DIFFUSION-BASED BIAS-COMPENSATED RLS FOR DISTRIBUTED ESTIMATION OVER ADAPTIVE SENSOR NETWORKS

Alexander Bertrand[*], *Marc Moonen*[*] *and Ali H. Sayed*[†]

[*] Electrical Engineering Dept. (ESAT-SCD)
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium
E-mail: alexander.bertrand@esat.kuleuven.be,
marc.moonen@esat.kuleuven.be

[†] Electrical Engineering Dept.
University of California
Los Angeles, CA 90095, USA
E-mail: sayed@ee.ucla.edu

ABSTRACT

We present a diffusion-based bias-compensated recursive least squares (RLS) algorithm for distributed estimation in ad-hoc adaptive sensor networks where nodes cooperate to estimate a common deterministic parameter vector. It is assumed that both the regressors and the output response are corrupted by stationary additive noise. In this case, the least-squares estimator is biased. Assuming that a good estimate of the noise statistics is available, this bias can be removed at the cost of a larger variance of the estimator. However, by letting nodes cooperate in a diffusion-based fashion, it is possible to significantly reduce the variance, and furthermore improve the stability of the algorithm. If there are estimation errors in the noise statistics, the diffusion also results in a smaller residual bias. We provide closed-form expressions for the residual bias and mean-square deviation of the estimate (without full derivations). We also provide simulation results to demonstrate the beneficial effect of diffusion.

1. INTRODUCTION

We study the problem of distributed bias-compensated least-squares estimation over ad-hoc adaptive sensor networks, where the nodes collaborate to estimate and track a common deterministic parameter vector. It is assumed that both the regressors and the output response are corrupted by stationary additive noise. In this case, the least-squares estimator is biased, which is often undesirable. This is for example a common problem in the analysis of auto-regressive (AR) processes based on noisy observations.

If the noise is white, total least squares (TLS) estimation can be applied [1], which has also been described for the case of ad-hoc networks [2]. If accurate estimates of the noise statistics are available, the bias can also be removed by means of the bias-compensation principle [3]. In this paper, we use a similar bias compensation for the case where the noise is colored and/or correlated with the noise on the output response. A bias-compensated recursive least-squares (BC-RLS) algorithm is proposed, based on an exponentially weighted least-squares estimation problem. The latter allows to track the parameter vector when it changes over time, by putting less weight on older samples in the estimation.

The compensation of the bias usually results in an increased variance of the estimator, which is a common observation in estimation theory. However, recent developments in adaptive filtering have demonstrated that it is possible to significantly reduce the variance, by letting different nodes combine their local estimates with those of their neighbors [4–8]. The mode of cooperation that is adopted in this paper, is known as diffusion adaptation [4–7]. Simulations will demonstrate that diffusion indeed reduces the mean-square deviation[1] (MSD) of the algorithm, and furthermore, it reduces the residual bias resulting from possible estimation errors in the estimates of the noise statistics.

We provide closed-form expressions for the MSD and the residual bias, under some assumptions that are common in the adaptive filtering literature. Due to space constraints, we only list the most relevant results, without the full derivation. The latter can be found in a complementary paper [9]. We explain how these expressions give insight in the fact that diffusion increases stability, and decreases the residual bias and estimator MSD.

Notation

We use boldface letters for random quantities and normal font for non-random (deterministic) quantities or samples of random quantities. We use capital letters for matrices and small letters for vectors. The superscript H denotes complex-conjugate transposition. The index i is used to denote time instants, and the index k is used to denote different nodes in a network with a total of N nodes. We use $E\{\mathbf{x}\}$ to denote the expected value of $\mathbf{x}$.

2. LEAST SQUARES ESTIMATION WITH BIAS COMPENSATION

2.1 Problem Statement

Consider an ad-hoc sensor network with N nodes. The objective for each node is to estimate a common deterministic $M \times 1$ parameter vector w^o. At every time instant i, node k collects a measurement $d_k(i)$ (referred to as the 'output response') that is assumed to be related to the unknown vector w^o by

$$d_k(i) = \overline{u}_{k,i} w^o + v_k(i) \tag{1}$$

where the regressor $\overline{u}_{k,i}$ is a $1 \times M$ row vector[2] of length M, and $v_k(i)$ is a sample of a zero-mean stationary noise process $\mathbf{v}_k$ with variance $\sigma^2_{v_k}$. In [5–8], it was assumed that node k also has access

Alexander Bertrand is supported by a Ph.D. grant of I.W.T. (Flemish Institute for the Promotion of Innovation through Science and Technology). This research work was conducted during A. Bertrand's visit to the UCLA Adaptive Systems Laboratory as part of a collaboration with the ESAT Laboratory of Katholieke Universiteit Leuven, which was funded by a travel grant from FWO Flanders. The research was supported by K.U.Leuven Research Council CoE EF/05/006 'Optimization in Engineering' (OPTEC) and PFV/10/002 (OPTEC), Concerted Research Action GOA-MaNet, the Belgian Programme on Interuniversity Attraction Poles initiated by the Belgian Federal Science Policy Office IUAP P6/04 (DYSCO, 'Dynamical systems, control and optimization', 2007-2011), and Research Project FWO nr. G.0600.08 ('Signal processing and network design for wireless acoustic sensor networks'). The scientific responsibility is assumed by its authors. The work of A. H. Sayed was supported in part by NSF grants CCF-1011918 and CCF-0942936.

[1] The MSD is closely related to the variance of the estimator. It is defined as the expected value of the squared Euclidean norm of the difference between the true parameter vector and the estimated vector.

[2] We adopt the notation of [5–7, 10], i.e., the regressors are defined as row vectors, rather than column vectors.

to the regressors $\{\overline{u}_{k,i}\}$. Here, we assume that node k observes noisy regressors $\{u_{k,i}\}$, given by

$$u_{k,i} = \overline{u}_{k,i} + n_{k,i} \tag{2}$$

with the $1 \times M$ regression vector $n_{k,i}$ denoting a sample of a zero-mean stationary noise process $\mathbf{n}_k$ with covariance matrix $R_{n_k} = E\{\mathbf{n}_k^H \mathbf{n}_k\}$. We assume that $\mathbf{n}_k$ is uncorrelated with the regressors $\overline{\mathbf{u}}_{k,i}$, and that $\mathbf{n}_k$ and $\mathbf{v}_k$ are correlated[3], yielding a non-zero covariance vector $r_{n_k v_k} = E\{\mathbf{n}_k^H \mathbf{v}_k\}$.

The local least squares (LS) estimate of w^o at node k at time instant i, based on the noisy regressors, is the solution of the optimization problem

$$\hat{w}_{k,i} = \arg\min_w \sum_{j=1}^{i} \left(d_k(j) - u_{k,j} w\right)^2 + \delta \|w\|_2^2 \tag{3}$$

where δ is a small positive number that serves as a regularization parameter. The solution of (3) is given by

$$\hat{w}_{k,i} = \hat{R}_{u_k,i}^{-1} \hat{r}_{u_k d_k,i} \tag{4}$$

where

$$\hat{R}_{u_k,i} = \frac{1}{i+1} \left(\sum_{j=1}^{i} u_{k,j}^H u_{k,j} + \delta I_M \right) \tag{5}$$

$$\hat{r}_{u_k d_k,i} = \frac{1}{i+1} \sum_{j=1}^{i} u_{k,j}^H d_k(j) \tag{6}$$

and where I_M denotes the $M \times M$ identity matrix. The normalization with $1/(i+1)$ does not have an influence on $\hat{w}_{k,i}$, but its purpose will become clear in Section 2.2. Since we use noisy regressors, the LS estimate has a bias. In the case of stationary and ergodic data, it can be verified that

$$\hat{w}_k = w^o + w_k^b \tag{7}$$

$$w_k^b = R_{u_k}^{-1} \left(r_{n_k v_k} - R_{n_k} w^o \right) . \tag{8}$$

where $\hat{w}_k = \lim_{i\to\infty} \hat{w}_{k,i} = R_{u_k}^{-1} r_{u_k d_k}$ with $R_{u_k} = E\{\mathbf{u}_{k,i}^H \mathbf{u}_{k,i}\}$ and $r_{u_k d_k} = E\{\mathbf{u}_{k,i}^H \mathbf{d}_k(i)\}$, for all $i \in \mathbb{N}$.

2.2 Bias-Compensated Least Squares (BC-LS)

Several BC-LS algorithms have been proposed for the white noise case ($R_{n_k} = \sigma_{n_k}^2 I_M$), which are asymptotically unbiased when the number of observations goes to infinity, e.g., [11]. BC-LS algorithms are based on the bias compensation principle [3], i.e., if the asymptotic bias w_k^b can be estimated, it can be subtracted from the LS estimate $w_{k,i}^{LS}$ to obtain the unbiased estimate (generalized here to incorporate colored noise and mutually correlated noise):

$$\psi_{k,i} \stackrel{\Delta}{=} \hat{w}_{k,i} + \hat{R}_{u_k,i}^{-1} \left(\hat{R}_{n_k} w^o - \hat{r}_{n_k v_k} \right) \tag{9}$$

where $\hat{r}_{n_k v_k}$ and $\hat{R}_{n_k}$ are estimates of $r_{n_k v_k}$ and R_{n_k}, respectively. In the sequel, it is assumed that good estimates $\hat{r}_{n_k v_k}$ and $\hat{R}_{n_k}$ are available. In the case of white noise, these estimates can be computed blindly during operation of the algorithm [11]. Otherwise, other techniques are required, e.g., in the case of AR estimation in speech signals, the noise statistics can be estimated during silent periods in between words and sentences.

Since w^o is unknown in (9), it has to be replaced with an estimate. The common approach is then to use $\psi_{k,i-1}$ as an estimate for w^o in (9). We then obtain the recursive algorithm

$$\psi_{k,i} = \hat{w}_{k,i} + \hat{R}_{u_k,i}^{-1} \left(\hat{R}_{n_k} \psi_{k,i-1} - \hat{r}_{n_k v_k} \right) . \tag{10}$$

[3]For example, this may be the case for the estimation of the prediction coefficients of an auto-regressive (AR) process where the data is corrupted by additive colored noise, e.g., in real-time analysis of a speech signal that is recorded in a distributed microphone network.

2.3 Bias-Compensated Recursive Least Squares (BC-RLS)

In this paper, we first modify the above estimation algorithm to fit into an adaptive filtering context, and incorporate exponential weighting (for tracking purposes). The exponentially-weighted LS estimate (at node k) solves the optimization problem

$$\hat{w}_{k,i} = \arg\min_w \sum_{j=1}^{i} \lambda^{i-j} \left(d_k(j) - u_{k,j} w\right)^2 + \lambda^i \delta \|w\|_2^2 \tag{11}$$

where $0 \ll \lambda \leq 1$ is a forgetting factor, putting more weight on more recent observations. The solution of this problem is again given by (4), but the estimates $\hat{R}_{u_k,i}$ and $\hat{r}_{u_k d_k,i}$ are now given by

$$\hat{R}_{u_k,i} = \sum_{j=1}^{i} \lambda^{i-j} u_{k,j}^H u_{k,j} + \delta I_M \tag{12}$$

$$\hat{r}_{u_k d_k,i} = \sum_{j=1}^{i} \lambda^{i-j} u_{k,j}^H d_k(j) . \tag{13}$$

It is noted that the effective window length is equal to $\frac{1}{1-\lambda} = \sum_{j=0}^{\infty} \lambda^j$, and since there is no normalization for the window length, $\hat{R}_{u_k,i}$ and $\hat{r}_{u_k d_k,i}$ can be considered to be estimates of $\frac{1}{1-\lambda} R_{u_k,i}$ and $\frac{1}{1-\lambda} r_{u_k d_k,i}$, respectively [10].

The solution of (11) is recursively computed by means of the recursive least squares (RLS) algorithm [10]:

$$P_{k,i} = \lambda^{-1} \left(P_{k,i-1} - \frac{\lambda^{-1} P_{k,i-1} u_{k,i}^H u_{k,i} P_{k,i-1}}{1 + \lambda^{-1} u_{k,i} P_{k,i-1} u_{k,i}^H} \right) \tag{14}$$

$$\hat{w}_{k,i} = \hat{w}_{k,i-1} + P_{k,i} u_{k,i}^H \left(d_k(i) - u_{k,i} \hat{w}_{k,i-1}\right) \tag{15}$$

with $\hat{w}_{k,0} = 0$ and $P_{k,0} = \delta^{-1} I_M$. The matrix $P_{k,i}$ is always equal to $\hat{R}_{u_k,i}^{-1}$ as defined in (12).

Using this construction, (10) is transformed into the recursion

$$\psi_{k,i} = \hat{w}_{k,i} + \frac{1}{1-\lambda} P_{k,i} \left(\hat{R}_{n_k} \psi_{k,i-1} - \hat{r}_{n_k v_k} \right) . \tag{16}$$

The factor $\frac{1}{1-\lambda}$ scales $\hat{R}_{n_k}$ and $\hat{r}_{n_k v_k}$ to match with the effective window length in (12)-(13). We will refer to the above algorithm as bias-compensated RLS (BC-RLS). It is noted that (16) reduces to the BC-LS recursion (10) if $\lambda = 1$ and if the scaling factor $\frac{1}{1-\lambda}$ in (16) is omitted. We do not provide a convergence analysis of BC-RLS here, since it is a special case of the diffusion BC-RLS algorithm when cooperation is turned off, as described in the sequel.

3. DIFFUSION BC-RLS

In a sensor network, each node k has its own node-specific BC-RLS estimate of w^o, denoted by $\psi_{k,i}$. However, the bias compensation usually increases the variance of the estimator in each node. It is to be expected that the spatial average of all the $\psi_{k,i}$'s will be a better estimate for w^o, with a smaller variance. This average could in principle be computed in a distributed fashion by iterative consensus averaging algorithms [12]. The main idea of these algorithms is to collect the estimates $\{\psi_{l,i}\}$ at the neighbors of node k at time i and to iterate over them repeatedly by computing a weighted average of the form:

$$\psi_{k,i} \leftarrow \sum_{l \in \mathcal{N}_k} a_{kl} \psi_{l,i} \tag{17}$$

where $\mathcal{N}_k$ denotes the set of neighboring nodes of node k (node k included), and a_{kl} is the entry at row k and column l of an $N \times N$

combiner matrix[4] A, where A satisfies

$$A\mathbb{1} = \mathbb{1} \tag{18}$$

with $\mathbb{1} = [1 \dots 1]^H$ and where $a_{kl} = 0$ if $l \notin \mathcal{N}_k$. After convergence, the result of (17) becomes the actual estimate $\psi_{k,i}$ by node k at time i. Thus, observe that at every time instant i, multiple consensus iterations can be applied to the data $\{\psi_{l,i}\}$ to approximate their mean and obtain an improved $\psi_{k,i}$.

Applying consensus averaging in the case of BC-RLS would therefore result in a 2-step approach involving two time-scales: one over i and another scale between successive i's. First, the nodes estimate a local $\psi_{k,i}$ based on (16), after which an average consensus algorithm is started to iteratively obtain

$$\psi_i = \frac{1}{N}\sum_{l=1}^{N}\psi_{l,i} \tag{19}$$

at each node. However, this approach is impractical in real-time systems with large sampling rates since the consensus averaging at each iteration i requires a large amount of communication bandwidth and processing power. By applying diffusion strategies (see, e.g., [5]), the iterations of the consensus averaging are merged with those of the BC-RLS algorithm. As a result, the computational complexity and communication bandwidth are significantly reduced while the network is still endowed with learning and tracking abilities. The following table summarizes the diffusion BC-RLS (diffBC-RLS) algorithm that would result from a diffusion strategy. Observe how the left-hand side of (23) is a new variable $w_{k,i}$, which then enters into the update (22). In contrast, in a consensus implementation (apart from the second time-scale), the variables that appear on both sides of (17) are the same ψ variables. In (22)-(23), a filtering operation is embedded into (22) to map $w_{k,i-1}$ to $\psi_{k,i}$ at each node and all $\psi_{l,i}$ are then combined into $w_{k,i}$ in (23).

Diffusion BC-RLS algorithm

Start with $w_{k,0} = 0$, $\hat{w}_{k,0} = 0$ and $P_{k,0} = \delta^{-1} I_M$ for each node $k \in \mathcal{J}$. For every time instant $i > 0$, repeat

1. **RLS update**: for every node $k \in \mathcal{J}$, repeat

$$P_{k,i} = \lambda^{-1}\left(P_{k,i-1} - \frac{\lambda^{-1}P_{k,i-1}u_{k,i}^H u_{k,i}P_{k,i-1}}{1+\lambda^{-1}u_{k,i}P_{k,i-1}u_{k,i}^H}\right) \tag{20}$$

$$\hat{w}_{k,i} = \hat{w}_{k,i-1} + P_{k,i}u_{k,i}^H\left(d_k(i) - u_{k,i}\hat{w}_{k,i-1}\right) . \tag{21}$$

2. **Bias correction update**: for every node $k \in \mathcal{J}$, repeat

$$\psi_{k,i} = \hat{w}_{k,i} + \frac{1}{1-\lambda}P_{k,i}\left(\hat{R}_{n_k}w_{k,i-1} - \hat{r}_{n_k v_k}\right) . \tag{22}$$

3. **Spatial update**: for every node $k \in \mathcal{J}$, repeat

$$w_{k,i} = \sum_{l\in\mathcal{N}_k} a_{kl}\psi_{l,i} \tag{23}$$

It is noted that the RLS estimates $\hat{w}_{k,i}$ are not diffused, since their variance is small compared to the bias-compensated estimates $\psi_{k,i}$.

[4]This combiner matrix has to satisfy some constraints to let the consensus averaging algorithm converge [12]. However, since the diffusion BC-RLS algorithm, as derived in the sequel, does not have the same conditions, we omit them here.

4. MEAN AND MEAN-SQUARE PERFORMANCE

In this section, we address the steady-state performance of the BC-RLS and diffBC-RLS algorithm described in Section 3. We provide a closed-form expression for the mean-square deviation (MSD), i.e.

$$\text{MSD}_k = \lim_{i\to\infty} E\{\|\tilde{w}_{k,i}\|^2\} \tag{24}$$

where

$$\tilde{w}_{k,i} = w^o - w_{k,i} . \tag{25}$$

Similar to [10], $\tilde{w}_{k,i}$ is treated here as a random variable, although it is deterministic. We also provide a closed form expression for the resulting bias if there are estimation errors in $\hat{R}_{n_k}$ and $\hat{r}_{n_k v_k}$. It is noted that all results of diffBC-RLS also apply to the undiffused BC-RLS algorithm (16), by choosing the combiner matrix A equal to the identity matrix. Due to space constraints, we only list the most relevant results of the steady-state analysis, without the full derivation. The latter can be found in [9].

4.1 Data Model and Extra Notation

The performance analysis of adaptive filters is challenging [10], and it is common to adopt some simplifying assumptions to gain insight in the properties of these algorithms. It is noted that, even in cases where these assumptions are not perfectly satisfied, the obtained formulas are still useful to analyze the influence of different parameters. Even though the exact MSD's are not provided, the formulas usually reflect the correct trends when parameters are varied.

Assumption 1: The regressors $\overline{u}_{k,i}$ and the additive noise components $n_{k,i}$ are both zero-mean and temporally independent. Furthermore, the covariance matrix $R_{\overline{u}_{k,i}} = E\{\overline{\mathbf{u}}_{k,i}^H\overline{\mathbf{u}}_{k,i}\}$ is time invariant, i.e., $R_{\overline{u}_{k,i}} = R_{\overline{u}_k}$, $\forall\, i \in \mathbb{N}$. We will therefore often omit the index i in the sequel, when referring to random processes.

Assumption 2: All data is spatially uncorrelated, i.e., for $k \neq l : E\{\mathbf{u}_k^H\mathbf{u}_l\} = 0$, $E\{\overline{\mathbf{u}}_k^H\overline{\mathbf{u}}_l\} = 0$, $E\{\mathbf{n}_k^*\mathbf{n}_l\} = 0$, $E\{\mathbf{v}_k^*\mathbf{v}_l\} = 0$, $E\{\mathbf{v}_k^*\mathbf{n}_l\} = 0$ and $E\{\mathbf{u}_k^H\mathbf{d}_l\} = 0$.

Since we only address the steady state of the diffBC-RLS algorithm, we investigate the steady-state behavior of the matrix $P_{k,i}$. As $i \to \infty$, we find from (12), and the fact that $P_{k,i}^{-1} = \hat{R}_{u_k,i}$, that

$$\lim_{i\to\infty} E\{P_{k,i}^{-1}\} = \frac{1}{1-\lambda}R_{u_k} \triangleq P_k^{-1} . \tag{26}$$

The following two assumptions are made to make the analysis of diffBC-RLS tractable, and both of them are common in the analysis of RLS-type algorithms (see for example [10]).

Assumption 3: $\exists\, i_0$ such that for all $i > i_0$, $P_{k,i}$ and $P_{k,i}^{-1}$ can be replaced with their expected values, i.e.

$$P_{k,i} \approx E\{P_{k,i}\} \tag{27}$$

$$P_{k,i}^{-1} \approx E\{P_{k,i}^{-1}\} . \tag{28}$$

Assumption 4: $\exists\, i_0$ such that for all $i > i_0$:

$$E\{P_{k,i}\} \approx E\{P_{k,i}^{-1}\}^{-1} = P_k = (1-\lambda)R_{u_k}^{-1} . \tag{29}$$

The last assumption is a coarse approximation, since the expected values $E\{P_{k,i}^{-1}\}$ and $E\{P_{k,i}\}$ do not necessarily share the same inverse relation as their arguments. However, for λ close to unity and a not too large condition number for R_{u_k}, this is a good approximation [7, 10].

For the sake of an easy exposition, we define the following notation, based on stacked variables from all nodes. Let

$$\begin{array}{lcll} \tilde{\mathcal{w}}_i & = & \text{col}\{\tilde{w}_{1,i},\dots,\tilde{w}_{N,i}\} & (MN\times 1)\\ r_{nv} & = & \text{col}\{r_{n_1v_1},\dots,r_{n_Nv_N}\} & (MN\times 1)\\ \mathcal{A} & = & A\otimes I_M & (MN\times MN)\\ \mathcal{R}_n & = & \text{blockdiag}\{R_{n_1},\dots,R_{n_N}\} & (MN\times MN)\\ \mathcal{R}_u & = & \text{blockdiag}\{R_{u_1},\dots,R_{u_N}\} & (MN\times MN)\end{array}$$

where $\otimes$ denotes a Kronecker product. All the derived quantities (such as $\hat{r}_{nv}$, $\hat{R}_n$, etc.) have a similar notation for the stacked case, but are omitted for conciseness.

4.2 Stability in the Mean

Under the above assumptions, it can be shown [9] that $\lim_{i\to\infty} E\{\tilde{w}_i\} = 0$ (stability in the mean) if, and only if,

$$\boxed{\rho\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) < 1} \tag{30}$$

where $\rho(X)$ denotes the spectral radius of the matrix X, i.e., the magnitude of the eigenvalue of X with largest absolute value.

Note that setting $\mathcal{A} = I_{MN}$ yields the stability condition for the BC-RLS algorithm (16) without diffusion. It is not possible to make general statements whether diffusion ($\mathcal{A} \neq I_{MN}$) will increase the stability of the algorithm, since this depends on the space-time data statistics (represented by $\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n$) and the network topology (represented by $\mathcal{A}$). However, since $\mathcal{A}$ has a unity spectral radius, it often has a 'non-expanding' effect. For some particular cases, it can be mathematically verified that the stability indeed increases (i.e. the spectral radius in (30) decreases), and we refer to Subsection 4.5 for some examples.

4.3 Asymptotic Bias

In the derivation of the asymptotic bias of the diffBC-RLS algorithm, we incorporate possible estimation errors on the noise covariances, say,

$$\hat{R}_{n_k} = R_{n_k} + \Delta R_{n_k} \tag{31}$$

$$\hat{r}_{n_k v_k} = r_{n_k v_k} + \Delta r_{n_k v_k} \ . \tag{32}$$

It can then be shown [9] that, if the stability condition (30) holds and if the assumptions in Subsection 4.1 are satisfied, the asymptotic bias of the diffBC-RLS algorithm is equal to

$$\boxed{E\{\tilde{w}_i\} = \left(I_{MN} - \mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)^{-1}\mathcal{A}\mathcal{R}_u^{-1}\left(\Delta r_{nv} - \Delta\mathcal{R}_n w^o\right).} \tag{33}$$

A first important observation is that the (diff)BC-RLS algorithm is asymptotically unbiased if $\Delta\mathcal{R}_n = 0$ and $\Delta r_{nv} = 0$, i.e., if there is perfect knowledge of the noise covariance. The smaller the error in $\hat{\mathcal{R}}_n$ and $\hat{r}_{nv}$, the smaller the resulting bias.

It is again not possible to make general statements whether diffusion ($\mathcal{A} \neq I_{MN}$) will decrease the bias of the algorithm. However, if the stability increases, which is often the case, this usually also yields a smaller bias. To see this, observe that $\rho\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) \leq \rho\left(\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)$ implies that

$$\rho\left(\left(I_{MN} - \mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)^{-1}\right) \leq \rho\left(\left(I_{MN} - \mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right)^{-1}\right) \ . \tag{34}$$

This implies that a mapping based on the lefthand side of (34) is 'more contractive' or 'less expanding' than the mapping on the righthand side (corresponding to the undiffused case). Therefore, the bias given in (33) with $\mathcal{A} \neq I_{MN}$ is often (but not necessarily) smaller than the undiffused case ($\mathcal{A} = I_{MN}$). Note that, if diffusion is applied, there is an additional effect, namely an averaging operator $\mathcal{A}$ applied to the error vector $\mathcal{R}_u^{-1}\left(\Delta r_{nv} - \Delta\mathcal{R}_n w^o\right)$. If the combiner matrix A is symmetric, this is a non-expanding mapping, i.e. $\|\mathcal{A}x\| \leq \|x\|$ for all x.

4.4 Mean-square Performance

To define the closed-form expression of the MSD of the diffBC-RLS algorithm, we need to define some extra variables. Consider the eigenvalue decomposition $\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n = Q\Sigma Q^{-1}$ where Σ is a diagonal matrix with the eigenvalues as its diagonal elements, and where Q contains the corresponding normalized eigenvectors in its columns. We also define the (MN)-dimensional vector η containing the diagonal elements of Σ^H (the conjugated eigenvalues) in the same order as they appear on the diagonal. Let

$$\mathcal{M}_k = \mathcal{A}^H\left(\mathcal{M}_{k,2} + \mathcal{M}_{k,2}^H - \mathcal{M}_{k,1}\right)\mathcal{A} \tag{35}$$

with

$$\mathcal{M}_{k,1} = Q^{-H}\left(\frac{Q^H\mathcal{E}_k Q}{\mathbb{1}\mathbb{1}^H - \eta\eta^H}\right)Q^{-1} \tag{36}$$

$$\mathcal{M}_{k,2} = Q^{-H}\left(I_{MN} - \lambda\Sigma^H\right)^{-1}\left(\frac{Q^H\mathcal{E}_k Q}{\mathbb{1}\mathbb{1}^H - \eta\eta^H}\right)Q^{-1} \tag{37}$$

where the double-lined fraction denotes an elementwise division of the matrices in the numerator and denominator (i.e. a Hadamard quotient), and where $\mathcal{E}_k = E_k \otimes I_M$ with E_k denoting an $N \times N$ matrix with zero-valued entries, except for a one on the k-th diagonal entry. The matrix $\mathcal{E}_k$ serves as a selector matrix to select the part of $\tilde{w}_i$ corresponding to the k-th node.

If $\Delta\mathcal{R}_n = 0$ and $\Delta r_{nv} = 0$, and if the stability condition (30) holds, together with the assumptions in Subsection 4.1, it can be shown [9] that the MSD at node k is equal to

$$\boxed{\text{MSD}_k = \tfrac{1-\lambda}{2}\text{Tr}\left(\mathcal{M}_k\mathcal{V}\mathcal{R}_u^{-1}\right)} \tag{38}$$

where

$$\mathcal{V} = \text{diag}\{\sigma_1^2, \ldots, \sigma_N^2\} \otimes I_M \tag{39}$$

$$\sigma_k^2 = w^{o\,H}b_k - r_{n_k v_k}^H w^o + \sigma_{v_k}^2 - b_k^H R_{u_k}^{-1} b_k \tag{40}$$

$$b_k = R_{n_k} w^o - r_{n_k v_k} \ . \tag{41}$$

It is noted that only the matrix $\mathcal{M}_k$ depends on the combiner matrix A, since it is incorporated in the eigenvalue decomposition of $\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n$. Note that $\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n$ is the same matrix that appears in the stability condition (30). Note also that, if (30) holds, the denominators in (36) and (37) cannot become zero and $\left(I_{MN} - \lambda\Sigma^H\right)$ cannot become singular, i.e., the MSD is finite.

Again, it is hard to make general statements about the impact of diffusion on the MSD at a certain node. However, from the Hadamard quotient in (36)-(37), one can expect that the norm of $\mathcal{M}_k$ will be smaller if the norm of η is small. In many cases, setting the matrix $\mathcal{A} \neq I_{MN}$ will decrease the norm of η (although this is not true in general), and then diffusion has a beneficial influence with respect to the MSD of the algorithm. This means that, if diffusion increases the stability (i.e., the spectral radius of $\mathcal{A}\mathcal{R}_u^{-1}\mathcal{R}_n$ decreases), this often also improves the mean-square performance. In Section 4.5, we will consider some special cases where it can indeed be mathematically verified that diffusion decreases the infinity norm of η.

4.5 Special Cases

As explained in the previous subsections, a stability increase due to diffusion usually also yields an improvement in terms of the MSD and the residual bias of the diffBC-RLS algorithm. It is therefore important that

$$\rho\left(\mathcal{A}\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) \leq \rho\left(\mathcal{R}_u^{-1}\hat{\mathcal{R}}_n\right) \ . \tag{42}$$

Without going into detail, it can be shown [9] that (42) indeed holds in the following special cases (assuming a symmetric matrix A):

- *Invariant spatial profile*: This is the case where the regressor and noise covariance matrices are the same in each node (and each node uses the same estimate $\hat{R}_{n_k}$), i.e. $R_{n_k} = R_n$, $R_{u_k} = R_u$, and $\hat{R}_{n_k} = \hat{R}_n$, for $k \in \{1, \ldots, N\}$. Although diffusion has no influence on the stability in this case, the MSD and residual bias can be shown to decrease in general.
- *White noise on regressors*: If $R_{n_k} = \sigma_{n_k}^2 I_M$ and $\hat{R}_{n_k} = \hat{\sigma}_{n_k}^2 I_M$, for $k \in \{1, \ldots, N\}$, it can be shown that (42) indeed holds. Furthermore, if the noise variances $\sigma_{n_k}^2$ are not (too much) overestimated, it can be shown that the diffBC-RLS algorithm is always stable for this case, i.e., (30) is always satisfied.

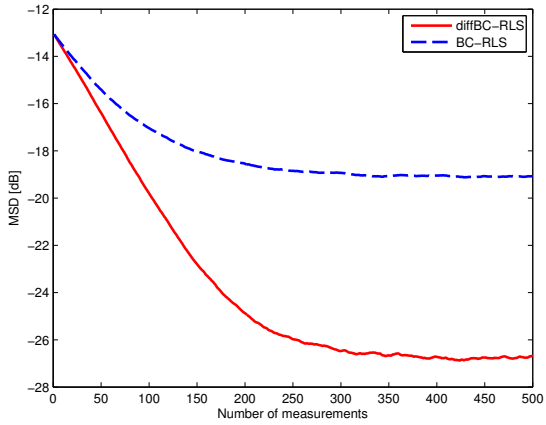

Figure 1: MSD curves of the BC-RLS and diffBC-RLS algorithm with $\lambda = 0.99$.

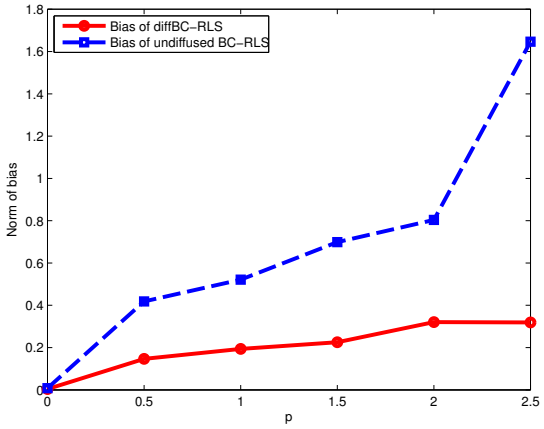

Figure 2: The norm of the stacked asymptotic bias as a function of p ($\lambda = 0.999$).

- *White regressors*: If $R_{\overline{u}_k} = \sigma^2_{\overline{u}_k} I_M$, for $k \in \{1, \ldots, N\}$ and $\hat{\mathcal{R}}_n = \mathcal{R}_n$, i.e., a good estimate of the noise covariance is available, then it can be shown that both (42) and (30) hold.

It is noted that diffusion in general provides better results (with respect to stability, bias and MSD) due to the non-expanding effect of the combiner matrix A. The beneficial influence of diffusion is therefore not limited to the special cases listed above. These merely serve as "motivating" examples where the beneficial influence of diffusion can be theoretically verified. A more general case is simulated in Section 5, where it is demonstrated that diffusion indeed improves overall performance.

5. SIMULATION RESULTS

In this section, we provide simulation results to compare the performance of the BC-RLS and diffBC-RLS algorithm. The measurements $d_k(i)$ were generated according to (1), and the clean regressors $\overline{u}_{k,i}$ were chosen Gaussian i.i.d. with a covariance matrix $R_{\overline{u}_k} = Q_1 \text{diag}\{5, 4, 3, 2, 1\} Q_1^H$, where Q_1 is a random orthogonal matrix. The stacked vectors of the regressor noises and the measurement noises $\overline{n}_{k,i} = [n_{k,i} \; v_k(i)]$ were also chosen Gaussian i.i.d. with a random covariance matrix $E\{\overline{\mathbf{n}}_{k,i}^H \overline{\mathbf{n}}_{k,i}\} = s_k Q_2 \text{diag}\{2, 1.8, 1.6, 1.4, 1.2, 1\} Q_2^H$, where Q_2 is again a random orthogonal matrix, and where s_k is a random scalar drawn from a uniform distribution in the interval $[0.1, 1]$. Note that, due to the scaling with s_k, this does not correspond to a spatial invariant profile, since there is a different SNR in each node. The network had a total of $N = 20$ nodes, and the topology was chosen randomly with a connectivity of 5 links per node on average. The size of the unknown vector w^o was $M = 5$, and the combiner matrix A was constructed using Metropolis weights (see, e.g., [12]). All results are averaged over 200 experiments.

In Fig. 1, the MSD is plotted as a function of the number of measurements, both for BC-RLS (without cooperation) and diffBC-RLS. It is observed that the MSD is significantly lower when the nodes diffuse their estimations.

To see the effect on the bias of BC-RLS and diffBC-RLS, we added some errors to the noise estimates $\hat{R}_{n_k} = R_{n_k} + \Delta R_{n_k}$ and $\hat{r}_{n_k v_k} = r_{n_k v_k} + \Delta r_{n_k v_k}$. The errors were modelled as

$$\Delta R_{n_k} = \sqrt{p|R_{n_k}|} \odot R_k, \quad \Delta r_{n_k v_k} = \sqrt{p|r_{n_k v_k}|} \odot r_k \quad (43)$$

where $\odot$ denotes a Hadamard product (elementwise multiplication), the operator $|.|$ denotes an elementwise absolute value operator, and p is a positive scalar variable that is used to increase the error. The entries of the $M \times M$ matrix R_k and the M-dimensional vector were independently drawn from a normal distribution (i.e. with zero mean and unity variance).

Fig. 2 shows $\|E\{\tilde{w}_i\}\|$, i.e. the norm of the stacked bias, as a function of p, both for BC-RLS (without cooperation) and diffBC-RLS in steady state (with $\lambda = 0.999$). We observe that diffusion indeed significantly decreases the asymptotic bias of the BC-RLS algorithm.

6. CONCLUSIONS

We have described a diffusion-based bias-compensated RLS algorithm for distributed estimation in ad-hoc adaptive networks. We have demonstrated that the variance increase due to the bias compensation can be significantly reduced by letting nodes cooperate by means of diffusion adaptation. Furthermore, diffusion often increases the stability of the algorithm, and reduces the residual bias due to errors in the estimates of the noise statistics. This is demonstrated with theoretical results (omitting the full derivation for conciseness), and by means of Monte-Carlo simulations.

REFERENCES

[1] C. E. Davila, "An efficient recursive total least squares algorithm for FIR adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 42, pp. 267–280, 1994.

[2] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *Accepted for publication in IEEE Transactions on Signal Processing*, 2011.

[3] S. Sagara and K. Wada, "On-line modified least-squares parameter estimation of linear discrete dynamic systems," *International Journal of Control*, vol. 25, pp. 329–343, 1977.

[4] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, vol. 3, pp. III–917 –III–920, 2007.

[5] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. on Signal Proc.*, vol. 56, pp. 3122–3136, July 2008.

[6] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. on Signal Proc.*, vol. 58, pp. 1035–1048, March 2010.

[7] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. on Signal Proc.*, vol. 56, pp. 1865–1877, May 2008.

[8] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Performance analysis of the consensus-based distributed LMS algorithm," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 981030, 19 pages, 2009. doi:10.1155/2009/981030.

[9] A. Bertrand, M. Moonen, and A. H. Sayed, "Diffusion bias-compensated RLS estimation over adaptive networks," *submitted for publication*, 2011.

[10] A. H. Sayed, *Adaptive Filters*. NJ: John Wiley & Sons, 2008.

[11] W. X. Zheng, "A least-squares based algorithm for FIR filtering with noisy data," in *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, pp. IV–444 – IV–447 vol.4, May 2003.

[12] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. Int. Symposium on Information Proc. in Sensor Networks (IPSN)*, (Piscataway, NJ, USA), pp. 63–70, IEEE Press, 2005.