Asilomar Conference on Signals, Systems and Computers - Asilomar 2015, November $8 - 11^{th}$, 2015, Pacific Grove, CA

Sparsity-Aware Distributed Conjugate Gradient Algorithms for Parameter Estimation over Sensor Networks

Tamara Guerra Miller[1], Songcen Xu[2], Rodrigo C. de Lamare[1,2], Vítor H. Nascimento[3] and Yuriy Zakharov[2]
[1]CETUC, PUC-Rio, Brazil
[2]Department of Electronics, University of York, United Kingdom
[3] Department of Electronic Systems, University of São Paulo, Brazil

Abstract— **This paper proposes distributed adaptive algorithms based on the conjugate gradient (CG) method and the diffusion strategy for parameter estimation over sensor networks. We present sparsity-aware conventional and modified distributed CG algorithms using l_1 and log-sum penalty functions. The proposed sparsity-aware diffusion distributed CG algorithms have an improved performance in terms of mean square deviation (MSD) and convergence as compared with the consensus least-mean square (Diffusion-LMS) algorithm, the diffusion CG algorithms and a close performance to the diffusion distributed recursive least squares (Consensus-RLS) algorithm. Numerical results show that the proposed algorithms are reliable and can be applied in several scenarios.**

Keywords— **Distributed Processing, Diffusion Strategy, Conjugate Gradient, Sparsity Aware.**

I. Introduction

Distributed processing has become a very common and useful approach to extract information in a network by performing estimation of the desired parameters. The efficiency of the network depends on the communication protocol used to exchange information between the nodes, as well as the algorithm to obtain the parameters. Another important aspect is to prevent a failure in any agent that may affect the operation and the performance of the network. Distributed schemes can offer better estimation performance of the parameters as compared with the centralized approach, based on the principle that each node communicates with several other nodes and exploits the spatial diversity in the network [1].

The main strategies for communication in distributed processing are incremental, consensus and diffusion. In the incremental protocol, the communication flows cyclically and the information is exchanged from one node to the adjacent nodes. In this strategy the flow of information must be preset at the initialization [2]. The consensus strategy is an elegant procedure to enforce agreement among cooperating nodes [3]. In the diffusion mechanism, each node communicates with the rest of the nodes [4] without any enforcement constraint.

In many scenarios, the parameters of unknown systems can be assumed sparse, containing only a few large coefficients interspersed among many negligible ones [5]. Many studies have shown that exploiting the sparsity of a system is beneficial to enhancing the estimation performance [6]. Most of the studies developed for distributed processing exploiting sparsity focus on the least-mean square (LMS) and recursive least-squares (RLS) algorithms using different penalty functions [7]-[10]. These penalty functions perform a regularization that attracts to zero the coefficients of the parameter vector that are not associated with the weights of interest. The most well-known and exploited penalty functions are the l_0-norm, the l_1-norm and the log-sum [10].

The conjugate gradient (CG) algorithm has been studied and developed for distributed processing [12][23]. The faster learning of CG algorithms over the LMS algorithm and its lower computational complexity combined with better numerical stability than the RLS algorithm makes it suitable for this task. However, prior work on distributed CG techniques is rather limited and techniques that exploit possible sparsity of the signals have not been developed so far.

In this paper we propose distributed CG algorithms based on two variants of the diffusion strategy for parameter estimation over sensor networks. Specifically, we develop standard and sparsity-aware distributed CG algorithms using the diffusion protocol and the l_1 and log-sum penalty functions. The proposed algorithms are compared with recently reported algorithms in the literature. The application scenario in this work is parameter estimation over sensor networks, which can be found in many scenarios of practical interest.

This paper is organized as follows. Section II describes the system model and the problem statement. Section III presents the proposed distributed CG algorithm conventional and modified versions. Section IV details the proposed sparsity-aware distributed diffusion CG algorithm. Section V presents and discusses the simulation results. Finally, Section VI gives the conclusions.

II. System Model and Problem Statement

A. System Model

The network consists of N nodes that exchange information between them, where each node represents an adaptive parameter vector with neighborhood described by the set N_k. The main task of parameter estimation is to adjust the unknown $M\times1$ weight vector $\boldsymbol{\omega}_k$ of each node, where M is the length of the filter [3]. The desired signal $d_{k,i}$ at each time i is drawn from a random process and given by

$$d_{k,i} = \boldsymbol{\omega}_0^H \boldsymbol{x}_{k,i} + n_{k,i} \tag{1}$$

where $\boldsymbol{\omega}_0$ is the $M\times1$ system weight vector, $\boldsymbol{x}_{k,i}$ is the $M\times1$ input signal vector and $n_{k,i}$ is the measurement noise. The output estimate is given by

$$y_{k,i} = \boldsymbol{\omega}_{k,i}^H \boldsymbol{x}_{k,i} \tag{2}$$

The main goal of the network is to minimize the following cost function:

$$C(\boldsymbol{\omega}_{k,i}) = \sum_{k=1}^{N} E[|d_{k,i} - \boldsymbol{\omega}_{k,i}^H \boldsymbol{x}_{k,i}|^2] \tag{3}$$

By solving this minimization problem it is possible to obtain the optimum solution of the weight vector at each node. The optimum solution for the cost function is given by

$$\boldsymbol{\omega}_{k,i} = \boldsymbol{R}_{k,i}^{-1}\boldsymbol{b}_{k,i} \tag{4}$$

where $\boldsymbol{R}_{k,i} = E[\boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H]$ is the $M \times M$ correlation matrix of the input data vector $\boldsymbol{x}_{k,i}$, and $\boldsymbol{b}_{k,i} = E[d_{k,i}^*\boldsymbol{x}_{k,i}]$ is the $M \times 1$ cross-correlation vector between the input data and $d_{k,i}$.

B. Problem Statement

We consider a diffusion algorithm for a network where each agent k has access at each time instant to the realization $\{d_{k,i}, \boldsymbol{x}_{k,i}\}$ of zero-mean spatial data $\{d_{k,i}, \boldsymbol{x}_{k,i}\}$ [12]-[16]. For a network with possibly sparse parameter vectors, the cost function also involves a penalty function which exploits sparsity. In this case the network needs to solve the following optimization problem:

$$\min\, C(\boldsymbol{\omega}_{k,i}) = \sum\nolimits_{k=1}^{N} E[|d_{k,i} - \boldsymbol{\omega}_{k,i}^H\boldsymbol{x}_{k,i}|^2] + f(\boldsymbol{\omega}_{k,i}) \tag{5}$$

where $f(\boldsymbol{\omega}_{k,i})$ is a penalty function that exploits the sparsity in the parameter vector $\boldsymbol{\omega}_{k,i}$. In the following sections we focus on distributed diffusion CG algorithms to solve (5)

III. Proposed Distributed Diffusion CG Algorithm

In this section, we present the proposed distributed CG algorithm using the diffusion strategy with a penalty function that is equal to zero. This corresponds to the diffusion strategy without the exploitation of sparsity. We first derive the CG algorithm and then consider the diffusion protocol.

A. Derivation of the CG algorithm

The CG method can be applied to adaptive filtering problems [11][12][17]. The main objective in this task is to solve (4). The cost function for one agent is given by

$$C_{CG}(\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^H\boldsymbol{R}\omega - \boldsymbol{b}^H\boldsymbol{\omega} \tag{6}$$

For distributed processing over sensor networks, we present the following derivation. The CG algorithm does not need to compute the matrix inversion of $\boldsymbol{R}$, which is an advantage as compared with RLS algorithms. It computes the weights $\boldsymbol{\omega}_{k,i}$ for each iteration j until convergence, i.e., $\boldsymbol{\omega}_{k,i}(j)$. The gradient of the method in the negative direction is obtained as follows [11]:

$$\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) - \boldsymbol{R}_{k,i}(j)\boldsymbol{\omega}_{k,i}(j) \tag{7}$$

Calculating the Krylov subspace [13] through different operations, the recursion is given by

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j) \tag{8}$$

where $\boldsymbol{p}$ is the conjugate direction vector of $\boldsymbol{g}$ and α is the step size that minimizes the cost function in (6) by replacing (7) in (4). Both parameters are calculated as follows:

$$\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)} \tag{9}$$

$$\boldsymbol{p}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j) \tag{10}$$

The parameter β is calculated using the Gram-Schmidt orthogonalization procedure [14] as given by

$$\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)} \tag{11}$$

Applying the CG method to a distributed network the cost function is expressed based on the information exchanged between all nodes $k = 1, 2..., N$. Each of the equations presented so far takes place at each agent during the iterations of the CG algorithm. Therefore, we have the cost function:

$$C_{CG}(\boldsymbol{\omega}_{k,i}) = \frac{1}{2}\sum_{k=1}^{N}\boldsymbol{\omega}_{k,i}^H\boldsymbol{R}_{k,i}\boldsymbol{\omega}_{k,i} - \boldsymbol{b}_{k,i}^H\boldsymbol{\omega}_{k,i} \tag{12}$$

Using the data window with an exponential decay, the resulting autocorrelation matrix and cross-correlation vector are defined using the forgetting factor λ as given by

$$\boldsymbol{R}_{k,i} = \lambda\boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H \tag{13}$$

$$\boldsymbol{b}_{k,i} = \lambda\boldsymbol{b}_{k,i-1} + d_{k,i}^*\boldsymbol{x}_{k,i} \tag{14}$$

In the diffusion strategy, all nodes interact with their neighbors sharing and updating the system parameter vector. Each node k is able to run its update simultaneously with the other agents [1] [4]. Figure 1 illustrates the diffusion strategy.

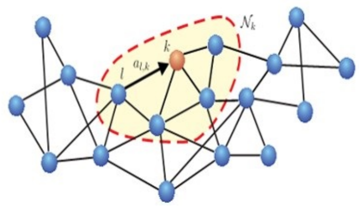

Fig. 1. Distributed consensus-based network processing.

In diffusion protocols there are two well-known variants that switch the order of the combination and adaptation steps, namely, Combine-then-Adapt (CTA) and Adapt-then-Combine (ATC), each one based on the connectivity among nodes. These mechanisms perform adaptation and learning at the same time [3][4].

B. CTA Diffusion Distributed CG algorithm

In the CTA diffusion strategy, the convex combination term is first evaluated into an intermediate state variable and subsequently used to perform the weight update [4]. The local estimation is given by

$$\boldsymbol{\varphi}_{k,i} = \sum_{l \epsilon N_k} a_{lk}\boldsymbol{\omega}_{l,i-1} \tag{15}$$

where a_{lk} represents the combining coefficients of the data fusion which should comply with

$$\sum_{l \epsilon N_k} a_{lk} = 1, l \epsilon N_{k,i}, \forall k. \tag{16}$$

In this work the strategy adopted for the a_{lk} combiner is the Metropolis rule [1] given by

$$c_{kl} = \begin{cases} \frac{1}{max\{|\mathcal{N}_k|,|\mathcal{N}_l|\}}, & \text{if } k \neq l \text{ are linked} \\ 1 - \sum\limits_{l \in \mathcal{N}_k / k} c_{kl}, & \text{for } k = l, \end{cases} \tag{17}$$

TABLE I
CTA CG ALGORITHM

```
Parameters initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$
For each time instant $i > 0$
        For each agent $k$=1,2, …, N
                $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
                $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^{*}\boldsymbol{x}_{k,i}$
                $\boldsymbol{\varphi}_{k,i} = \sum_{l \epsilon N_k} a_{lk}\omega_{l,i-1}$
                $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\varphi}_{k,i}$
                $\boldsymbol{g}k, i(0) = \boldsymbol{b}_{k,i}(0) - \boldsymbol{R}_{k,i}(0)\boldsymbol{\omega}_{k,i-1}$
                $\boldsymbol{p}k, i(0) = \boldsymbol{g}_{k,i}(0)$
                For each CG iteration $j = 1$ until convergence
                        $\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}$
                        $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j)$
                        $\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) - \alpha(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j-1)$
                        $\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}$
                        $\boldsymbol{p}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j-1)$
                End For
                $\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k,i}(j_{last})$
        End for
End for
```

The distributed CTA CG algorithm based on the derivation steps obtains the updated weight substituting (15) in (8), giving as result:

$$\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j) \tag{18}$$

where $\boldsymbol{\omega}_{k,i}(0) = \boldsymbol{\varphi}_{k,i}$ [12]. The rest of the derivation is based on the solution presented in the previous section and the pseudo-code is detailed in Table I

The modified CG (MCG) algorithm comes from the conventional CG algorithm previously presented and only requires one iteration per coefficient update. Specifically, the residual is calculated using (7) (8) and (13) [11]:

$$\begin{aligned}\boldsymbol{g}_{k,i} &= \boldsymbol{b}_{k,i} - \boldsymbol{R}_{k,i}\boldsymbol{\varphi}_{k,i}\\ &= \lambda \boldsymbol{g}_{k,i-1} - \alpha_{k,i}\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1}\\ &+\boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H\boldsymbol{x}_{k,i}]\end{aligned} \tag{19}$$

The previous equation (19) is multiplied for the search direction vector:

$$\begin{aligned}\boldsymbol{p}_{k,i}^H\boldsymbol{g}_{k,i} &= \lambda \boldsymbol{p}_{k,i}^H\boldsymbol{g}_{k,i-1} - \alpha_{k,i}\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1}\\ &+\boldsymbol{p}_{k,i}^H\boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H\boldsymbol{x}_{k,i}]\end{aligned} \tag{20}$$

Applying the expected value, considering $\boldsymbol{p}_{k,i-1}$ uncorrelated with $\boldsymbol{x}_{k,i}$, $d_{k,i}$ and $\boldsymbol{\varphi}_{k,i}$, and that the algorithm converges the last term of (20) can be neglected. The line search to compute α has to satisfy the convergence bound [11] given by

$$(\lambda_f - 0.5)\frac{E[\boldsymbol{p}_{k,i-1}^H\boldsymbol{g}_{k,i}]}{E[\boldsymbol{p}_{k,i-1}^H\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1}]} \leq E[\alpha_{k,i}] \leq \frac{E[\boldsymbol{p}_{k,i-1}^H\boldsymbol{g}_{k,i}]}{E[\boldsymbol{p}_{k,i-1}^H\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1}]} \tag{21}$$

$$\alpha_{k,i} = \eta\frac{\boldsymbol{p}_{k,i}^H\boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i}}, \tag{22}$$

where $(\lambda - 0.5) \leq \eta \leq \lambda$. The Polak-Ribiere method [11] for the computation of β is given by

$$\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1})^H\boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i}^H\boldsymbol{g}_{k,i}} \tag{23}$$

TABLE II
ATC CG ALGORITHM

```
Parameters initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$
For each time instant $i > 0$
        For each agent $k$=1,2, …, N
                $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
                $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^{*}\boldsymbol{x}_{k,i}$
                $\boldsymbol{g}k, i(0) = \boldsymbol{b}_{k,i}(0) - \boldsymbol{R}_{k,i}(0)\boldsymbol{\omega}_{k,i-1}(0)$
                $\boldsymbol{p}k, i(0) = \boldsymbol{g}_{k,i}(0)$
                $\boldsymbol{\omega}_{k,i}(0) = \boldsymbol{\omega}_{k,i-1}$
                For each CG iteration $j = 1$ until convergence
                        $\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}$
                        $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j)$
                        $\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) - \alpha(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j-1)$
                        $\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}$
                        $\boldsymbol{p}_{k,i}(j+1) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j)$
                End For
                $\boldsymbol{\omega}_{k,i} = \sum_{l \epsilon N_k} a_{lk}\boldsymbol{\omega}_{k,i}(j_{last})$
        End for
End for
```

TABLE III
ATC MCG ALGORITHM

```
Parameters initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$
For each time instant $i > 0$
        For each agent $k$=1,2, …, N
                $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
                $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^{*}\boldsymbol{x}_{k,i}$
                $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\varphi}_{k,i}$
                $\boldsymbol{g}_{k,1} = \boldsymbol{b}_{k,0}$
                $\boldsymbol{p}_{k,1} = \boldsymbol{g}_{k,1}$
                $\alpha_{k,i} = \eta\frac{\boldsymbol{p}_{k,i}^H\boldsymbol{g}_{k,i}}{\boldsymbol{p}_{k,i}^H\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i}}, (\lambda - 0.5) \leq \eta \leq \lambda$
                $\boldsymbol{\varphi}_{k,i} = \boldsymbol{\omega}_{k,i-1} - \alpha_{k,i}\boldsymbol{p}_{k,i}$
                $\boldsymbol{g}_{k,i} = \lambda \boldsymbol{g}_{k,i} - \alpha_{k,i}\boldsymbol{R}_{k,i}\boldsymbol{p}_{k,i-1}$
                        $+\boldsymbol{x}_{k,i}[d_{k,i} - \boldsymbol{\omega}_{k,i-1}^H\boldsymbol{x}_{k,i}]$
                $\beta_{k,i} = \frac{(\boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1})^H\boldsymbol{g}_{k,i}}{\boldsymbol{g}_{k,i-1}^H\boldsymbol{g}_{k,i-1}}$
                $\boldsymbol{p}_{k,i} = \boldsymbol{g}_{k,i} + \beta_{k,i}\boldsymbol{p}_{k,i-1}$
        End for
        $\boldsymbol{\omega}(i) = \sum_{l \epsilon N_k} a_{lk}\boldsymbol{\varphi}_{l,i}$
End for
```

C. *ATC Distributed CG algorithm*

Similarly to CTA, the ATC protocol switches the order of the operations. The difference lies in the variable chosen to update the weight $\boldsymbol{\omega}_{k,i}$. In this case, the update estimate is the convex combination of the adaptation step. Table II shows the pseudo code of the ATC strategy. The MCG version for the ATC strategy is very similar to the CTA version presented. Table III shows the details of the ATC MCG algorithm taking into account the considerations previously discussed.

IV. Proposed Sparsity-Aware Diffusion CG

Based on the previous development of distributed CG algorithms, this section presents the proposed distributed sparsity-aware diffusion CG algorithms using l_1 (ZA) and *log-sum* (RZA) norm penalty functions.

TABLE IV
SPARSITY-AWARE ATC CG ALGORITHM

```
Parameters initialization:
$\boldsymbol{\omega}_{k,0} = \mathbf{0}$
For each time instant $i > 0$
    For each agent $k$=1,2, …, N
        $\boldsymbol{R}_{k,i} = \lambda \boldsymbol{R}_{k,i-1} + \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^H$
        $\boldsymbol{b}_{k,i} = \lambda \boldsymbol{b}_{k,i-1} + d_{k,i}^* \boldsymbol{x}_{k,i}$
        $\boldsymbol{g}k,i(0) = \boldsymbol{b}_{k,i}(0) - \boldsymbol{R}_{k,i}(0)\boldsymbol{\omega}_{k,i-1}(0)$
        $\boldsymbol{p}k,i(0) = \boldsymbol{g}_{k,i}(0)$
        $\boldsymbol{\omega}_{k,i}(0) = \boldsymbol{\omega}_{k,i-1}$
        For each CG iteration $j = 1$ until convergence
            $\alpha(j) = \frac{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}{\boldsymbol{p}_{k,i}^H(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j)}$
            $\boldsymbol{\omega}_{k,i}(j) = \boldsymbol{\omega}_{k,i}(j-1) - \alpha(j)\boldsymbol{p}_{k,i}(j)$
            $\boldsymbol{g}_{k,i}(j) = \boldsymbol{g}_{k,i}(j) - \alpha(j)\boldsymbol{R}_{k,i}(j)\boldsymbol{p}_{k,i}(j-1)$
            $\beta(j) = \frac{\boldsymbol{g}_{k,i}^H(j)\boldsymbol{g}_{k,i}(j)}{\boldsymbol{g}_{k,i}^H(j-1)\boldsymbol{g}_{k,i}(j-1)}$
            $\boldsymbol{p}_{k,i}(j+1) = \boldsymbol{g}_{k,i}(j) + \beta(j)\boldsymbol{p}_{k,i}(j)$
        End For
        $\boldsymbol{\omega}_{k,i} = \sum_{l \epsilon N_k} a_{lk}\omega_{k,i}(j_{last}) - \rho \frac{\partial(f_{1,2})}{\partial \boldsymbol{\omega}_{k,i}^*}$
    End for
End for
```

A. ZA and RZA CG algorithms

The cost function in this case is given by

$$C_{CG}(\boldsymbol{\omega}_{k,i}) = \frac{1}{2}\sum_{k=1}^{N} \boldsymbol{\omega}_{k,i}^H \boldsymbol{R}\omega_{k,i} - \boldsymbol{b}_{k,i}^H \boldsymbol{\omega}_{k,i} + f_1, \tag{24}$$

where f_1 denotes the l_1 penalty function and is defined by

$$f_1 = \rho\|\boldsymbol{\omega}_{k,i}(j)\|_1. \tag{25}$$

Applying the partial derivative of the penalty function gives

$$\frac{\partial(f_1)}{\partial(\boldsymbol{\omega}_{k,i}^*)} = sgn(\boldsymbol{\omega}_{k,i}) = \begin{cases} \frac{\boldsymbol{\omega}_{k,i}}{|\boldsymbol{\omega}_{k,i}|}, & \text{if } \boldsymbol{\omega}_{k,i} \neq 0 \\ 0, & \text{if } \boldsymbol{\omega}_{k,i} = 0, \end{cases} \tag{26}$$

When the logarithmic penalty function f_2 instead f_1 is used in the cost function, we have

$$f_2 = \rho\sum_{i=1}^{M} \log(1 + \frac{|\omega_{k,i}|}{\varepsilon}). \tag{27}$$

The partial derivative of the penalty function applied with respect to $\boldsymbol{\omega}_{k,i}^*$ is described by

$$\frac{\partial(f_1)}{\partial(\boldsymbol{\omega}_{k,i}^*)} = \frac{sgn(\boldsymbol{\omega}_{k,i})}{1 + \varepsilon\|\boldsymbol{\omega}_{k,i}\|_1} \tag{28}$$

In both cases these sparsity-aware algorithms attract to zero the values of the parameter vector which are very small or are not useful. This results in an algorithm with a faster convergence and lower MSD values as can be seen in the following sections. Using the penalty functions (26) and (28), we obtain the sparsity-aware algorithms with the CTA and ATC strategies. Table IV shows the sparsity-aware method for the ACT protocol. In case of CTA, the same steps applied with the ZA or RZA penalty functions to the steps previously presented in Section III are carried out.

B. ZA and RZA Modified CG algorithms

The ATC and CTA MCG algorithms are very similar as presented in previous section, including the penalty functions. In the ATC strategy, the generation of the first state resulting from the adaptation step, is used in the final update.

C. Computational Complexity

The Table V shows the computational complexity of all diffusion distributed methods proposed in terms of additions and multiplications.

TABLE V
COMPUTATIONAL COMPLEXITY OF DIFFUSION CG ALGORITHMS

Method	Additions	Multiplications
CTA-CG	$L(M^2+2M)$ $+LJ(2M^2+6M-3)$	$L(2M^2+4M)$ $+LJ(3M^2+4M-1)$
ATC-CG	$L(M^2+3M-1)$ $+LJ(M^2+6M-3)$	$L(2M^2+3M)$ $+LJ(3M^2+4M-1)$
CTA-MCG	$L(3M^2+9M-4)$	$L(4M^2+9M-1)$
ATC-MCG	$L(4M^2+9M-3)$	$L(6M^2+8M-1)$
ZA-CTA-CG	$L(M^2+3M)$ $+LJ(2M^2+6M-3)$	$L(2M^2+5M)$ $+LJ(3M^2+4M-1)$
ZA-ATC-CG	$L(M^2+3M)$ $+LJ(2M^2+6M-3)$	$L(2M^2+5M)$ $+LJ(3M^2+4M-1)$
ZA-CTA-MCG	$L(3M^2+10M-4)$	$(4M^2+10M-1)$
ZA-ATC-MCG	$L(4M^2+10M-3)$	$(6M^2+9M-1)$
RZA-CTA-CG	$L(M^2+2M)$ $+LJ(2M^2+8M-3)$	$L(2M^2+4M)$ $+LJ(3M^2+6M-1)$
RZA-ATC-CG	$L(M^2+3M-1)$ $+LJ(2M^2+8M-3)$	$L(2M^2+3M)$ $+LJ(3M^2+6M-1)$
RZA-CTA-MCG	$L(3M^2+11M-4)$	$L(4M^2+11M-1)$
RZA-ATC-MCG	$L(4M^2+11M-3)$	$L(6M^2+10M-1)$

It can be seen that the complexity of the modified versions is lower than the conventional methods

V. SIMULATIONS RESULTS

In this section, we evaluated the proposed distributed diffusion CG algorithms and compare them with existing algorithms. The results are based on the mean square deviation MSD of the network. We consider a network with 20 nodes and 1000 iterations per run. Each iteration corresponds to a time instant. The results are averaged over 100 experiments. The length of the filter is 10 and the variance of the input signal 1, which has been modeled as a complex Gaussian noise and the SNR is 30 dB.

A. Comparison between standard and sparsity-aware CTA distributed CG algorithms

For the standard CTA CG version, the system parameter vector was randomly set. In the case of the sparsity-aware algorithms it was set to two values equal to one and the remaining values were set to zero. After all the iterations, the performance of each algorithm in terms of MSD is shown in Fig. 2. The results show that the sparsity-aware versions outperforms the standard versions and the best results are obtained for the RZA versions. At the same time the MCG algorithms have a better performance than the standard ones.

B. Comparison between sparsity-aware ATC distributed CG algorithms.

The same configuration used before was set for CTA strategy. Fig. 3 below shows the performance of the results in the simulations. Fig. 4 shows the comparison between the consensus and diffusion algorithms with RZA. It can be observed that the diffusion ATC CG algorithm has a faster convergence as compared to the CTA and consensus strategy, as well as the MSD value at steady state.

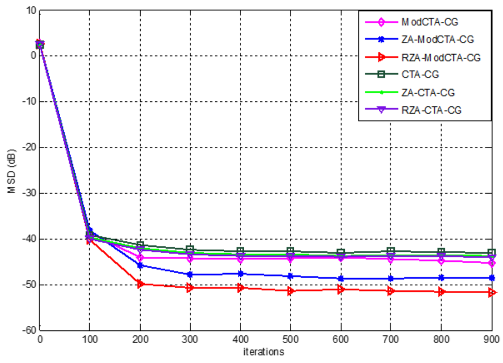

Fig. 2. MSD of the network against the iterations number for distributed CTA standard and sparsity-aware CG versions with $\lambda = 0.99$, $\rho_{RZA} = 0.5{*}10^{-4}$, $\rho_{ZA} = 10^{-3}$, $\varepsilon = 0.1$, $\gamma = 10^{-2}$, $S = 2/10$. Number of CG iterations $J = 5$. For modified versions $\lambda = 0.95$, $\eta = 0.55$,$\rho_{ZA} = 0.7 * 10^{-4}$, $\rho_{RZA} = 0.2 * 10^{-3}$, $\varepsilon = 0.1$, $\gamma = 10^{-2}$, $S = 2/10$.

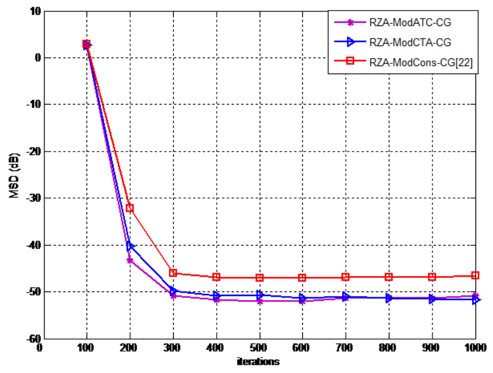

Fig. 4. MSD of the network against the iteration's number for consensus and distributed diffusion RZA CG versions with $\lambda_{Cons} = 0.99$, $\rho_{Cons} = 0.5 * 10^{-3}$, $\varepsilon = 0.1$, $\gamma = 10^{-1}$,$\lambda_{CTA} = 0.99$, $\rho_{CTA} = 0.2 * 10^{-3}$, $\varepsilon = 0.1$, $\gamma = 10^{-2}$,$\lambda_{ATC} = 0.99$, $\rho_{ATC} = 0.3 * 10^{-3}$, $\varepsilon = 0.1$, $\gamma = 10^{-1}$, $\eta = 0.55$, $S = 2/10$. Number of CG iterations $J = 5$.

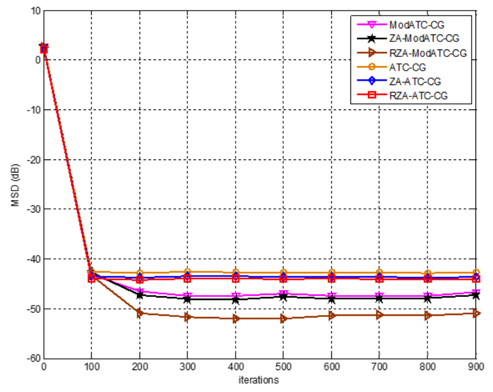

Fig. 3. MSD of the network against the iteration's number for distributed ATC and sparse-aware CG versions with $\lambda = 0.99$, $\rho_{ZA} = 0.5 * 10^{-4}$, $\rho_{RZA} = 9{*}10^{-4}$, $\varepsilon = 0.1$, $\gamma = 10^{-1}$, $S = 2/10$. Number of CG iterations $J = 5$. For modified versions $\lambda = 0.95$, $\eta = 0.55$,$\rho_{ZA} = 0.4 * 10^{-4}$, $\rho_{ZA} = 0.3 * 10^{-3}$, $\varepsilon = 0.1$, $\gamma = 10^{-1}$, $S = 2/10$.

VI. Conclusions

In this work we proposed distributed CG algorithms for parameter estimation over sensor networks as well as the modified versions of them. The proposed ATC diffusion CG algorithms have a faster convergence than the CTA. The ATC strategy outperforms both consensus[ref] and CTA protocols. In all cases, the modified versions obtained the low MSD values and faster convergence rate. Simulations have shown that the proposed distributed CG algorithms are suitable techniques for adaptive parameter estimation problems.

References

[1] S. Yuan T, and A. H. Sayed, *Diffusion Strategies Outperform Consensus Strategies for Distributed Estimation over Adaptive Networks, IEEE Transactions on signal processing*, vol. 60, no. 12, December 2012.

[2] C. G. Lopes and A. H. Sayed, *Incremental adaptive strategies over distributed networks*, IEEE Transactions Signal Process, vol. 48, no. 8, pp.223$-$229, Aug 2007.

[3] M. H. DeGroot, *Reaching a consensus*, Journal of the American Statistical Association, vol. 69, no. 345, pp. 118-121 ,Mar. 1974.

[4] A. H. Sayed, *Adaptation, Learning, and Optimization over Networks*, Foundations and Trends in Machine Learning, vol. 7, no. $4-5$, pp. $311-801$, 2014.

[5] P. D. Lorenzo and A. H. Sayed, *Sparse Distributed Learning Based on Diffusion Adaptation*, IEEE Transactions on signal processing, vol. 61, no. 6, March 15, 2013.

[6] Y. Liu, C. L. and Z. Zhang, *Diffusion Sparse Least-Mean Squares Over Networks*, IEEE Transactions on Signal Processing, vol. 60, no. 8, August 2012.

[7] Y. Chen, Y. Gu, and A. O. Hero, *Sparse LMS for System Identification*, IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, April 2009.

[8] P. D. Lorenzo and S. Barbarossa, *Distributed Least Mean Squares Strategies for Sparsity-Aware Estimation over Gaussian Markov Random Fields*, IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP), May 2014.

[9] E. M. Eksioglu, *RLS Adaptive Filtering with Sparsity Regularization*, 10th International Conference on Information Science, Signal Processing and their Applications ISSPA, may 2010.

[10] R. C. de Lamare and R. Sampaio-Neto, *Sparsity-Aware Adaptive Algorithms Based on Alternating Optimization with shrinkage*, IEEE Signal Processing Letters, vol. 21, no. 2, February 2014.

[11] P. S. Chang and A. N. Willson, Jr., *Analysis of Conjugate Gradient Algorithms for Adaptive Filtering*, IEEE Transactions on Signal Processing, vol. 48, no. 2, February 2000

[12] S. Xu and R.C de Lamare, , *Distributed conjugate gradient strategies for distributed estimation over sensor networks*, Sensor Signal Processing for Defense SSPD, September 2012

[13] O. Axelson, *Iterative Solution Methods*. New York: Cambridge Univ. Press,1994

[14] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd Ed. Baltimore, MD: John's Hopkins Univ. Press,1989

[15] S. Xu, R. C. de Lamare and H. V. Poor, *Distributed Compressed Estimation Based on Compressive Sensing*, IEEE Signal Processing letters, vol. 22, no. 9, September 2014.

[16] S. Xu, R. C. de Lamare and H. V. Poor, "Adaptive Link Selection Algorithms for Distributed Estimation", EURASIP Journal on Advances in Signal Processing, 2015.

[17] S. Wang, H. M. and B. Xi, D. Sun, *Conjugate Gradient-based Parameters Identification*, 8th IEEE International Conference on Control and Automation, China, June 2010.

[18] S. Theodoridis, *Machine Learning a Bayesian and Optimization Perspective*, Academic Press, March 2015.

[19] I. D. Schizas. G. Mateos and G. B. Giannakis, *Distributed LMS for Consensus-Based In-Network Adapting Processing*, IEEE Transactions on Signal Processing, vol. 57, no. 6, June 2009

[20] E. Wei and A. Ozdaglar, *Distributed Alternating Direction Method of Multipliers*,National Science Foundation under Career grant DMI$-$0545910 and AFOSR MURI FA $9550-09-1-0538$.Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[21] X.-L. Feng and T. Z. Huang, *A finite-time consensus protocol of the multi-agent systems*, World Academy of Science, Engineering and Technology Vol: 5 2011-03-24.

[22] N. A. Lynch, *Distributed algorithms*, San Francisco, CA: Morgan Kaufmann, 1997.

[23] T. G. Miller, S. Xu and R.C de Lamare, , *Consensus Distributed Conjugate Gradient Algorithms for Parameter Estimation over Sensor Networks*, XXXIII Brazilian Telecommunications Symposium SBrT2015, September 2015.