Circuits, Systems, and Signal Processing (2020) 39:4136–4154
https://doi.org/10.1007/s00034-020-01363-4

Correction-Based Diffusion LMS Algorithms for Distributed Estimation

Huining Chang[1] · Wenling Li[1]

Received: 20 March 2019 / Revised: 29 January 2020 / Accepted: 30 January 2020 /
Published online: 10 February 2020

Abstract
In this paper, we consider distributed estimation problems where a set of agents are used for jointly estimating an interesting parameter from the noise measurements. By using the adaptation-then-combination rule in the traditional diffusion least mean square (DLMS), a DLMS is proposed by introducing a correction step with a gain factor between the adaptation and combination steps. An explicit expression for the network mean-square deviation is derived for the proposed algorithm, and a sufficient condition is established to guarantee the mean stability. Simulation results are provided to verify theoretical results, and it is shown that the proposed algorithm outperforms the traditional DLMS.

Keywords Distributed estimation · Correction based · Diffusion LMS · Adaptive networks

1 Introduction

Distributed estimation, that estimates the unknown parameter vector of interest through the noisy observation data of each node, has recently received much attention. Adaptive networks are attractive solutions for distributed estimation problems. An adaptive network consists of a group of nodes with data processing, learning, and data transmission capabilities. These nodes are connected together in a variety of topologies to form an interconnected network. Nodes in the network can interact with their neighbors to exchange information and collaborate to estimate common target parameters. As a research hotspot in distributed networks, distributed estimation has a wide range of

✉ Wenling Li
lwlmath@buaa.edu.cn

Huining Chang
ZY1703102@buaa.edu.cn

1 School of Automation Science and Electrical Engineering, Beihang University (BUAA), Beijing 100191, China

practical applications, including biomedicine, sensory networks, environmental monitoring, target location, smart agriculture, and counter-terrorism [20].

The problem of distributed estimation in adaptive networks has received more and more attention, and a large number of distributed estimation algorithms have been proposed. In addition, there are at least two types of the communication modes between nodes in a distributed network: the consensus strategy [2,17,19,25] and the diffusion strategy [5–7,22,23]. However, it is known that traditional diffusion strategy outperforms traditional consensus strategy [26]. Therefore, we choose the diffusion network structure in this paper. Diffusion networks are sought after by researchers and have been widely applied in cognitive radio, environmental monitoring, and industrial automation. In [24], the diffusion least mean-square (DLMS) algorithm was proposed to improve the estimation performance of the algorithm. The adapt-then-combine (ATC) DLMS algorithm and the combine-then-adapt (CTA) DLMS algorithm were proposed according to the order of fusion and adaptation in [3]. In order to improve the convergence speed of the algorithm, the researchers presented a variable step size DLMS algorithm to accelerate the convergence speed by dynamically changing the step size in real time [18]. Aiming at the problem of noise influence in regression vector, a compensatory diffusion LMS algorithm was studied in [1]. In [9], a DLMS algorithm based on data selection, which enabled nodes to decide whether to spread data according to transmission criteria, was designed to reduce the communication volume of nodes to a certain extent. In [10], the authors considered the influence of malicious nodes on the network under specific attacks and have introduced a DLMS algorithm based on reputation mechanism. The principle is to assign the corresponding reputation value according to the contribution made by the node. In wireless sensor networks, energy saving is an important research content, and the regression vector may be a non-Gaussian distribution. Therefore, a quantized minimum error entropy criterion was proposed in the literature [4], which was only used to transmit error signals between nodes and effectively reduced the amount of information transmission.

It is worth pointing out that improving the estimation performance of the least mean-square algorithm itself has been largely ignored despite their irreplaceable significance in parameter estimation. As a result, we focus on designing an effective strategy to improve the steady-state performance of the algorithm. In [26], the author proposed an exact diffusion strategy with guaranteed exact convergence for deterministic optimization problems. The exact diffusion algorithm has proven to remove the bias that is characteristic of distributed solutions for deterministic optimization problems [26]. In addition, the algorithm was shown to be applicable to a larger set of combination policies than earlier approaches in the literature [26]. The exact diffusion resembles standard diffusion strategy, with the addition of a “correction” step between the adaptation and combination step. Inspired by this paper, we directly add the “correction” step between the adaptation and combination in the DLMS algorithm. Nevertheless, the estimated performance of the algorithm is not improved. Therefore, we have thought of adding a gain factor to the correction step as shown in the third part and propose correction-based diffusion LMS Algorithms. In comparison with [24] which have focused on DLMS algorithms, the correction-based diffusion LMS algorithm performs estimating unknown parameters more efficiently. Simulation results illustrate the theoretical findings and reveal the enhanced learning abilities of the proposed filters.

The structure of this paper is as follows. In Sect. 2, we develop an estimation problem and propose a solution. The algorithm is presented in Sect. 3. In Sect. 4, we analyze the performance of the algorithm. In Sect. 5, we simulate the algorithms and theoretical results.

Notation: In this paper, we adopt normal font letters for scalars, boldface lowercase letters for column vectors, and boldface uppercase letters for matrices. The symbol $(\cdot)^T$ denotes matrix transpose, and the symbol $(\cdot)^{-1}$ denotes matrix inverse. The operators $\mathrm{diag}\{\cdot\}$, $\mathrm{col}\{\cdot\}$, $\mathrm{tr}(\cdot)$, and $\mathbb{E}\{\cdot\}$ denote the (block) diagonal matrix, the column vector, the trace of a matrix, the expectation, respectively. The symbol $\otimes$ denotes the Kronecker product. The symbol $\mathrm{vec}\{\cdot\}$ refers to the standard vectorization operator that stacks the columns of a matrix on top of each other. The notation $\|\cdot\|_2$ denotes the Euclidean norm of a vector, and $|\cdot|$ denotes the (element-wised) L_1-norm of a scalar or vector. We use indexes k and ℓ to denote nodes and use i to denote time. Other notations will be introduced if necessary.

2 Problem Formulation

Consider a network of N wireless sensors, spatially distributed over some region. At every time instant i, each node k can access to a zero-mean observation $d_k(i)$, and a zero-mean $M-$dimensional row regression vector $\boldsymbol{u}_k(i)$. We assume the data to be related via a linear model as follows:

$$d_k(i) = \boldsymbol{u}_k(i)\boldsymbol{w}^o + v_k(i) \tag{1}$$

where $\boldsymbol{w}^o$ is an $M-$dimensional unknown column vector, and $v_k(i)$ is a zero-mean measurement noise of variance $\sigma_{v,k}^2$. We assume covariance matrix $\boldsymbol{\mathcal{R}}_{u,k} = \mathbb{E}\{\boldsymbol{u}_k(i)^T\boldsymbol{u}_k(i)\}$ is positive definite, and let $\boldsymbol{\mathcal{R}}_{\mathrm{d}u,k} = \mathbb{E}\{d_k(i)\boldsymbol{u}_k(i)^T\}$, where $\boldsymbol{\mathcal{R}}_{\mathrm{d}u,k}$ is a cross-correlation vector.

We estimate the unknown parameter $\boldsymbol{w}^o$ by minimizing the following cost function:

$$\boldsymbol{w}^o = \underset{\boldsymbol{w}\in R^M}{\arg\min}\ J^{\mathrm{glob}}(\boldsymbol{w}) = \sum_{k=1}^{N}\mathbb{E}\{|d_k(i) - \boldsymbol{u}_k(i)\boldsymbol{w}|^2\} \tag{2}$$

which is strongly convex, second-order differentiable, and minimized at $\boldsymbol{w}^o$.

The optimal estimator is given by [21]

$$\boldsymbol{w}^o = \left(\sum_{k=1}^{N}\boldsymbol{\mathcal{R}}_{u,k}\right)^{-1}\left(\sum_{k=1}^{N}\boldsymbol{\mathcal{R}}_{\mathrm{d}u,k}\right) \tag{3}$$

It is worth pointing out that centralized networks have the disadvantages of the poor performance in dealing emergencies and in processing the transmitted data in real time [3]. Hence, we focus on designing distributed network topology for estimating unknown parameters.

Note the entry $a_{\ell,k}$ of the matrix $\boldsymbol{A}$, which can represent the topology of a network, satisfies

$$\mathbb{1}^T \boldsymbol{A} = \mathbb{1}^T,\ \boldsymbol{A}\mathbb{1} = \mathbb{1},\ \ a_{\ell,k} = 0 \text{ if } \ell \notin \mathcal{N}_k \tag{4}$$

where $\mathbb{1}$ denotes an $N-$dimensional column vector consisting of all ones. Let $a_{\ell,k}$ denotes the weight that is used to scale the data that flows from node ℓ to k. Let $\boldsymbol{A} \overset{\Delta}{=} [a_{\ell,k}] \in R^{N\times N}$ denotes the matrix that collects all these coefficient.

Now, we consider that it is feasible to replace the global cost with the local costs as follows:

$$J^{\text{glob}}(\boldsymbol{w}) = \sum_{\ell=1}^{N} J_\ell(\boldsymbol{w}) \tag{5}$$

where $J^{\text{glob}}(\boldsymbol{w}) = \mathbb{E}\{|d_\ell(i) - \boldsymbol{u}_\ell(i)\boldsymbol{w}|^2\}$ denotes the cost function in ℓ node.

By using the property of combination coefficients $a_{\ell,k} = a_{k,\ell}$ and $\sum_{k=1}^{N} a_{k,\ell} = 1$, we can rewrite Eq. (5) as follows:

$$J^{\text{glob}}(\boldsymbol{w}) = \sum_{\ell=1}^{N} \left(\sum_{k=1}^{N} a_{k,\ell} \right) J_\ell(\boldsymbol{w}) \tag{6}$$

It only exchanges the order and data with neighbor nodes in k node, so

$$\begin{aligned} J^{\text{glob}}(\boldsymbol{w}) &= \sum_{k=1}^{N} \left(\sum_{\ell \in \mathcal{N}_k} a_{\ell,k} \right) J_\ell(\boldsymbol{w}) \\ &= \sum_{k=1}^{N} J_\ell^{loc}(\boldsymbol{w}) \end{aligned} \tag{7}$$

Thus, when the combination coefficients are symmetric and convex, we will find out that optimizing the global cost in view of individual costs (2) is equivalent to optimizing the global cost in view of local costs (7). Hence, we can represent the global optimization problem in terms of the local optimization problem. In the following papers, we only give the algorithm of each node k.

3 Diffusion Adaptive Solutions

In order to estimate $\boldsymbol{w}^o$ more accurately, the diffusion least mean-square algorithms are proposed, which include the ATC algorithm and the CTA algorithm. Here, we focus on the algorithm using the adapt-then-combine cooperation rule as it shows better performance than the combine-then-adapt cooperation strategy. The ATC diffusion LMS is implemented by [11]

$$\begin{cases} \varphi_{k,i} = \boldsymbol{w}_{k,i-1} + \mu \boldsymbol{u}_{k,i}^T (d_{k,i} - \boldsymbol{u}_{k,i} \boldsymbol{w}_{k,i-1}) & \text{(8a)} \\ \boldsymbol{w}_{k,i} = \sum\limits_{\ell \in \mathcal{N}_k} a_{\ell,k} \varphi_{\ell,i} & \text{(8b)} \end{cases}$$

where μ is a constant step size parameter, $\boldsymbol{w}_{k,i}$ is an estimate at i iteration for each node k, and $\varphi_{k,i}$ is the intermediate variable. The coupling coefficient $a_{\ell,k}$, which is the ℓ and k entity of the matrix $\boldsymbol{A}$ that is a double-stochastic matrix satisfied Eq. (4).

We note that the mean-square error value of DLMS algorithm still needs to be reduced, although it already has good estimation performance. It is of importance to design an effective strategy to improve the performance of the algorithm. Therefore, we hope to design a distributed algorithm to improve steady-state performance of the algorithm. The exact diffusion algorithm has been proposed to remove the bias that is characteristic of distributed solution for deterministic optimization problems, and the algorithm is as follows [26,27]:

$$\begin{cases} \varphi_{k,i} = \boldsymbol{w}_{k,i-1} + \mu \nabla J_k(\boldsymbol{w}_{k,i-1}) & \text{(9a)} \\ \phi_{k,i} = \varphi_{k,i} + \boldsymbol{w}_{k,i-1} - \varphi_{k,i-1} & \text{(9b)} \\ \boldsymbol{w}_{k,i} = \sum\limits_{\ell \in \mathcal{N}_k} \bar{a}_{\ell,k} \phi_{\ell,i} & \text{(9c)} \end{cases}$$

where $\bar{a}_{\ell,k}$ is the ℓ and k entry of the matrix $\bar{A}$ that satisfies $\bar{A} = (I_N + A)/2$. The symbol $\nabla J_k(\boldsymbol{w}_{k,i-1})$ denotes the gradient vector of J_k relative to w, which is a convex and differentiable function for deterministic optimization problems. $\phi_{k,i}$ and $\varphi_{k,i}$ are intermediate variables.

It is observed that the "correction" step has been added to the algorithm between the adaptation and combination step.

We apply this idea of adding "correction" step between combination and adaptation to the DLMS algorithm. Therefore, we design a correction-based diffusion LMS algorithm by adding a gain factor before the difference between $\boldsymbol{w}_{k,i-1}$ and $\varphi_{k,i-1}$ from the previous iteration. The algorithm is as follows:

$$\begin{cases} \varphi_{k,i} = \boldsymbol{w}_{k,i-1} + \mu \boldsymbol{u}_{k,i}^T (d_{k,i} - \boldsymbol{u}_{k,i} \boldsymbol{w}_{k,i-1}) & \text{(10a)} \\ \phi_{k,i} = \varphi_{k,i} + \lambda(\boldsymbol{w}_{k,i-1} - \varphi_{k,i-1}) & \text{(10b)} \\ \boldsymbol{w}_{k,i} = \sum\limits_{\ell \in \mathcal{N}_k} a_{\ell,k} \phi_{\ell,i} & \text{(10c)} \end{cases}$$

where both $\varphi_{k,i}$ and $\phi_{k,i}$ are intermediate variables, λ is a positive gain factor, and $a_{\ell,k}$ is the coupling coefficient.

It is observed that correction-based diffusion LMS algorithm is different from the exact diffusion strategy, with the addition of a gain factor. Another difference is that the exact diffusion is to study deterministic optimization problem, while diffusion LMS is to study parameter estimation problem with noise in measurement equation.

Before proceeding with this paper, we introduce the following assumptions:

Assumption 1 The regressor vector $\boldsymbol{u}_{k,i}$ arises from a zero-mean random process that is temporally stationary, temporally white, and independent in time and space.

Assumption 2 The noise $\boldsymbol{v}_{k,i}$ is zero-mean Gaussian, which is identically independent distributed in time and also spatially independent and is independent of other signals.

Assumption 3 The regressor vectors $\{\boldsymbol{u}_{k,i}\}$ are independent of $\{\boldsymbol{w}_{\ell,i}\}$ for all ℓ and for $j < i$.

These assumptions are commonly used in the adaptive filtering literature since they help simplify the analysis, and the performance results obtained under this assumption match well the actual performance of stand-alone filters for sufficiently small step sizes.

4 Performance Analysis

In this section, we will analyze the convergence and mean-square error performance of the algorithm (10) which is proposed in the third part.

4.1 Error Vector Recursion

We define the error quantities

$$\tilde{w}_{k,i} = \boldsymbol{w}^o - \boldsymbol{w}_{k,i}\ ,\ \tilde{\varphi}_{k,i} = \boldsymbol{w}^o - \varphi_{k,i}\ ,\ \tilde{\phi}_{k,i} = \boldsymbol{w}^o - \phi_{k,i} \tag{11}$$

By substituting Eq. (11) into Eq. (10), we get the following

$$\begin{cases} \tilde{\varphi}_{k,i} = \tilde{\boldsymbol{w}}_{k,i-1} - \mu \cdot \boldsymbol{u}_{k,i}^T(\boldsymbol{u}_{k,i}\tilde{\boldsymbol{w}}_{k,i-1} + v_{k,i}) & \text{(12a)} \\ \tilde{\phi}_{k,i} = \tilde{\varphi}_{k,i} + \lambda(\tilde{\boldsymbol{w}}_{k,i-1} - \tilde{\varphi}_{k,i-1}) & \text{(12b)} \\ \tilde{\boldsymbol{w}}_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell,k}\tilde{\phi}_{\ell,i} & \text{(12c)} \end{cases}$$

Then, by doing the dimension expansion, we have

$$\begin{cases} \tilde{\varphi}_i = (I_{MN} - \mu \cdot \boldsymbol{\mathcal{D}}_i)\tilde{\boldsymbol{w}}_{i-1} - \mu \cdot \boldsymbol{g}_i & \text{(13a)} \\ \tilde{\phi}_i = \tilde{\varphi}_i + \lambda\left(\tilde{\boldsymbol{w}}_{i-1} - \tilde{\varphi}_{i-1}\right) & \text{(13b)} \\ \tilde{\boldsymbol{w}}_i = \mathcal{A}^T\tilde{\phi}_i & \text{(13c)} \end{cases}$$

where

$$\boldsymbol{\mathcal{A}} \overset{\Delta}{=} \boldsymbol{A} \otimes \boldsymbol{I}_M \tag{14}$$

$$\tilde{\boldsymbol{w}}_i \overset{\Delta}{=} \text{col}\{\tilde{\boldsymbol{w}}_{1,i}, \tilde{\boldsymbol{w}}_{2,i}, \ldots\ldots., \tilde{\boldsymbol{w}}_{N,i}\} \tag{15}$$

$$\tilde{\varphi}_i \overset{\Delta}{=} \text{col}\{\tilde{\varphi}_{1,i}, \tilde{\varphi}_{2,i}, \ldots\ldots., \tilde{\varphi}_{N,i}\} \tag{16}$$

$$\tilde{\phi}_i \overset{\Delta}{=} \text{col}\{\tilde{\phi}_{1,i}, \tilde{\phi}_{2,i}, \ldots\ldots., \tilde{\phi}_{N,i}\} \tag{17}$$

$$\boldsymbol{g_i} \overset{\Delta}{=} \text{col}\{\boldsymbol{u}_{1,i}^T v_{1,i}, \ldots\ldots., \boldsymbol{u}_{N,i}^T v_{N,i}\} \tag{18}$$

$$\boldsymbol{\mathcal{D}}_i \overset{\Delta}{=} \text{diag}\{\boldsymbol{u}_{1,i}^T \boldsymbol{u}_{1,i}, \ldots\ldots., \boldsymbol{u}_{N,i}^T \boldsymbol{u}_{N,i}\} \tag{19}$$

Then, it can be verified that the network error vector $\tilde{\boldsymbol{w}}_{\boldsymbol{i}}$ for the diffusion strategy (10) evolves according to the following recursion:

$$\tilde{\boldsymbol{w}}_i = \boldsymbol{\mathcal{B}}_{1,i}\tilde{\boldsymbol{w}}_{i-1} + \boldsymbol{\mathcal{B}}_{2,i-1}\tilde{\boldsymbol{w}}_{i-2} - \mu\boldsymbol{\mathcal{A}}^T\boldsymbol{g}_i + \lambda\mu\boldsymbol{\mathcal{A}}^T\boldsymbol{g}_{i-1} \tag{20}$$

where

$$\boldsymbol{\mathcal{B}}_{1,i} = \boldsymbol{\mathcal{A}}^T\left[(\lambda+1)\,\boldsymbol{I}_{MN} - \mu\boldsymbol{\mathcal{D}}_i\right] \tag{21}$$

$$\boldsymbol{\mathcal{B}}_{2,i-1} = -\lambda\boldsymbol{\mathcal{A}}^T\left(\boldsymbol{I}_{MN} - \mu\boldsymbol{\mathcal{D}}_{i-1}\right) \tag{22}$$

Remark 1 In an extreme case when $\lambda = 0$, the error vector recursion (20) of the correction-based diffusion LMS algorithm equals to the error vector recursion of the DLMS algorithm.

4.2 Mean Stability Analysis

Since $\{\boldsymbol{v}_{k,i}\}$ is a zero-mean measurement noise and is independent of the regressor vectors $\{\boldsymbol{u}_{k,i}\}$. Therefore, taking the expectation of both sides of (20), we get

$$\mathbb{E}\{\tilde{\boldsymbol{w}}_i\} = \boldsymbol{\mathcal{B}}_1\mathbb{E}\{\tilde{\boldsymbol{w}}_{i-1}\} + \boldsymbol{\mathcal{B}}_2\mathbb{E}\left\{\tilde{\boldsymbol{w}}_{i-2}\right\} \tag{23}$$

where

$$\boldsymbol{\mathcal{B}}_1 = \mathbb{E}\{\boldsymbol{\mathcal{B}}_{1,i}\} = \boldsymbol{\mathcal{A}}^T\left[(\lambda+1)\,\boldsymbol{I}_{MN} - \mu\boldsymbol{\mathcal{R}}_u\right] \tag{24}$$

$$\boldsymbol{\mathcal{B}}_2 = \mathbb{E}\{\boldsymbol{\mathcal{B}}_{2,i-1}\} = -\lambda\boldsymbol{\mathcal{A}}^T\left(\boldsymbol{I}_{MN} - \mu\boldsymbol{\mathcal{R}}_u\right) \tag{25}$$

$$\boldsymbol{\mathcal{R}}_u = \mathbb{E}\{\boldsymbol{\mathcal{D}}_i\} = \text{diag}\{\boldsymbol{R}_{u,1}, \boldsymbol{R}_{u,2}, \ldots\ldots, \boldsymbol{R}_{u,N}\} \tag{26}$$

In order to facilitate the analysis below, we extend the equation (23) as follows

$$\begin{aligned}\mathbb{E}\left\{\begin{bmatrix}\tilde{\boldsymbol{w}}_i\\ \tilde{\boldsymbol{w}}_{i-1}\end{bmatrix}\right\} &= \begin{bmatrix}\boldsymbol{\mathcal{B}}_1 & \boldsymbol{\mathcal{B}}_2\\ \boldsymbol{I}_{MN} & \boldsymbol{0}_{MN}\end{bmatrix}\mathbb{E}\left\{\begin{bmatrix}\tilde{\boldsymbol{w}}_{i-1}\\ \tilde{\boldsymbol{w}}_{i-2}\end{bmatrix}\right\}\\ &= \boldsymbol{B}\mathbb{E}\left\{\begin{bmatrix}\tilde{\boldsymbol{w}}_{i-1}\\ \tilde{\boldsymbol{w}}_{i-2}\end{bmatrix}\right\}\end{aligned} \tag{27}$$

Recursion (27) converges as $i \to \infty$ if the matrix $\boldsymbol{B}$ is stable. Since $\boldsymbol{\mathcal{A}}$ is a double random matrix, we get

$$\begin{aligned}\rho(\boldsymbol{B}) &= \rho\left(\begin{bmatrix}\boldsymbol{\mathcal{B}}_1 & \boldsymbol{\mathcal{B}}_2 \\ \boldsymbol{I}_{MN} & \boldsymbol{0}_{MN}\end{bmatrix}\right) \\ &\le \rho\left(\begin{bmatrix}(\lambda+1)\boldsymbol{I}_{MN} - \mu\boldsymbol{\mathcal{R}}_u & -\lambda(\boldsymbol{I}_{MN} - \mu\boldsymbol{\mathcal{R}}_u) \\ \boldsymbol{I}_{MN} & \boldsymbol{0}_{MN}\end{bmatrix}\right) \\ &\le \rho\left(\begin{bmatrix}\lambda\boldsymbol{I}_{MN} & -\boldsymbol{I}_{MN} \\ \boldsymbol{0}_{MN} & \boldsymbol{I}_{MN} - \mu\boldsymbol{\mathcal{R}}_u\end{bmatrix}\right) \\ &< 1\end{aligned} \tag{28}$$

Thus, the stability of the algorithm (10) is ensured by choosing μ and λ such that:

$$0 < \mu < \frac{2}{\lambda_{\max}(\boldsymbol{R}_{u,k})} \quad \text{and} \quad 0 < \lambda < 1 \tag{29}$$

where $\lambda_{\max}(\boldsymbol{R}_{u,k})$ is the maximum eigenvalue of $\boldsymbol{R}_{u,k}$. We observe that the deviation reduces to 0, when $i \to \infty$.

It can be seen from condition (29) that the gain factor λ is not directly related to the step size μ, but only when the conditions of the step size and the gain factor are simultaneously established, the algorithm (10) converges.

Remark 2 When the gain factor satisfies the constraint condition (29), the step size condition of the algorithm is the same as the step size condition of the DLMS algorithm to guarantee stability.

4.3 Mean-Square Deviation Behavior Analysis

To perform the mean-square-error analysis, we shall use the Kronecker product operator [8] and the vectorization operator vec(·). First, we write the results by extending the equation (20).

$$\begin{aligned}\tilde{w}_{i-1}^{i} &= B_i \tilde{w}_{i-2}^{i-1} + A_2 g_{i-1}^{i} \\ &= \left(A_1 + A_2 \cdot \left(\mathcal{D}_{1,i} + \mathcal{D}_{2,i-1}\right)\right)\tilde{w}_{i-2}^{i-1} + A_2 g_{i-1}^{i}\end{aligned} \tag{30}$$

where

$$\tilde{\boldsymbol{w}}_{i-1}^{i} = \begin{bmatrix}\tilde{\boldsymbol{w}}_i \\ \tilde{\boldsymbol{w}}_{i-1}\end{bmatrix} \tag{31}$$

$$\boldsymbol{g}_{i-1}^{i} = \begin{bmatrix}\boldsymbol{g}_i \\ \boldsymbol{g}_{i-1}\end{bmatrix} \tag{32}$$

$$\boldsymbol{B}_i = \begin{bmatrix}\boldsymbol{\mathcal{B}}_{1,i} & \boldsymbol{\mathcal{B}}_{2,i-1} \\ \boldsymbol{I}_{MN} & \boldsymbol{0}_{MN}\end{bmatrix} \tag{33}$$

$$\boldsymbol{A}_1 = \begin{bmatrix} (\lambda+1)\boldsymbol{\mathcal{A}}^T & -\lambda\boldsymbol{\mathcal{A}}^T \\ \boldsymbol{I}_{MN} & \boldsymbol{0}_{MN} \end{bmatrix} \tag{34}$$

$$\boldsymbol{A}_2 = \begin{bmatrix} -\mu\boldsymbol{\mathcal{A}}^T & \lambda\mu\boldsymbol{\mathcal{A}}^T \\ \boldsymbol{0}_{MN} & \boldsymbol{0}_{MN} \end{bmatrix} \tag{35}$$

$$\boldsymbol{D}_{1,i} = \begin{bmatrix} \boldsymbol{\mathcal{D}}_i & \boldsymbol{0}_{MN} \\ \boldsymbol{0}_{MN} & \boldsymbol{0}_{MN} \end{bmatrix} \tag{36}$$

$$\boldsymbol{D}_{2,i-1} = \begin{bmatrix} \boldsymbol{0}_{MN} & \boldsymbol{0}_{MN} \\ \boldsymbol{0}_{MN} & \boldsymbol{\mathcal{D}}_{i-1} \end{bmatrix} \tag{37}$$

To analyze the convergence in mean-square-error sense, we consider evaluating a weighted variance of the error vector $\tilde{\boldsymbol{w}}_i$. Let $\boldsymbol{\Sigma}_1$ denotes an arbitrary nonnegative definite matrix that we are free to choose.

Let

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{38}$$

We have

$$\mathbb{E}\left\|\tilde{\boldsymbol{w}}_i\right\|^2_{\boldsymbol{\Sigma}_1} = \mathbb{E}\left\|\tilde{\boldsymbol{w}}^i_{i-1}\right\|^2_{\boldsymbol{\Sigma}} \tag{39}$$

Thus, we consider the variance of the weight error vector $\tilde{\boldsymbol{w}}^i_{i-1}$, weighted by any positive-definite matrix $\boldsymbol{\Sigma}$. According to assumption 1, 2, and 3, we obtain:

$$\begin{aligned} \mathbb{E}\left\|\tilde{\boldsymbol{w}}_i\right\|^2_{\boldsymbol{\Sigma}_1} &= \mathbb{E}\left\|\tilde{\boldsymbol{w}}^i_{i-1}\right\|^2_{\boldsymbol{\Sigma}} \\ &= \mathbb{E}\left\{\left\|\left(\boldsymbol{A}_1 + \boldsymbol{A}_2\cdot\left(\boldsymbol{D}_{1,i}+\boldsymbol{D}_{2,i-1}\right)\right)\tilde{\boldsymbol{w}}^{i-1}_{i-2} + \boldsymbol{A}_2\boldsymbol{g}^i_{i-1}\right\|^2_{\boldsymbol{\Sigma}}\right\} \\ &= \mathbb{E}\left\|\tilde{\boldsymbol{w}}^{i-1}_{i-2}\right\|^2_{\boldsymbol{\Sigma}'} + \mathbb{E}\left\|\boldsymbol{A}_2\boldsymbol{g}^i_{i-1}\right\|^2_{\boldsymbol{\Sigma}} + 2\mathbb{E}\left\{(\tilde{\boldsymbol{w}}^{i-1}_{i-2})^T\boldsymbol{B}_i{}^T\boldsymbol{\Sigma}\boldsymbol{A}_2\boldsymbol{g}^i_{i-1}\right\} \end{aligned} \tag{40}$$

where matrix $\boldsymbol{\Sigma}'$ is give by:

$$\begin{aligned} \boldsymbol{\Sigma}' = {} & \boldsymbol{B}^T\boldsymbol{\Sigma}\boldsymbol{B} - \boldsymbol{R}_{u,1}\boldsymbol{A}_2^T\boldsymbol{\Sigma}\boldsymbol{A}_2\boldsymbol{R}_{u,1} - \boldsymbol{R}_{u,2}\boldsymbol{A}_2^T\boldsymbol{\Sigma}\boldsymbol{A}_2\boldsymbol{R}_{u,2} \\ & + \mathbb{E}\left\{\boldsymbol{D}_{1,i}^T\boldsymbol{A}_2^T\boldsymbol{\Sigma}\boldsymbol{A}_2\boldsymbol{D}_{1,i}\right\} + \mathbb{E}\left\{\boldsymbol{D}_{2,i-1}^T\boldsymbol{A}_2^T\boldsymbol{\Sigma}\boldsymbol{A}_2\boldsymbol{D}_{2,i-1}\right\} \end{aligned} \tag{41}$$

where

$$\boldsymbol{B} = \boldsymbol{A}_1 + \boldsymbol{A}_2\cdot\left(\boldsymbol{R}_{u,1}+\boldsymbol{R}_{u,2}\right) \tag{42}$$

$$= \begin{bmatrix} \boldsymbol{\mathcal{B}}_1 & \boldsymbol{\mathcal{B}}_2 \\ \boldsymbol{I}_{MN} & \boldsymbol{0}_{MN} \end{bmatrix} \tag{43}$$

$$\boldsymbol{R}_{u,1} = \mathbb{E}\left\{\boldsymbol{D}_{1,i}^T\right\} = \begin{bmatrix} \boldsymbol{\mathcal{R}}_u & \boldsymbol{0}_{MN} \\ \boldsymbol{0}_{MN} & \boldsymbol{0}_{MN} \end{bmatrix} \tag{44}$$

$$\boldsymbol{R}_{u,2} = \mathbb{E}\left\{\boldsymbol{D}_{2,i-1}\right\} = \begin{bmatrix} \boldsymbol{0}_{MN} & \boldsymbol{0}_{MN} \\ \boldsymbol{0}_{MN} & \boldsymbol{\mathcal{R}}_u \end{bmatrix} \tag{45}$$

By performing the vectorization operator of the positive definite matrix $\boldsymbol{\Sigma}$, we obtain σ=vec($\boldsymbol{\Sigma}$). Similarly, let σ'=vec($\boldsymbol{\Sigma}'$). According to the property $\text{vec}(\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{B}) = (\boldsymbol{B}^T \otimes \boldsymbol{A})\sigma$, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the arbitrary matrices, we get

$$\text{vec}(\boldsymbol{B}^T\boldsymbol{\Sigma}\boldsymbol{B}) = (\boldsymbol{B}^T \otimes \boldsymbol{B}^T)\sigma \tag{46}$$

$$\begin{aligned} vec(\boldsymbol{R}_{u,1}\boldsymbol{A}_2^T\boldsymbol{\Sigma}\boldsymbol{A}_2\boldsymbol{R}_{u,1}) &= \left(\boldsymbol{R}_{u,1}\boldsymbol{A}_2^T \otimes \boldsymbol{R}_{u,1}\boldsymbol{A}_2^T\right)\sigma \\ &= \left(\boldsymbol{R}_{u,1} \otimes \boldsymbol{R}_{u,1}\right)\left(\boldsymbol{A}_2^T \otimes \boldsymbol{A}_2^T\right)\sigma \end{aligned} \tag{47}$$

$$\begin{aligned} \text{vec}(\boldsymbol{R}_{u,2}\boldsymbol{A}_2^T\boldsymbol{\Sigma}\boldsymbol{A}_2\boldsymbol{R}_{u,2}) &= \left(\boldsymbol{R}_{u,2}\boldsymbol{A}_2^T \otimes \boldsymbol{R}_{u,2}\boldsymbol{A}_2^T\right)\sigma \\ &= \left(\boldsymbol{R}_{u,2} \otimes \boldsymbol{R}_{u,2}\right)\left(\boldsymbol{A}_2^T \otimes \boldsymbol{A}_2^T\right)\sigma \end{aligned} \tag{48}$$

Assume now that the regressors are Gaussian zero-mean random vectors. Then, for any Hermitian matrix H it holds [21]:

$$\mathbb{E}\left\{\boldsymbol{u}_{k,i}^T\boldsymbol{u}_{k,i}\boldsymbol{H}\boldsymbol{u}_{\ell,i}^T\boldsymbol{u}_{\ell,i}\right\} = \boldsymbol{R}_{u,k}\boldsymbol{H}\boldsymbol{R}_{u,\ell} + \beta\delta_{k\ell}\boldsymbol{R}_{u,k}\text{Tr}(\boldsymbol{H}\boldsymbol{R}_{u,k}) \tag{49}$$

where $\beta = 1$ if the regressors are complex, and $\beta = 2$ if the regressors are real. And $\delta_{k\ell}$ indicates the correlation coefficient of $\boldsymbol{u}_{k,i}$ and $\boldsymbol{u}_{\ell,i}$.

Consider the matrices $\boldsymbol{K}_1 = \mathbb{E}\left\{\boldsymbol{D}_{1,i}^T\boldsymbol{Q}\boldsymbol{D}_{1,i}\right\}$ and $\boldsymbol{K}_2 = \mathbb{E}\left\{\boldsymbol{D}_{2,i-1}^T\boldsymbol{Q}\boldsymbol{D}_{2,i-1}\right\}$, where $\boldsymbol{Q} = \boldsymbol{A}_2^T\boldsymbol{\Sigma}\boldsymbol{A}_2$, we obtain

$$\begin{aligned} \boldsymbol{K} = &\begin{bmatrix} \boldsymbol{\mathcal{R}}_{u,1}^T\boldsymbol{Q}_{11}\boldsymbol{\mathcal{R}}_{u,1} & \cdots & \boldsymbol{\mathcal{R}}_{u,1}^T\boldsymbol{Q}_{1N}\boldsymbol{\mathcal{R}}_{u,N} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\mathcal{R}}_{u,N}^T\boldsymbol{Q}_{N1}\boldsymbol{\mathcal{R}}_{u,1} & \cdots & \boldsymbol{\mathcal{R}}_{u,N}^T\boldsymbol{Q}_{NN}\boldsymbol{\mathcal{R}}_{u,N} \end{bmatrix} \\ &+ \begin{bmatrix} \boldsymbol{\mathcal{R}}_{u,1}\boldsymbol{r}_1^T\boldsymbol{q}_{1,1} & \cdots & \boldsymbol{0}_M \\ \vdots & \ddots & \vdots \\ \boldsymbol{0}_M & \cdots & \boldsymbol{\mathcal{R}}_{u,N}\boldsymbol{r}_N^T\boldsymbol{q}_{N,N} \end{bmatrix} \end{aligned} \tag{50}$$

$$\boldsymbol{q}_{k,\ell} = \text{vec}(\boldsymbol{Q}_{k,\ell}), \boldsymbol{r}_k = \text{vec}(\boldsymbol{\mathcal{R}}_{u,k}) \tag{51}$$

$$\begin{aligned} \boldsymbol{K}_1 &= \text{diag}\{\boldsymbol{K}, \boldsymbol{0}_{MN}\} \\ &= \boldsymbol{R}_{u,1}^T\boldsymbol{Q}\boldsymbol{R}_{u,1} + \boldsymbol{R}_{u,1}\boldsymbol{Z}_m \end{aligned} \tag{52}$$

where $\boldsymbol{Z}_m$ =diag$\{\boldsymbol{r}_1^T\boldsymbol{q}_{1,1}, \cdots, \boldsymbol{r}_N^T\boldsymbol{q}_{N,N}, 0, \cdots, 0\}$. Taking the vector operator of the above matrix $\boldsymbol{K}$, we get

$$\begin{aligned} \text{vec}(\boldsymbol{K}_1) &= \left(\boldsymbol{R}_{u,1} \otimes \boldsymbol{R}_{u,1}\right)\text{vec}(\boldsymbol{Q}) + (\boldsymbol{I}_{2MN} \otimes \boldsymbol{R}_{u,1})\boldsymbol{\mathcal{L}}_m\text{vec}(\boldsymbol{Q}) \\ &= \left(\boldsymbol{R}_{u,1} \otimes \boldsymbol{R}_{u,1} + (\boldsymbol{I}_{2MN} \otimes \boldsymbol{R}_{u,1})\boldsymbol{\mathcal{L}}_m\right) \times \left(\boldsymbol{A}_2^T \otimes \boldsymbol{A}_2^T\right)\sigma \end{aligned} \tag{53}$$

where $\mathcal{L}_m$ is given as follows:

$$\mathcal{L}_m = \mathrm{diag}\{\mathcal{L}_{m,1}, \cdots, \mathcal{L}_{m,N}, 0, \cdots, 0\} \tag{54}$$

$$\mathcal{L}_{m,k} = \begin{bmatrix} \boldsymbol{I}_k \otimes e_1 \\ \vdots \\ \boldsymbol{I}_k \otimes e_M \end{bmatrix} \otimes \boldsymbol{r}_k^T \tag{55}$$

Similarly, we also have

$$\begin{aligned} \mathrm{vec}(\boldsymbol{K}_2) &= \left(\boldsymbol{R}_{u,2} \otimes \boldsymbol{R}_{u,2}\right) \mathrm{vec}(\boldsymbol{Q}) + (\boldsymbol{I}_{2MN} \otimes \boldsymbol{R}_{u,2})\mathcal{L}_n \mathrm{vec}(\boldsymbol{Q}) \\ &= \left(\boldsymbol{R}_{u,2} \otimes \boldsymbol{R}_{u,2} + (\boldsymbol{I}_{2MN} \otimes \boldsymbol{R}_{u,2})\mathcal{L}_n\right) \times \left(\boldsymbol{A}_2^T \otimes \boldsymbol{A}_2^T\right) \sigma \end{aligned} \tag{56}$$

where $\mathcal{L}_n$ is given as follows:

$$\mathcal{L}_n = \mathrm{diag}\{0, \cdots, 0, \mathcal{L}_{n,1}, \cdots, \mathcal{L}_{n,N}\} \tag{57}$$

$$\mathcal{L}_{n,k} = \begin{bmatrix} \boldsymbol{I}_{N+k} \otimes e_1 \\ \vdots \\ \boldsymbol{I}_{N+k} \otimes e_M \end{bmatrix} \otimes \boldsymbol{r}_k^T \tag{58}$$

where $\boldsymbol{I}_k$ is a $2N \times 2N$ matrix with a unit entry at position K and zeros elsewhere, e_k is a $M \times 1$ column vector with a unit entry at position K and zeros elsewhere.

Then, we have

$$\mathrm{vec}\left(\boldsymbol{\Sigma}'\right) = \mathcal{F}\sigma \tag{59}$$

where

$$\mathcal{F} = \left(\boldsymbol{B}^T \otimes \boldsymbol{B}^T\right) + ((\boldsymbol{I}_{2MN} \otimes \boldsymbol{R}_{u,1})\mathcal{L}_m + (\boldsymbol{I}_{2MN} \otimes \boldsymbol{R}_{u,2})\mathcal{L}_n) \left(\boldsymbol{A}_2^T \otimes \boldsymbol{A}_2^T\right) \sigma \tag{60}$$

The second term of Eq. (40) can be written as:

$$\begin{aligned} \mathbb{E}\left\|\boldsymbol{A}_2 \boldsymbol{g}_{i-1}^i\right\|_{\boldsymbol{\Sigma}}^2 &= \mathbb{E}\left\{(\boldsymbol{g}_{i-1}^i)^T \boldsymbol{A}_2^T \boldsymbol{\Sigma} \boldsymbol{A}_2 \boldsymbol{g}_{i-1}^i\right\} \\ &= \mathrm{vec}((\lambda^2 + 1)\boldsymbol{H}_1)^T \sigma \end{aligned} \tag{61}$$

where

$$\boldsymbol{H}_1 = \begin{bmatrix} \mu^2 \sigma_v^2 \boldsymbol{\mathcal{A}}^T \boldsymbol{\mathcal{R}}_u \boldsymbol{\mathcal{A}} & \boldsymbol{0}_{MN} \\ \boldsymbol{0}_{MN} & \boldsymbol{0}_{MN} \end{bmatrix} \tag{62}$$

The third term of Eq. (35) can be written as:

$$\mathbb{E}\left\{\boldsymbol{B}_i{}^T \boldsymbol{\Sigma} \boldsymbol{A}_2 \boldsymbol{G}_{i-1}^i\right\} = \mathrm{vec}(-\lambda \boldsymbol{H}_2)^T \sigma \tag{63}$$

where

$$\boldsymbol{H}_2 = \begin{bmatrix} \mu^2\sigma_v^2\mathcal{B}_1\boldsymbol{\mathcal{A}}^T\mathcal{R}_u\boldsymbol{\mathcal{A}}\ \mathbf{0}_{MN} \\ \mu^2\sigma_v^2\boldsymbol{\mathcal{A}}^T\mathcal{R}_u\boldsymbol{\mathcal{A}}\ \ \mathbf{0}_{MN} \end{bmatrix} \tag{64}$$

Finally, we have

$$\mathbb{E}\left\|\tilde{\boldsymbol{w}}_{i-1}^i\right\|_\sigma^2 = \mathbb{E}\left\|\tilde{\boldsymbol{w}}_{i-2}^{i-1}\right\|_{\mathcal{F}\sigma}^2 + \text{vec}\left((\lambda^2+1)\boldsymbol{H}_1 - 2\lambda\boldsymbol{H}_2\right)\sigma \tag{65}$$

4.3.1 Convergence Analysis

A necessary and sufficient condition for the convergence of $E\left\|\tilde{\boldsymbol{w}}_{i-1}^i\right\|_\sigma^2$ is that $\mathcal{F}$ is a stable matrix, namely, $\rho(\mathcal{F}) < 1$. A simpler, approximate condition as shown below can be obtained for small step sizes [16]:

$$\mathcal{F} = \mathbb{E}\{\boldsymbol{B}^T \otimes \boldsymbol{B}^T\} \approx \boldsymbol{B}^T \otimes \boldsymbol{B}^T \tag{66}$$

which $\mathcal{F}$ is stable if $\boldsymbol{B}$ is stable. This condition is the same as the condition for mean stability (29) and can be easily checked.

4.3.2 Steady-State Performance

Notice that the stability of $\mathcal{F}$ guarantees that $\boldsymbol{I} - \mathcal{F}$ will be invertible. Thus, we have

$$\lim_{i\to\infty} \mathbb{E}\left\|\boldsymbol{w}_{i-1}^i\right\|_{(\boldsymbol{I}-\mathcal{F})\sigma}^2 = \text{vec}((\lambda^2+1)\boldsymbol{H}_1 - 2\lambda\boldsymbol{H}_2)\sigma \tag{67}$$

Then, choosing

$$\sigma = (\boldsymbol{I} - \mathcal{F})^{-1}\text{vec}(\text{diag}(\boldsymbol{e}_k) \otimes \boldsymbol{I}_M) \tag{68}$$

where $\boldsymbol{e}_k$ is a $2N-$dimension column vector that has a position unit entry at k and zeros elsewhere. Then,

$$\begin{aligned} \text{MSD}_k &= \mathbb{E}\left\|\tilde{\boldsymbol{w}}_{k,\infty}\right\|^2 \\ &= \text{vec}((\lambda^2+1)\boldsymbol{H}_1 - 2\lambda\boldsymbol{H}_2)(\boldsymbol{I} - \mathcal{F})^{-1} \times \text{vec}(\text{diag}(\boldsymbol{e}_k) \otimes \boldsymbol{I}_M) \end{aligned} \tag{69}$$

where MSD_k denotes the mean-square deviation of node k. Then, the network mean-square deviation (MSD) as follows:

$$
\begin{aligned}
MSD^{network} &\triangleq \frac{1}{N}\sum_{k=1}^{N} MSD_k \\
&= \frac{1}{N}\sum_{k=1}^{N} \mathrm{vec}((\lambda^2+1)\boldsymbol{H}_1 - 2\lambda \boldsymbol{H}_2)(\boldsymbol{I}-\boldsymbol{\mathcal{F}})^{-1} \times \mathrm{vec}(\mathrm{diag}(\boldsymbol{e}_k)\otimes \boldsymbol{I}_M)
\end{aligned}
\tag{70}
$$

Remark 3 From Eq. (70), we can see that the performance of the correction-based diffusion LMS algorithm is related to the gain factor. In an extreme case when $\lambda = 0$, the performance of the correction-based diffusion LMS algorithm equals to the performance of the DLMS algorithm.

5 Simulation Results

5.1 Example 1

In this section, in order to analyze the performance of correction-based diffusion LMS algorithm, we choose a network topology consisting of 20 nodes. The combination matrix $\boldsymbol{A}$ satisfies the double-stochastic property. The entry $a_{\ell,k}$ satisfies the Metropolis rule [11–14] as follows:

$$
a_{\ell,k} = \begin{cases} \frac{1}{\max(n_k, n_\ell)}, & \text{if } \ell \neq k, \ell \in N_k \\ 1 - \sum\limits_{\substack{\ell\in N_k \\ \ell\neq k}} a_{\ell,k}, & \text{if } \ell = k \\ 0, & \text{if } \ell \notin N_k \end{cases} \tag{71}
$$

Figure 1 depicts the network topology that is generated according to matrix A with $N = 20$ nodes.

The initial estimate $\boldsymbol{w}_{k,1}$ and the initial intermediate value $\boldsymbol{\varphi}_{k,1}$ in the k node are, respectively, selected to be $\boldsymbol{w}_{k,1} = [0, 0]^T$, $\boldsymbol{\varphi}_{k,1} = [0, 0]^T$. The regression vectors $\boldsymbol{u}_{k,i}$ are a 1×2 zero-mean Gaussian distributed with covariance matrix $\boldsymbol{R}_{u,k} = \boldsymbol{I}_2$. The noises $v_{k,i}$ are a zero-mean Gaussian random variable, independent of any other signal with variance $\sigma_{v,k}^2 = 0.25$. In addition, we assume the estimated parameter $w^o = [1, 2]^T$.

We compared the proposed algorithm when gain factor is taken different values. To guarantee almost the same initial convergence rate, we set the step size at $\mu = 0.02$ for all the algorithms. And the results are averaged over 100 independent experiments. Figure 2 shows the learning curve for correction-based DLMS algorithm when the gain factor is taken $\lambda = 0$, $\lambda = 0.2$, $\lambda = 0.4$, and $\lambda = 0.6$, and $\lambda = 0.8$, respectively. Figure 3 shows the learning curve for correction-based DLMS algorithm when the gain factor is taken $\lambda = 0.8$, $\lambda = 0.85$, $\lambda = 0.9$, $\lambda = 0.95$, and $\lambda = 0.98$, respectively. As can be seen from the figure, when the gain factor takes different values, learning curve will also be different. From Fig. 2, we can observe that when the gain factor is

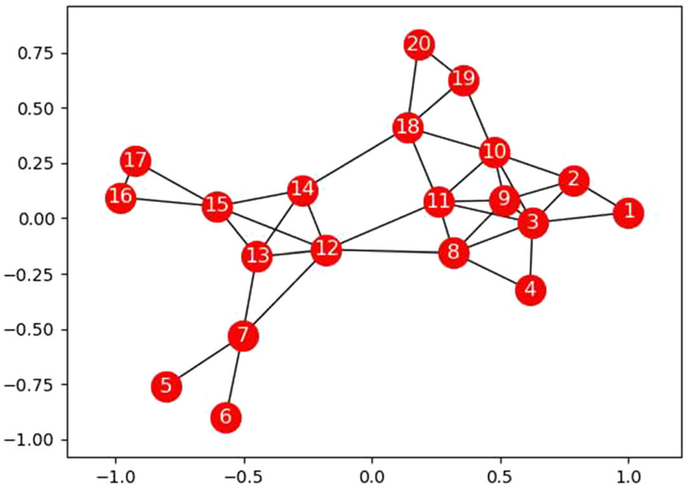


Fig. 1 Network topology consisting of 20 nodes

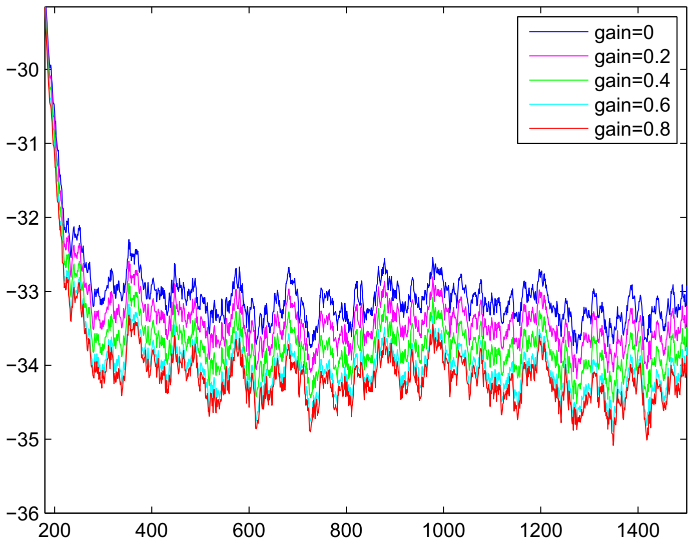


Fig. 2 The learning curve for correction-based DLMS algorithm when the gain factor is taken $\lambda = 0$, $\lambda = 0.2$, $\lambda = 0.4$, $\lambda = 0.6$, and $\lambda = 0.8$, respectively

between 0 and 0.8, the MSD gradually decreases as the gain factor increases. From Fig. 3, we can observe that the MSD is proportional to the gain factor if the gain factor is chosen between 0.8 and 0.98. Therefore, we obtain that the learning curve with a gain factor of 0.8 exhibits better performance than learning curves with other gain values.

We paint the theoretical mean-square deviation that was derived in Sect. 4. Figure 4 shows the theoretical MSD when the gain factor λ takes different values. By contrast,

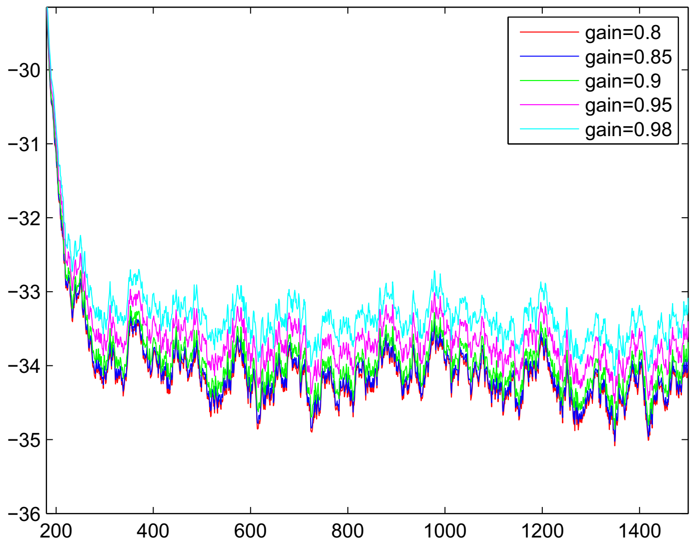


Fig. 3 The learning curve for correction-based DLMS algorithm when the gain factor is taken $\lambda = 0.8$, $\lambda = 0.85$, $\lambda = 0.9$, $\lambda = 0.95$, and $\lambda = 0.98$, respectively

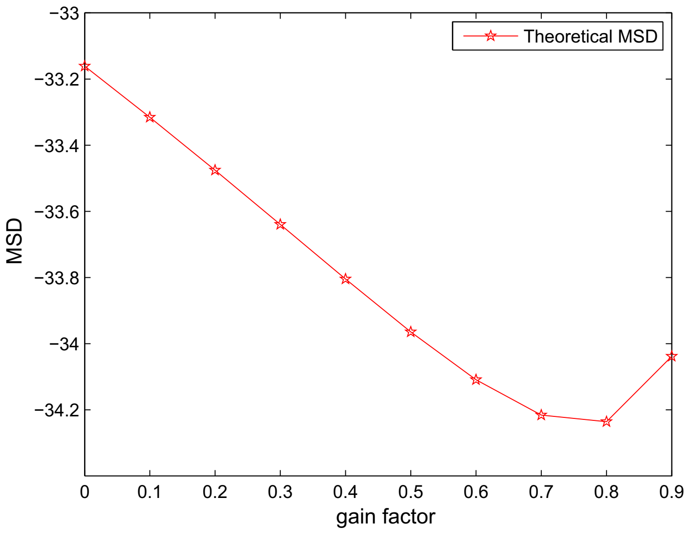


Fig. 4 The theoretical MSD when the gain factor λ takes different values

it is obvious that theoretical value of the MSD with a gain factor of 0.8 outperforms the theoretical MSD with other gain values.

We simulate the theoretical value of the MSD analyzed in Sect. 4 and compare the theoretical value with the simulated value of the algorithm as shown in Fig. 5. It can be observed from the figure that the theoretical value of the DLMS algorithm is -33.161dB, and the theoretical value of our proposed algorithms is -34.236dB

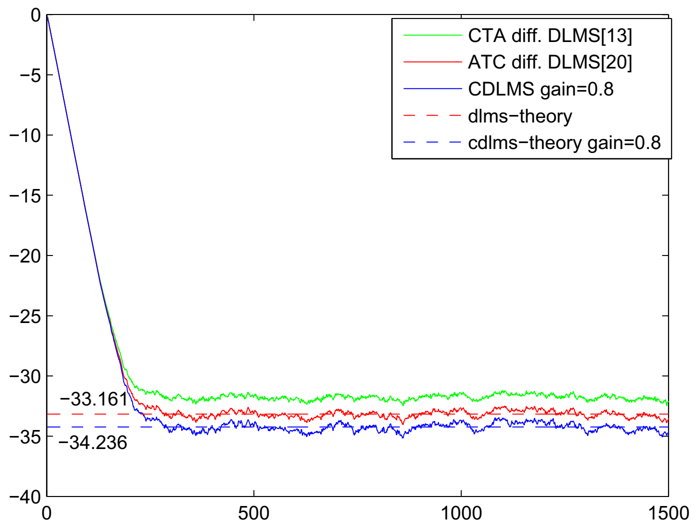


Fig. 5 Simulation values and theoretical values of two algorithms include DLMS algorithm [11] and C-DLMS algorithm

when the gain factor is 0.8. The difference between the two is about 1dB. It is obvious that our theoretical results are well matched to the simulation results. In addition, we compare the adapt-then-combine (ATC) DLMS algorithm, the combine-then-adapt (CTA) DLMS algorithm, and the correction-based diffusion LMS algorithm, as shown in Fig. 5, we can observe that correction-based diffusion LMS algorithm has better convergence performance than the other two algorithms.

5.2 Example 2

To verify the performance of the proposed algorithm in different networks, consider a distributed sensor network consisting of 12 nodes. These sensors are randomly distributed in an area of 100×100. When the distance between two sensors is less than $R = 20\,\text{m}$, data transmission is allowed. The final network topology is shown in Fig. 6 [15].

The unknown parameter $\boldsymbol{w}^o$ is set to a 4×1 random vector, the measurement noise $v_{k,i}$ is randomly selected between $(0.1, 0.2)$, and the regression vector is set to a $0-$mean Gaussian vector with $\boldsymbol{R}_{u,k} = \boldsymbol{I}_2$, and the step size μ is chosen to be 0.02.

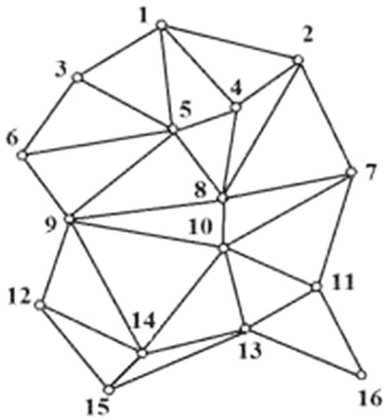
Figure 7 shows the learning curve for correction-based DLMS algorithm when the gain factor is taken $\lambda = 0$, $\lambda = 0.2$, $\lambda = 0.4$, and $\lambda = 0.6$, and $\lambda = 0.8$, respectively.

Figure 8 shows the learning curve for correction-based DLMS algorithm when the gain factor is taken $\lambda = 0.8$, $\lambda = 0.85$, $\lambda = 0.9$, $\lambda = 0.95$, and $\lambda = 0.98$, respectively.

It is obvious that theoretical value of the MSD with a gain factor of 0.8 outperforms the theoretical MSD with other gain values.

Simulation results suggest that our proposed algorithm performs better than the least mean-square algorithm when the gain factor is chosen to be appropriate.

Fig. 6 Network topology consisting of 12 nodes



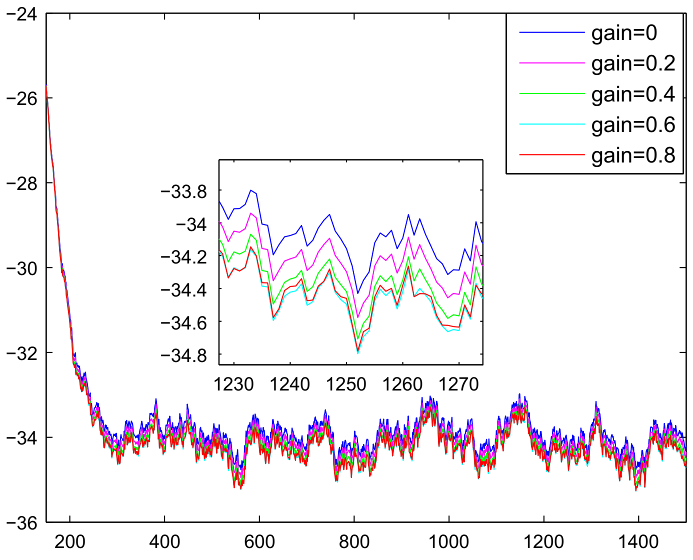


Fig. 7 The learning curve for correction-based DLMS algorithm when the gain factor is taken $\lambda = 0$, $\lambda = 0.2$, $\lambda = 0.4$, $\lambda = 0.6$, and $\lambda = 0.8$, respectively

6 Conclusion

In this paper, we proposed a correction-based diffusion least mean-square algorithm. Then, we analyzed the stability and mean-square error performance of the algorithm and derived sufficient conditions to ensure the convergence of the algorithm.

1. The correction-based diffusion least mean-square algorithm proposed outperforms the original diffusion LMS, and better performance can be obtained if the correction factor is properly selected.

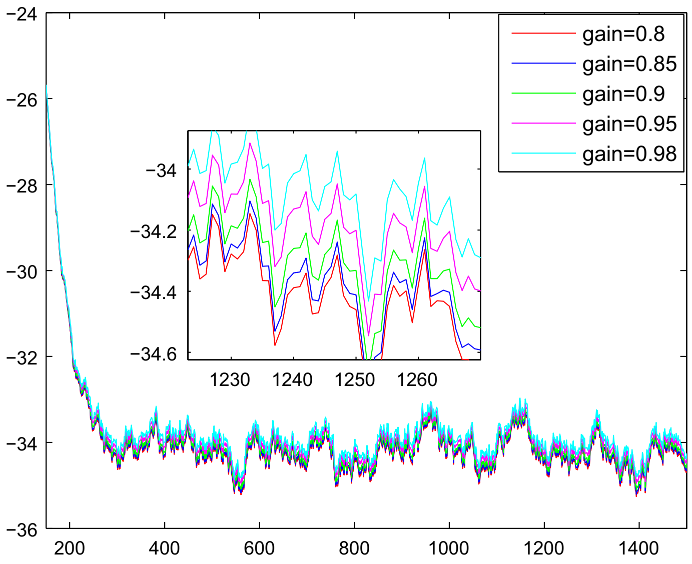


Fig. 8 The learning curve for correction-based DLMS algorithm when the gain factor is taken $\lambda = 0.8$, $\lambda = 0.85$, $\lambda = 0.9$, $\lambda = 0.95$, and $\lambda = 0.98$, respectively

2. In the traditional LMS algorithm, the convergence of the algorithm is only related to the step size. However, the convergence of the proposed algorithm is affected not only by the step size but also by the gain factor.

Finally, we expect our proposed algorithm to be applied to other situations, such as cyber attacks, equality constraints, and so on. Next, we will study the algorithm in the case of cyber attacks.

Acknowledgements This work was supported by National Key R&D Program of China (2018YFB1402600) and NSFC (61976013).

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. R. Abdolee, B. Champagne, A.H. Sayed, A diffusion LMS strategy for parameter estimation in noisy regressor applications, in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Bucharest, pp. 749–753 (2012)
2. P. Braca, S. Marano, V. Matta, Running consensus in wireless sensor networks, in *Proceedings of the IEEE International Conference on Information Fusion*, Cologne, Germany, p. 1C6 (2008)
3. F.S. Cattivelli, A.H. Sayed, Diffusion LMS strategies for distributed estimation. IEEE Trans. Signal Process. **58**(3), 1035–1048 (2010)
4. B. Chen, L. Xing, N. Zheng, J.C. Prncipe, Quantized minimum error entropy criterion. IEEE Trans. Neural Netw. Learn. Syst. **30**(5), 1370–1380 (2019)

5. J. Chen, A.H. Sayed, Diffusion adaptation strategies for distributed optimization and learning over networks. IEEE Trans. Signal Process. **60**(8), 4289C4305 (2012)
6. J. Chen, A.H. Sayed, On the learning behavior of adaptive networks part I: transient analysis. IEEE Trans. Inf. Theory **61**(6), 3487C3517 (2015)
7. J. Chen, A.H. Sayed, On the learning behavior of adaptive networks part II: performance analysis. IEEE Trans. Inf. Theory **61**(6), 3518C3548 (2015)
8. R.H. Koning, H. Neudecker, T. Wansbeek, Block Kronecker products and the vecb operator. Linear Algebra Appl. **149**, 165–184 (1991)
9. J. Lee, S.E. Kim, W.J. Song, Data-selective diffusion LMS for reducing communication overhead. Signal Process. **113**(7), 211–217 (2015)
10. Y. Liu, C. Li, Secure distributed estimation over wireless sensor networks under attacks. IEEE Trans. Aerosp. Electron. Syst. **54**(4), 1815–1831 (2018)
11. C.G. Lopes, A.H. Sayed, Diffusion least-mean squares over adaptive networks: formulation and performance analysis. IEEE Trans. Signal Process. **56**(7), 31223136 (2008)
12. L. Lu, H. Zhao, W. Wang, Y. Yu, Performance analysis of the robust diffusion normalized least mean p-power algorithm. IEEE Trans. Circuits Syst. II Express Briefs **65**(12), 2047–2051 (2018)
13. L. Lu, Z. Zheng, B. Champagne, X. Yang, W. Wu, Self-regularized nonlinear diffusion algorithm based on levenberg gradient descent. Signal Process. **163**, 107–114 (2019)
14. L. Lu, H. Zhao, B. Champagne, Diffusion total least-squares algorithm with multi-node feedback. Signal Process. **153**, 243–254 (2018)
15. R. Mohammadloo, G. Azarnia, M. A. Tinati, Increasing the initial convergence of distributed diffusion LMS algorithm by a new variable tap-length variable step-size method, in *21st Iranian Conference on Electrical Engineering (ICEE)*, Mashhad, pp. 1–5 (2013)
16. R. Nassif, C. Richard, A. Ferrari, A.H. Sayed, Diffusion LMS for multitask problems with local linear equality constraints. IEEE Trans. Signal Process. **65**(19), 4979–4993 (2017)
17. R. Olfati-Saber, J.S. Shamma, Consensus filters for sensor networks and distributed sensor fusion, in *Proceedings of IEEE Conference on Decision and Control (CDC)*. IEEE, p. 66986703 (2005)
18. M. Omer Bin Saeed, A. Zerguine, S.A. Zummo, Variable step-size least mean square algorithms over adaptive networks, in *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*, Kuala Lumpur, pp. 381–384 (2010)
19. S. Sardellitti, M. Giona, S. Barbarossa, Fast distributed average consensus algorithms based on advection-diffusion processes. IEEE Trans. Signal Process. **58**(2), 826842 (2010)
20. A.H. Sayed, Diffusion strategies for adaptation and learning over networks. IEEE Signal Process. Mag. **30**(3), 155–171 (2013)
21. A.H. Sayed, *Fundamentals of Adaptive Filtering* (Wiley, New York, 2003)
22. A.H. Sayed, Adaptive networks, in *Proceedings of the IEEE*, vol. 102, no. 4, p. 460C497 (2014)
23. A.H. Sayed, Adaptation, learning, and optimization over networks. Found. Trends Mach. Learn. **7**(4–5), 311C801 (2014)
24. N. Takahashi, I. Yamada, A.H. Sayed, Diffusion least-mean squares with adaptive combiners: formulation and performance analysis. IEEE Trans. Signal Process. **58**(9), 4795–4810 (2010)
25. K. Yuan, Q. Ling, W. Yin, On the convergence of decentralized gradient descent. SIAM J. Optim. **26**(3), 1835C1854 (2016)
26. K. Yuan, B. Ying, X. Zhao, A.H. Sayed, Exact diffusion for distributed optimization and learning-part I: algorithm development. IEEE Trans. Signal Process. **67**(3), 708–723 (2019)
27. K. Yuan, B. Ying, X. Zhao, A.H. Sayed, Exact diffusion for distributed optimization and learning-part II: convergence analysis. IEEE Trans. Signal Process. **67**(3), 724–739 (2019)