dAPA and dSPU-APA with different values for N. The convergence speed of dSPU-APA with $N=2$ and 3 is close to dAPA with low computational complexity feature. Also good agreement between theoretical and simulated learning curves can be seen in both figures.

Fig. 9 presents the theoretical and simulated MSE learning curves of dAPA and dDS-APA. Different values for P have been

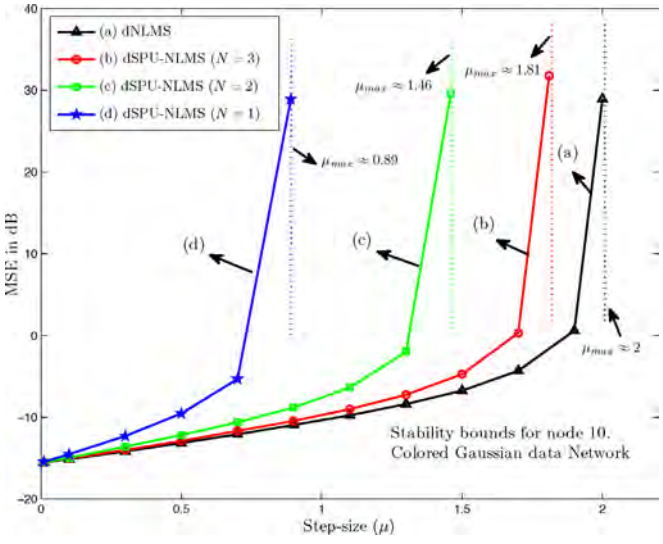

Fig. 3. Simulated MSE curves of dNLMS and dSPU-NLMS as a function of the step-size at node 10 ($N=1, 2, 3$).

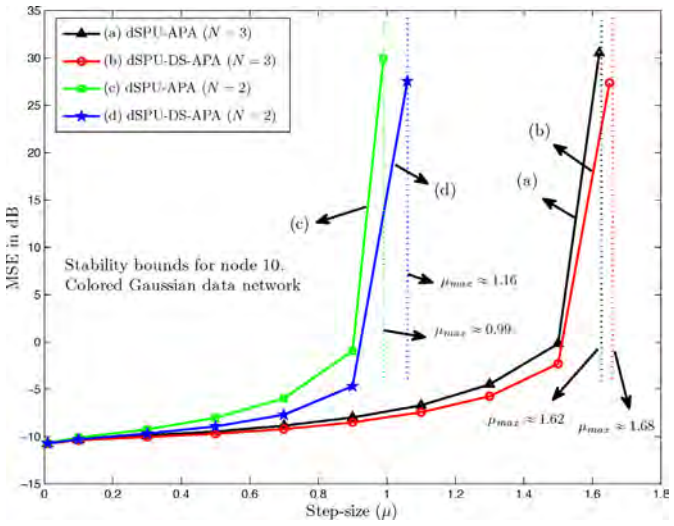

Fig. 4. Simulated MSE curves of dSPU-APA and dSPU-DS-APA as a function of the step-size at node 10 ($N = 2, 3$).

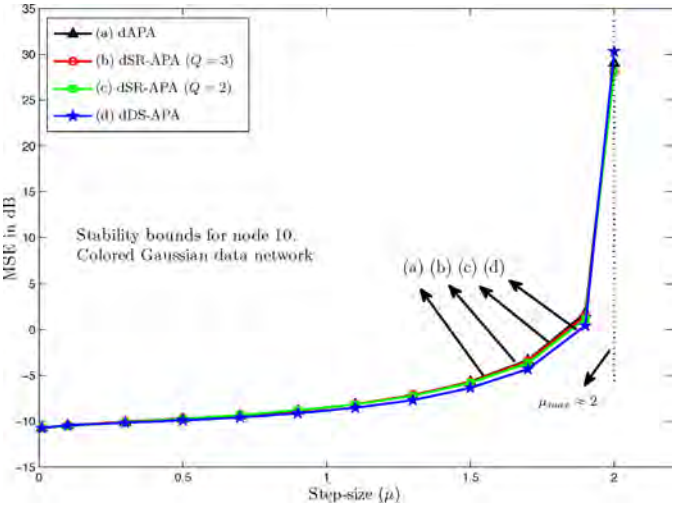

Fig. 5. Simulated MSE curves of dAPA, dDS-APA, and dSR-APA as a function of the step-size at node 10 ($Q = 2, 3$).

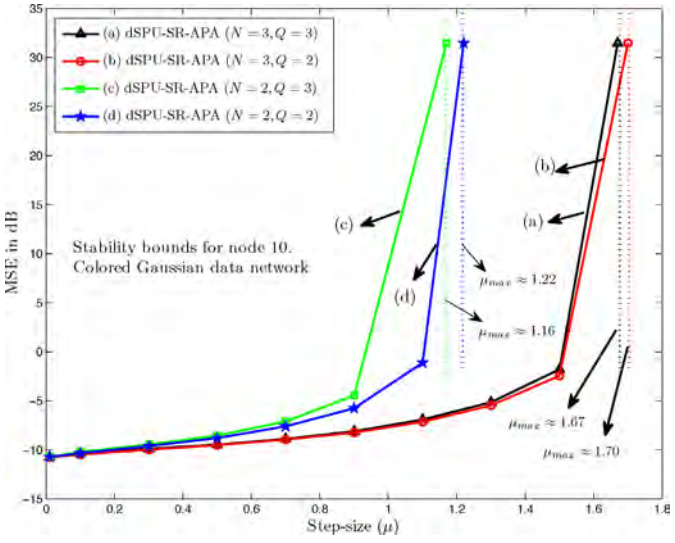

Fig. 6. Simulated MSE curves of dSPU-SR-APA as a function of the step-size at node 10 ($N = 2, 3$ and $Q = 2, 3$).

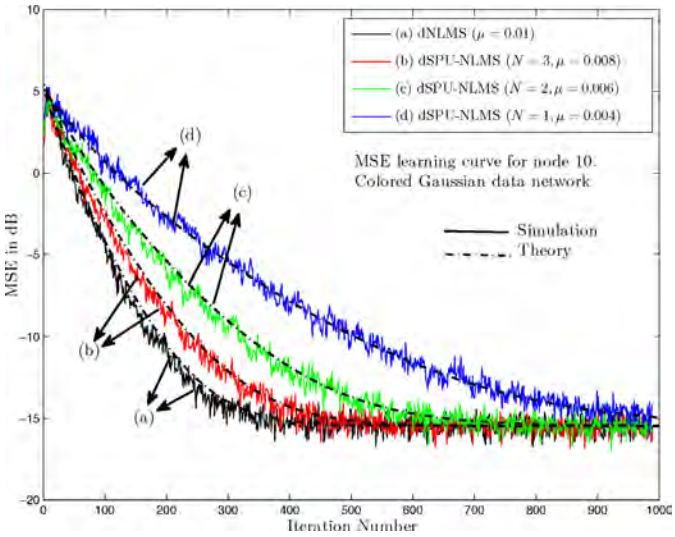

Fig. 7. Simulated and theoretical learning MSE curves of dNLMS and dSPU-NLMS with $N = 1, 2, 3$ for node 10.

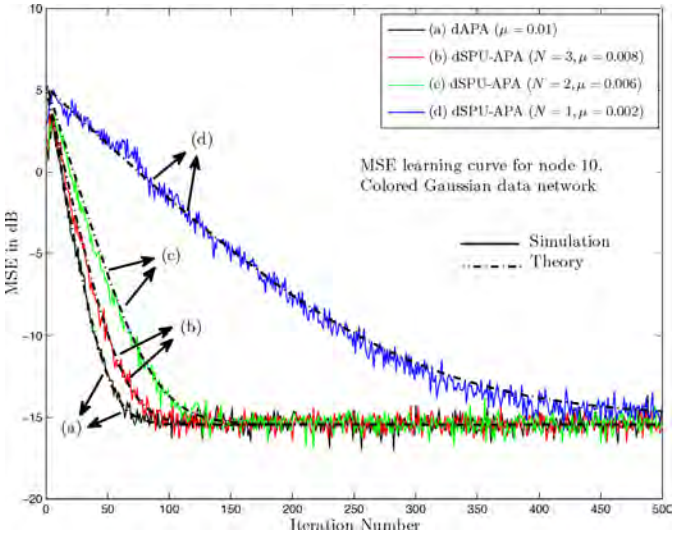

Fig. 8. Simulated and theoretical learning MSE curves of dAPA and dSPU-APA with $N = 1, 2, 3$ for node 10.

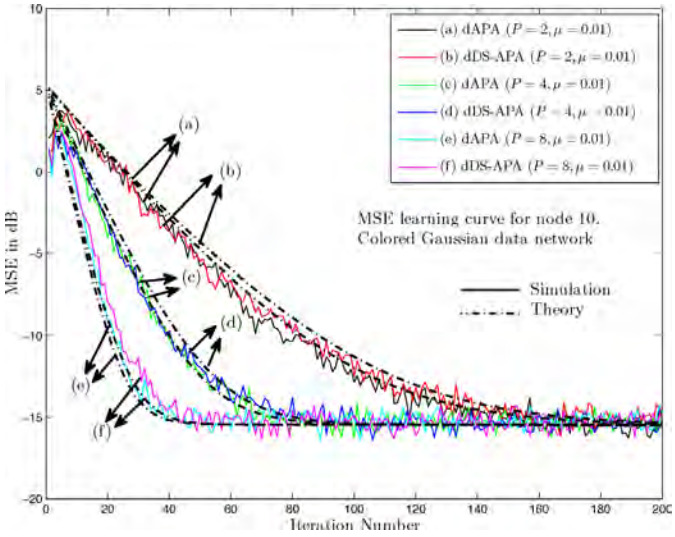

Fig. 9. Simulated and theoretical learning MSE curves of dAPA and dDS-APA with $P = 2, 4, 8$ for node 10.

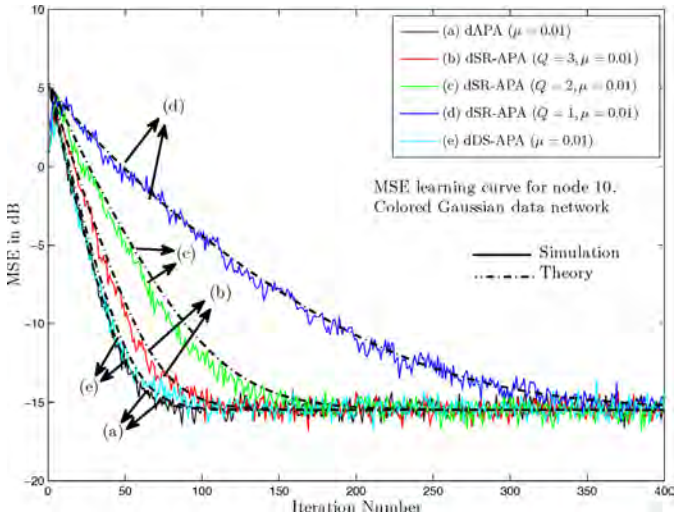

Fig. 10. Simulated and theoretical learning MSE curves of dAPA and dSR-APA with $Q = 1, 2, 3$ for node 10.

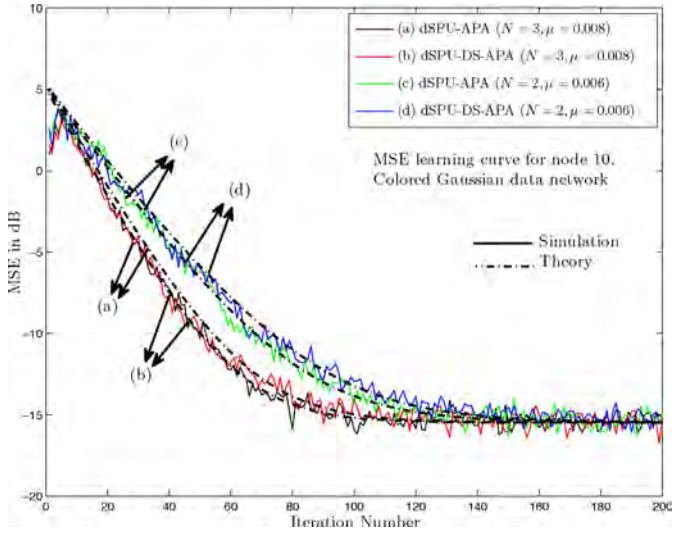

Fig. 11. Simulated and theoretical learning MSE curves of dSPU-APA and dSPU-DS-APA with $N = 2, 3$ for node 10.

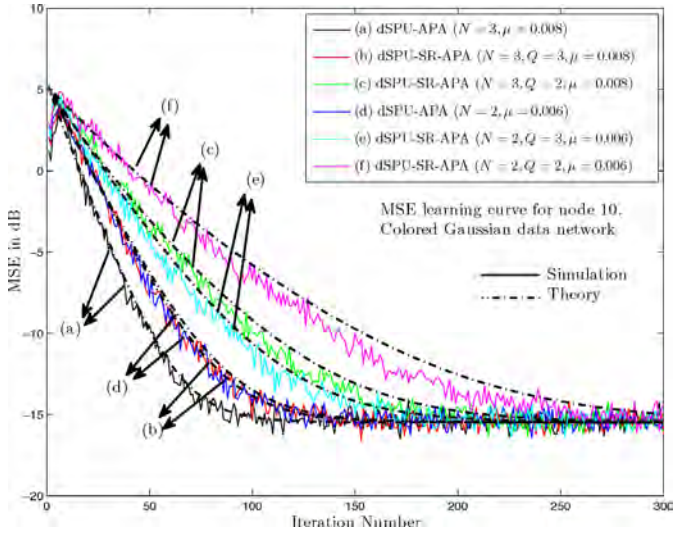

Fig. 12. Simulated and theoretical learning MSE curves of dSPU-APA and dSPU-SR-APA with $N = 2, 3$ and $Q = 2, 3$ for node 10.

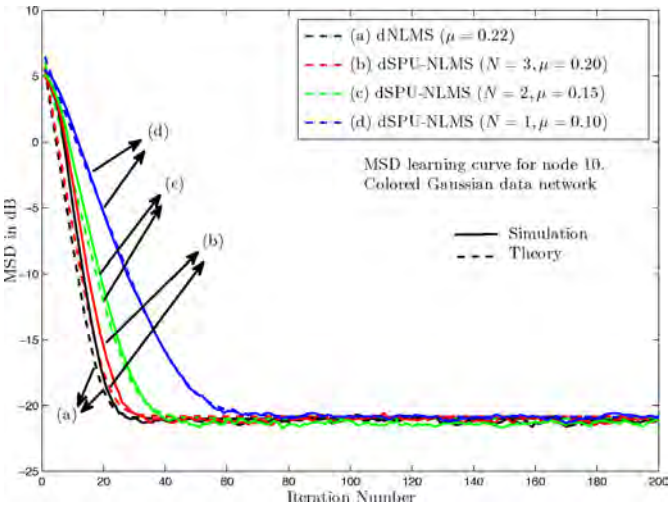

Fig. 13. Simulated and theoretical learning MSD curves of dNLMS and dSPU-NLMS with $N = 1, 2$ and 3 for node 10.

selected. This figure shows that the performances of dDS-APA are close to dAPA. Due to DS approach, the computational load of dDS-APA is lower than dAPA. Fig. 10 compares the performance of dAPA, dDS-APA, and dSR-APA. The parameter P was set to 4 and different values for Q were chosen. By increasing the parameter Q, the convergence speed of dSR-APA will be close to dAPA and dDS-APA especially for $Q = 3$. The SR approach leads to reduction in computational complexity in dSR-APA. Again, good agreement between simulated and theoretical MSE learning curves is observed.

In Fig. 11, the simulated and theoretical MSE learning curves of dSPU-APA and dSPU-DS-APA have been presented. Various values for N were selected. By increasing the parameter N, the convergence speed increases. Also, the performance of dSPU-DS-APA and dSPU-APA are close together. Combining SPU and DS features in dSPU-DS-APA lead to lower computational complexity than dSPU-APA. Fig. 12 compares the performance of dSPU-APA and dSPU-SR-APA. The parameters Q and N were set to 2 and 3. The performance of dSPU-APA with $N = 3$ is better than other distributed adaptive algorithms. The convergence speed of dSPU-APA with $N = 2$ and dSPU-SR-APA with $Q = 3$ and $N = 3$ are close. The suitable agreement between simulated and theoretical learning curves can be seen in all curves. In Fig. 13, the theoretical and simulated MSD learning curves for dNLMS and dSPU-NLMS algorithms were presented. As we can see, for various values of N, good agreement is observed.

8.3. Simulation results for steady-state performance

Fig. 14 shows the simulated and theoretical steady-state MSE values of dSPU-NLMS for each node with $\mu_k = 0.5$. The theoretical values are obtained from (41). The curves have been obtained for various values of N. The steady-state EMSE for dSPU-NLMS with high values of N is lower than dSPU-NLMS with low values of N. For $N = 3$, the steady-state MSE values are very close to dNLMS algorithm. Also, the good agreement between simulated and theoretical MSE values is observed. The results for dAPA, dDS-APA, and dSR-APA with $Q = 2$ and 3 have been shown in Fig. 15. This figure shows that dDS-APA and dSR-APA have close steady-state MSE to dAPA. Furthermore, the computational complexity of dDS-APA and dSR-APA is lower than dAPA. Again good agreement between simulated and theoretical values is observed. Fig. 16 presents the results for dSPU-APA and dSPU-DS-APA with $N = 2$ and 3. The dSPU-APA with $N = 3$ has lower steady-state MSE than dSPU-APA with $N = 2$. This figure shows that dSPU-DS-APA has close performance to

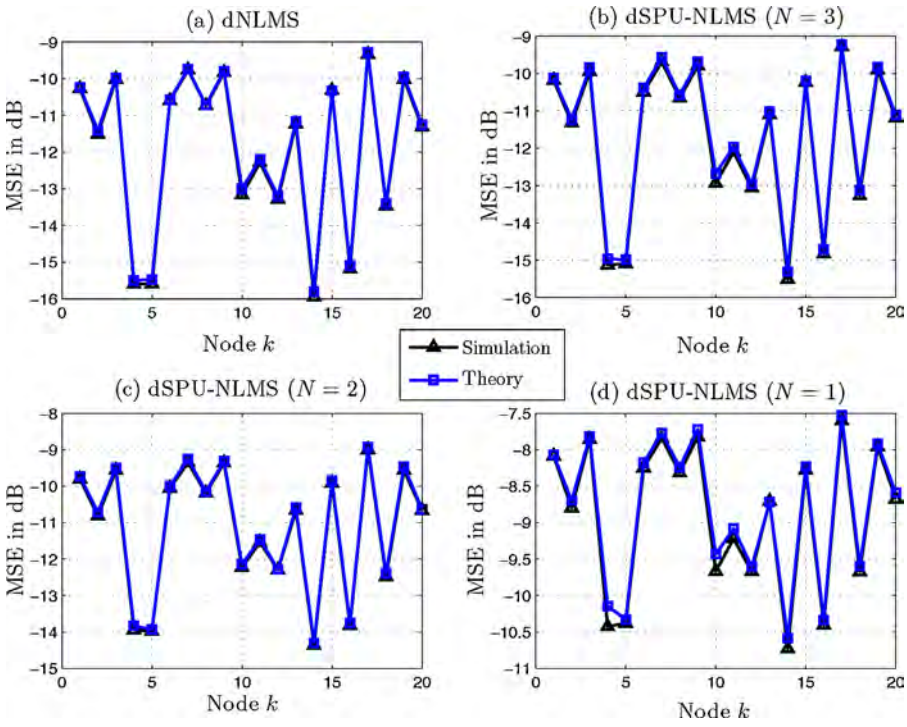

Fig. 14. (a–d) Steady-state MSE of all nodes for dNLMS and dSPU-NLMS using $\mu_k = 0.5$ ($N = 1, 2, 3$).

dSPU-APA. Furthermore, the computational complexity of dSPU-DS-APA is lower than dSPU-APA. Also, the agreement between simulated and theoretical values in dSPU-APA is better than dSPU-DS-APA. The steady-state MSE values for each node in dSPU-SR-APA have been presented in Fig. 17. Various values for N and Q have been used. The dSPU-SR-APA with $N = 3$ and $Q = 3$ has close steady-state MSE values to dSPU-SR-APA with $N = 3$ and $Q = 2$. This fact can be seen for $N = 2$. Good agreement between simulated and theoretical steady-state MSE values can be seen in this figure.

Fig. 18 shows the simulated and theoretical steady-state MSE values of dSPU-NLMS as function of the step-size for node 10. Various values for N have been chosen. The step-sizes change in the

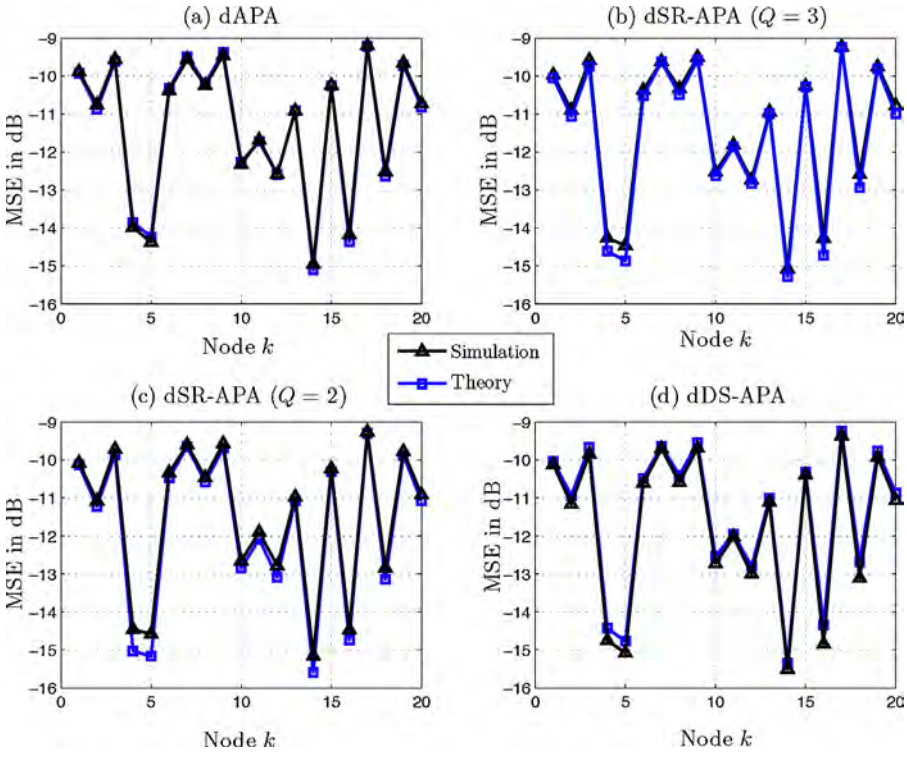

Fig. 15. (a–d) Steady-state MSE of all nodes for dAPA, dSR-APA, and dDS-APA using $\mu_k = 0.5$ ($Q = 2, 3$).

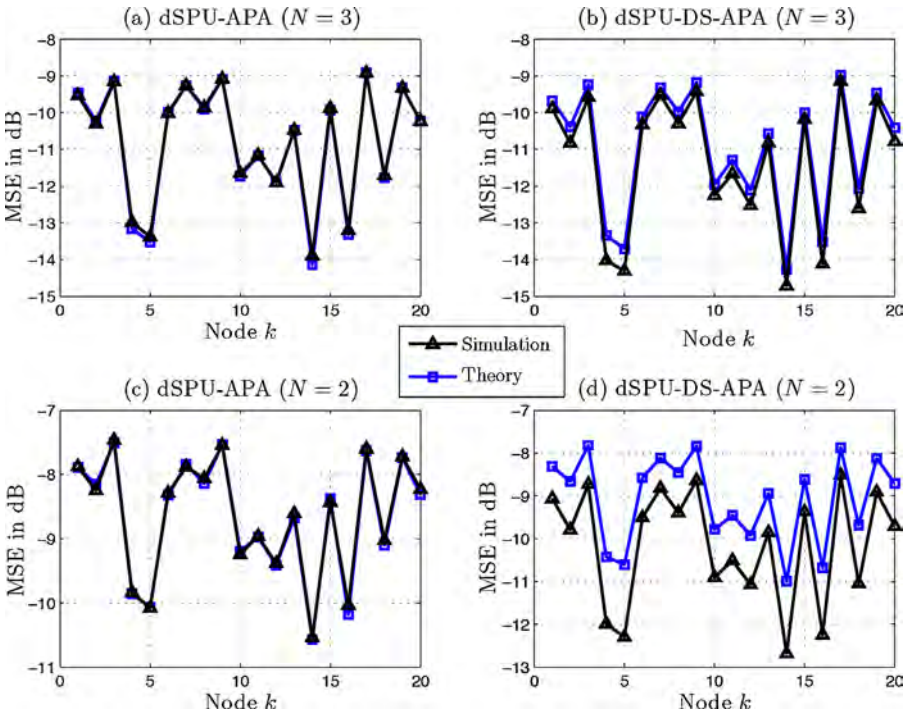

Fig. 16. (a–d) Steady-state MSE of all nodes for dSPU-APA and dSPU-DS-APA using $\mu_k = 0.5$ ($N = 2, 3$).

range [0.008, 0.5] which are in the stability bound of dSPU-NLMS. By increasing the parameter N, the steady-state MSE values of dSPU-NLMS are close to dNLMS algorithm. The theoretical values are in good agreement with simulated values. Fig. 19 presents the steady-state MSE values versus the step-size for dAPA, dSR-APA, and dDS-APA. The steady-state MSE values in these algorithms are close together and suitable agreement can be seen between simulated and theoretical steady-state MSE. In Fig. 20, we presented the results for dSPU-APA and dSPU-DS-APA. The steady-state MSE values for dSPU-APA are close to dSPU-DS-APA for both values of N. Fig. 21 shows the steady-state MSE values versus the step-size for dSPU-SR-APA with various values of N and Q. The results for $N = 3$ and $N = 2$ are close together and again suitable agreement between simulated and theoretical values is observed.

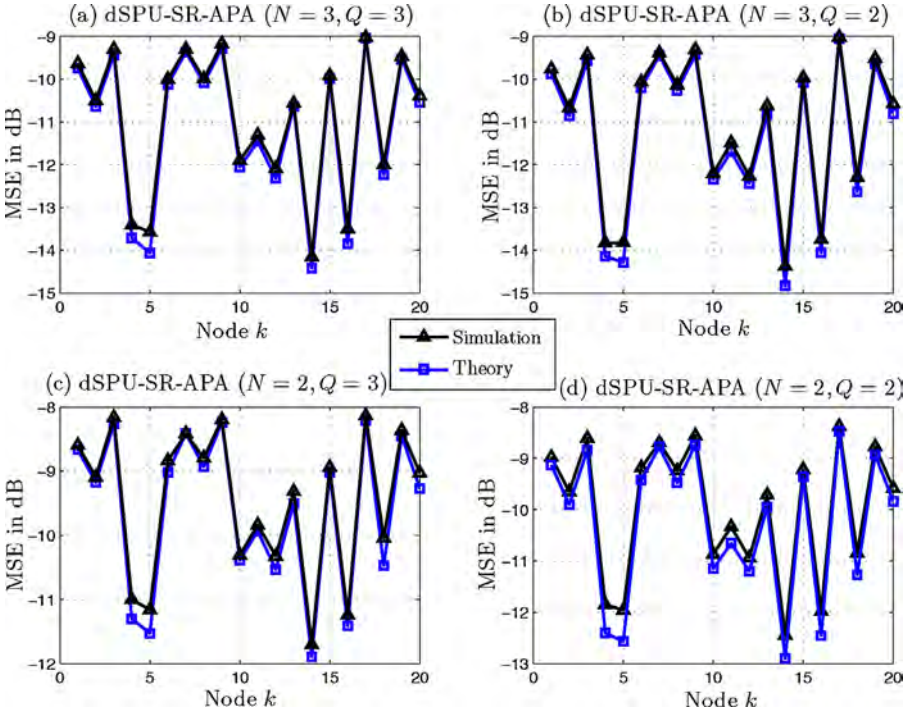

Fig. 17. (a–d) Steady-state MSE of all nodes for dSPU-SR-APA using $\mu_k = 0.5$ ($N = 2, 3$ and $Q = 2, 3$).

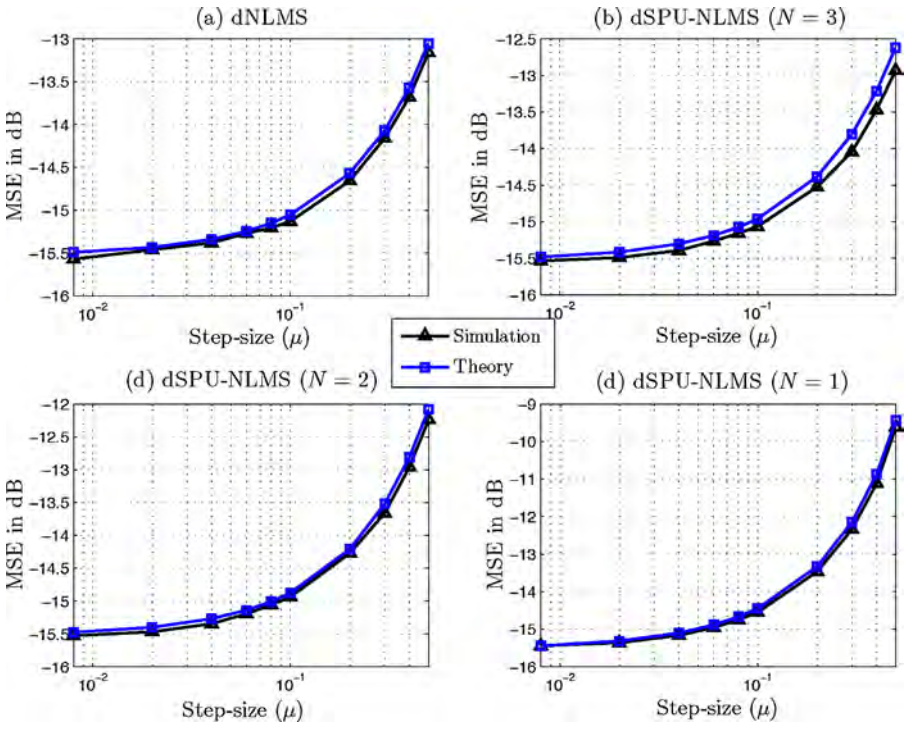

Fig. 18. (a–d) Simulated and theoretical steady-state MSE curves of dNLMS and dSPU-NLMS at node 10 as a function of the step-size ($N = 1, 2, 3$).

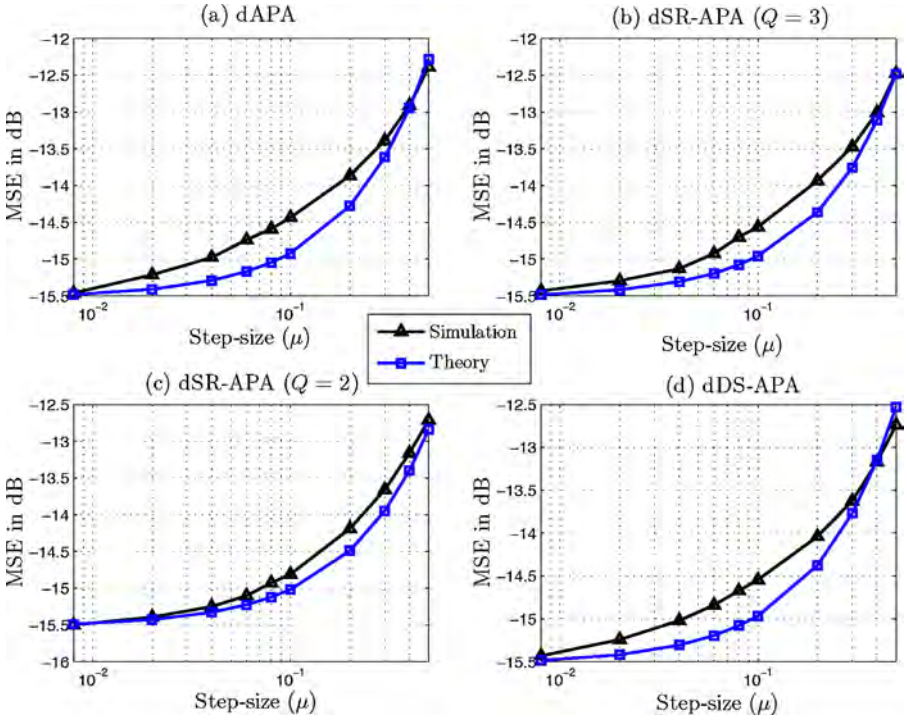

Fig. 19. (a–d) Simulated and theoretical steady-state MSE curves of dAPA, dSR-APA, and dDS-APA at node 10 as a function of the step-size ($Q = 2, 3$).

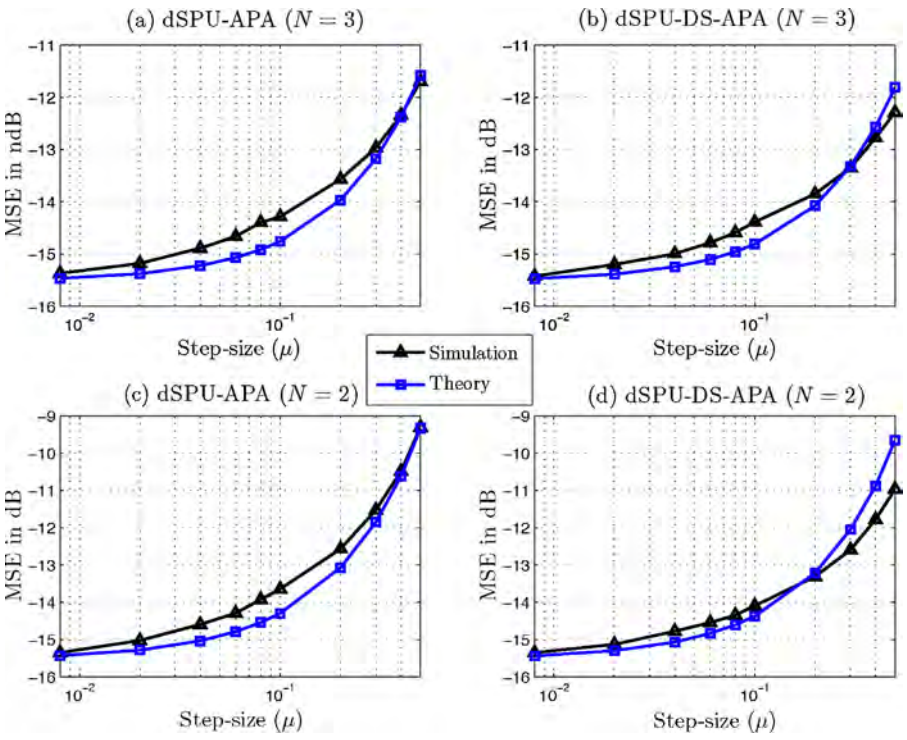

Fig. 20. (a–d) Simulated and theoretical steady-state MSE curves of dSPU-APA and dSPU-DS-APA at node 10 as a function of the step-size ($N = 2, 3$).

8.4. *Simulation results for computational complexity*

Fig. 22 presents the number of selected regressors in single trial and the average number of regressors during the adaptation for node 10 in dDS-APA. As a result, the dDS-APA has a low overall computational complexity because the number of used regressors becomes small. Finally in Fig. 23, the number of multiplications per node versus the filter length (M) for various distributed adaptive algorithms have been presented. As we can see, by increasing the filter length the number of multiplications will increase. The computational complexity of dAPAs is between dNLMS and dAPA.

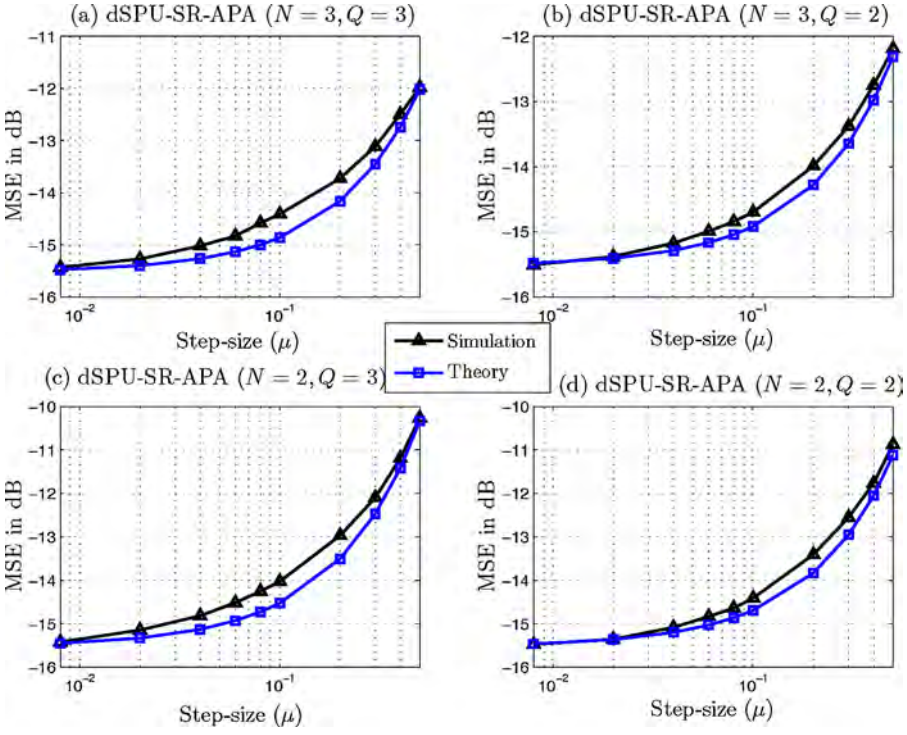

Fig. 21. (a–d) Simulated and theoretical steady-state MSE curves of dSPU-SR-APA at node 10 as a function of the step-size ($N = 2, 3$ and $Q = 2, 3$).

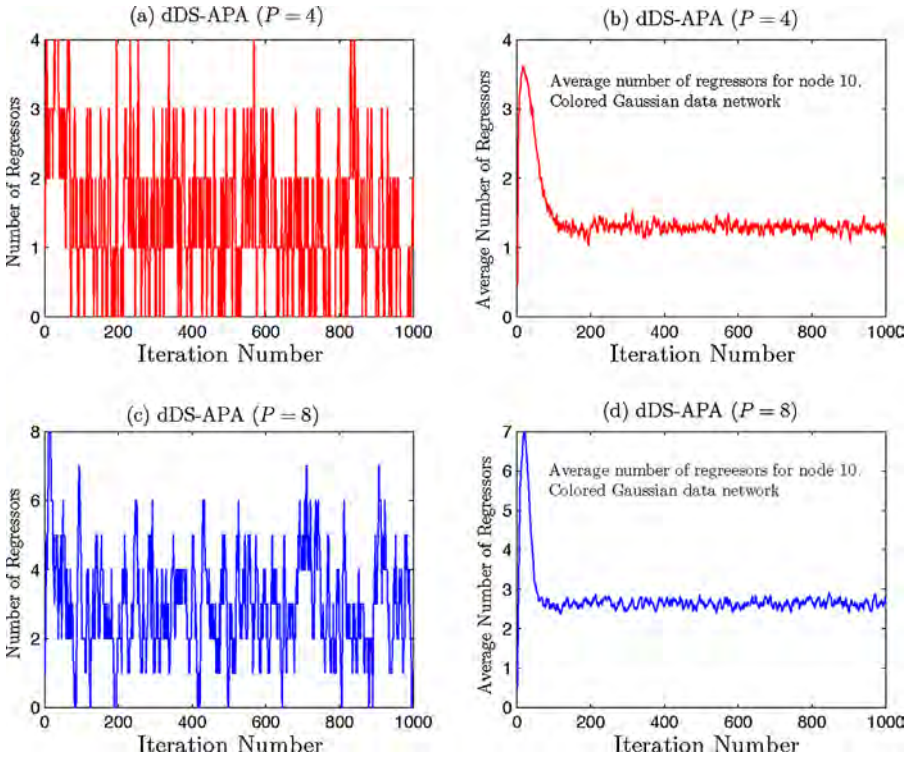

Fig. 22. (a–d) Number of selected regressors over single trial, and the average number of selected regressors in dDS-APA for node 10 ($P = 4$, 8, and $\mu = 0.01$).

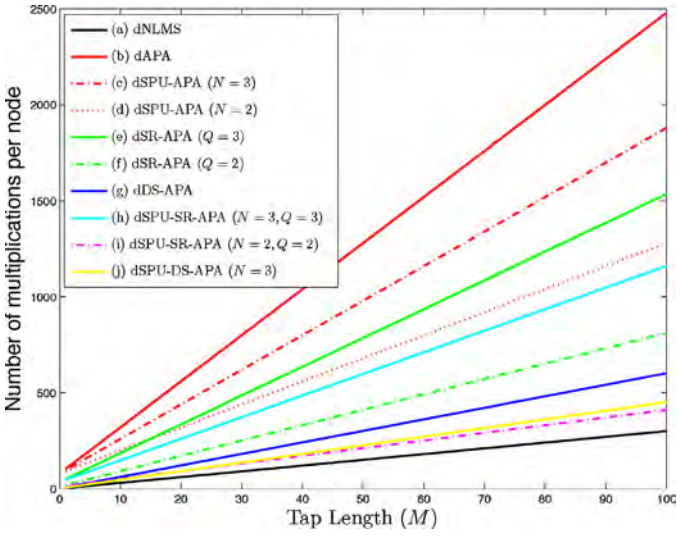

Fig. 23. Number of multiplications per node versus the filter length for the family of dAPAs.

9. Summary and conclusions

In this paper we applied various affine projection adaptive algorithms to the distributed estimation in an incremental network. Based on this, dSPU-NLMS, dSPU-APA, dSR-APA, dDS-APA, dSPU-SR-APA, and dSPU-DS-APA were established. Simulation results show that dAPAs have good convergence speed, low steady-state MSE, and low computational complexity. Also, the theoretical transient and steady-state performance of dAPAs as well as stability bounds were analyzed in a unified way and were confirmed by several computer simulations.

References

[1] Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. IEEE Commun Mag 2002;40:102–14.
[2] Li D, Wong KD, Hu YH, Sayeed AM. Detection classification and tracking of targets. IEEE Signal Process Mag 2002;19:17–29.
[3] Lopes CG, Sayed AH. Incremental adaptive strategies over distributed networks. IEEE Trans Signal Process 2007;55:4064–77.
[4] Li L, Chambers JA, Lopes CG, Sayed AH. Distributed estimation over an adaptive incremental network based on the affine projection algorithm. IEEE Trans Signal Process 2010;58:151–64.
[5] Douglas SC. Analysis and implementation of the max-NLMS adaptive filter. In: Proceedings of the 29th Asilomar Conference on Signals, Systems, and Computers. 1995. p. 659–63.
[6] Aboulnasr T, Mayyas K. Selective coefficient update of gradient-based adaptive algorithms. In: Proceedings of the IEEE International Conference on Acoustics,. Speech, and Signal Processing. 1997. p. 1929–32.
[7] Aboulnasr T, Mayyas K. Complexity reduction of the NLMS algorithm via selective coefficient update. IEEE Trans Signal Process 1999 May;47(5):1421–4.
[8] Doğançay K, Tanrıkulu O. Adaptive filtering algorithms with selective partial updates. IEEE Trans Circuits Syst II: Analog Digital Signal Process 2001 Aug;48(8):762–9.
[9] Werner S, de Campos MLR, Diniz PSR. Partial-update NLMS algorithms with data-selective updating. IEEE Trans Signal Process 2004 Apr;52(4):938–48.
[10] Hwang KY, Song WJ. An affine projection adaptive filtering algorithm with selective regressors. IEEE Trans Circuits Syst II: Express Briefs 2007 Jan;54(1):43–6.
[11] Kong SJ, Hwang KY, Song WJ. An affine projection algorithm with dynamic selection of input vectors. IEEE Signal Process Lett 2007 Aug;14(8):529–32.
[12] Abadi MSE, Mehrdad V. Family of affine projection adaptive filters with selective partial updates and selective regressors. IET Signal Process 2010;4:567–75.
[13] Soni RA, Jenkins WK, Gallivan KA. Acceleration of normalized adaptive filtering data-reusing methods using the Tchebyshev and conjugate gradient methods. In: Proceedings of the International Symposium on Circuits and Systems. 1998. p. 309–12.
[14] Diniz PSR. Adaptive filtering: algorithms and practical implementation. 2nd ed. Kluwer; 2002.
[15] Shin H-C, Sayed AH. Mean-square performance of a family of affine projection algorithms. IEEE Trans Signal Process 2004 Jan;52(1):90–102.
[16] Sayed AH. Fundamentals of adaptive filtering. Wiley; 2003.
[17] Knuth DE. Sorting and searching. vol. 3 of The art of computer programming. 2nd ed. Reading, MA: Addison-Wesley; 1973.