Distributed Conjugate Gradient Strategies for Distributed Estimation Over Sensor Networks

Songcen Xu [1], Rodrigo C. de Lamare [2]
Communications Research Group, Department of Electronics, University of York, U.K.
[1] sx520@ohm.york.ac.uk
[2] rcdl500@ohm.york.ac.uk

***Abstract*—This paper presents distributed adaptive algorithms based on the conjugate gradient (CG) method for distributed networks. Both incremental and diffusion adaptive solutions are all considered. The distributed conventional CG (CCG) and modified CG (MCG) algorithms have an improved performance in terms of mean square error as compared with least-mean square (LMS)-based algorithms and a performance that is close to recursive least-squares (RLS) algorithms. The resulting algorithms are distributed, cooperative and able to respond in real time to changes in the environment.**

I. Introduction

In recent years, distributed processing has become popular in wireless communication networks. This kind of processing can collect data at each node in a given area and convey the information to the whole network in a distributed way. For instance, each node can get the information from its neighbors, and then combine it with the use of distributed adaptive algorithms; each node has the ability to estimate the nearby environment itself [1]. When compared with the centralized solution, the distributed solution has significant advantages. The centralized solution needs a central processor, where each node sends its information to the central processor and gets the information back after the processor completes the task. This type of communication needs the central processor to be powerful and reliable. With distributed solutions, each node only requires the local information and its neighbors to process the information. This kind of processing can significantly reduce the amount of processing and the communications bandwidth requirements.

There are three main cooperation modes: the incremental, the diffusion, and the probabilistic diffusion modes [1]. For the incremental mode, we can interpret it as a cycle in which the information goes through the nodes in one direction, which means each node passes the information to its adjacent node in a pre-determined direction. Because of its simple method, the need for communication and power is the least [1]. For the diffusion mode, each node transfers information to all its neighbors. This kind of processing costs a huge amount of communication resources, but each node will get more information. To avoid the high communication cost, another kind of diffusion termed probabilistic diffusion is used. Instead of transferring information to all its neighbors, each node transfers data to a selected group of its neighbors, which can be chosen randomly.

Several algorithms have already been developed and reported in the literature for distributed networks. Steepest-descent, least mean square (LMS) [1] and affine projection (AP) [3] solutions have been considered with incremental adaptive strategies over distributed networks [1], while the LMS and recursive least squares (RLS) algorithms have been reported using diffusion adaptive strategies [2], [4], [5]. Although the LMS-based algorithms have their own advantages, when compared with conjugate gradient (CG) algorithms, their shortages are obvious. First, for the LMS- based algorithms, the adaptation speed is often slow, especially for the conventional LMS algorithm. Second, when we are trying to increase the adaptation speed, the system stability may decrease significantly[11]. Furthermore, the RLS-based algorithms usually have a high complexity. In order to develop a set of distributed solutions with a more attractive tradeoff between performance and complexity, we focus on the CG algorithm. The CG algorithm has a faster convergence rate [6] than the LMS-type algorithms and a lower computational complexity than RLS-type techniques. In this paper, the main contribution is to develop distributed CG algorithms for both incremental and diffusion adaptive strategies. In particular, we develop distributed versions of the conventional CG algorithm and of the modified CG algorithm for use in distributed estimation over sensor networks. These algorithms can be widely used in defence applications, such as battlefield information identification, movement estimation and detection and distributed spectrum estimation.

This paper is organized as follows. Section II describes the system model and states the problem. Section III presents the incremental distributed CG algorithms, whereas Section IV considers the diffusion distributed CG algorithms. Section V presents and discusses the simulation results, whereas Section VI gives the conclusions.

II. System model and problem statement

In this part, we describe a system model of the distributed estimation scheme over sensor networks and introduce the problem statement.

A. System model

The basic idea of this system model is that for each node in a sensor network a designer deals with a system identification problem. Each node is equipped with an adaptive filter. We focus on the processing of an adaptive filter for adjusting the weight vector $\boldsymbol{\omega}_o$ with coefficients $\boldsymbol{\omega}_k$ ($k = 1, \ldots, M$), where M is the length of the filter. The desired signal of each node at time instant i is

$$d^{(i)} = \boldsymbol{\omega}_0^H \boldsymbol{x}^{(i)} + n^{(i)}, \quad i = 1, 2, \ldots, N, \tag{1}$$

where $d^{(i)}$ is the received signal sample, $\boldsymbol{x}^{(i)}$ is the $M \times 1$ input signal vector, $\boldsymbol{\omega}_0$ is the $M \times 1$ system weight vector, $n^{(i)}$ is the noise sample at each receiver, $(\cdot)^H$ denotes Hermitian transpose and N is the number of time instants. At the same time, the output of the adaptive filter for each node is given by

$$y^{(i)} = {\boldsymbol{\omega}^{(i)}}^H \boldsymbol{x}^{(i)}, \quad i = 1, 2, \ldots, N, \tag{2}$$

where $\boldsymbol{\omega}^{(i)}$ is the local estimator $\boldsymbol{\omega}$ for each node at time instant i.

B. Problem Statement

To compute the optimum solution of the weight vector, we need to solve a problem expressed in the form of a minimization of a cost function. Consider a network which has N nodes, each node k has access to time realizations $\{d_k^{(i)}, \boldsymbol{u}_k^{(i)}\}$ of zero- mean spatial data $\{d_k, \boldsymbol{u}_k\}$, k=1,2, …, N, where each d_k is a scalar measurement and each $\boldsymbol{u}_k$ is a row regression vector [1]. After that, two global matrices

are built which are used to collect the measurement data and regression data that are expressed in the form of the matrices:

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ...\boldsymbol{x}_N], \ (N \times M) \tag{3}$$

$$\boldsymbol{d} = [d_1, d_2, \dots d_N]^T, \ (N \times 1) \tag{4}$$

The data associated with these two equations correspond to data collected across all nodes. In order to design an algorithm to compute the optimum estimation value, we need to first define a cost function:

$$J(\boldsymbol{\omega}) = \boldsymbol{E}[||\boldsymbol{d} - \boldsymbol{X}\boldsymbol{\omega}||^2], \tag{5}$$

where the $J(\boldsymbol{\omega})$ is used to calculate the MSE and our aim is to minimize the cost function. The optimal solution should satisfy [1]:

$$E[\boldsymbol{X}^H(\boldsymbol{d} - \boldsymbol{X}\boldsymbol{\omega}_o)] = \boldsymbol{0}. \tag{6}$$

Meanwhile, the $\boldsymbol{\omega}_o$ is also the solution to:

$$\boldsymbol{b} = \boldsymbol{R}\boldsymbol{\omega}_o, \tag{7}$$

where the $M \times M$ autocorrelation matrix is given by $\boldsymbol{R} = E[\boldsymbol{X}^H\boldsymbol{X}]$ and $\boldsymbol{b} = E[\boldsymbol{X}^H\boldsymbol{d}]$ is an $M \times 1$ cross-correlation matrix. In this work, we focus on incremental and diffusion CG-based algorithms to solve the equation and perform estimation in a distributed fashion.

III. Proposed Incremental Distributed CG - Based Algorithms

For distributed estimation over sensor networks, we develop two CG- based algorithms which are the CCG and MCG with incremental distributed solution (IDCG). We derive the CG-based algorithms first, then we devise distributed versions of these algorithm for use in the network in an incremental and distributed way.

A. Derivation of the CG- Based Algorithms

When a CG algorithm is used in adaptive signal processing, it solves the following equation[8]:

$$\boldsymbol{R}_k^{(i)}\boldsymbol{\omega}_k^{(i)} = \boldsymbol{b}_k^{(i)}, \tag{8}$$

where $\boldsymbol{R}_k^{(i)}$ is the $M \times M$ correlation matrix for the input data vector and $\boldsymbol{b}_k^{(i)}$ is the $M \times 1$ cross-correlation vector between the input data vector and d is the desired response. To solve this equation, we need to obtain:

$$\boldsymbol{\omega}_k^{(i)}(j) = [\boldsymbol{R}_k^{(i)}]^{-1}\boldsymbol{b}_k^{(i)}. \tag{9}$$

In the CG- based algorithm, the iteration procedure is introduced. For the jth iteration, we choose the negative direction as:

$$\boldsymbol{g}_k^{(i)}(j) = \boldsymbol{b}_k^{(i)} - \boldsymbol{R}_k^{(i)}\boldsymbol{\omega}. \tag{10}$$

The CG-based weight vector $\boldsymbol{\omega}_k^{(i)}(j)$ is defined as:

$$\boldsymbol{\omega}_k^{(i)}(j) = \boldsymbol{\omega}_{k-1}^{(i)}(j) + \alpha_k^{(i)}(j)\boldsymbol{p}_k^{(i)}(j), \tag{11}$$

where $\boldsymbol{p}(j)$ is the direction vector with conjugacy and $\alpha(j)$ is calculated by replacing (11) in (9), then taking the gradient with respect to $\alpha(j)$ and using (11), we get:

$$\alpha_k^{(i)}(j) = \frac{\rho_k^{(i)}(j-1)}{\boldsymbol{p}_k^{(i)}(j)^H\boldsymbol{c}_k^{(i)}(j)}, \tag{12}$$

where

$$\rho_k^{(i)}(j) = \boldsymbol{g}_k^{(i)}(j)^H\boldsymbol{g}_k^{(i)}(j)$$

and

$$\boldsymbol{c}_k^{(i)}(j) = \boldsymbol{R}_k^{(i)}\boldsymbol{p}_k^{(i)}(j).$$

The direction vector $\boldsymbol{p}_k^{(i)}(j)$ in (11) is defined as:

$$\boldsymbol{p}_k^{(i)}(j+1) = \boldsymbol{g}_k^{(i)}(j) + \beta_k^{(i)}(j)\boldsymbol{p}_k^{(i)}(j), \tag{13}$$

where $\beta_k^{(i)}(j)$ is calculated by the Gram-Schmidt orthogonalization procedure[7] for the conjugacy:

$$\beta_k^{(i)}(j) = \frac{\rho_k^{(i)}(j)}{\rho_k^{(i)}(j-1)} \tag{14}$$

with

$$\rho_k^{(i)}(j) = \boldsymbol{g}_k^{(i)}(j)^H\boldsymbol{g}_k^{(i)}(j).$$

Besides the basic CG algorithm, there are two ways to define the correlation and cross-correlation matrices which are the 'finite sliding data window' and the 'exponentially decaying data window'[8]. In this paper, we mainly focus on the 'exponentially decaying data window' because this approach employs the same correlation matrix as the RLS algorithm. The recursions are given by:

$$\boldsymbol{R}_k^{(i)} = \lambda_f\boldsymbol{R}_{k-1}^{(i)} + \boldsymbol{x}_k^{(i)}[\boldsymbol{x}_k^{(i)}]^H \tag{15}$$

$$\boldsymbol{b}_k^{(i)} = \lambda_f\boldsymbol{b}_{k-1}^{(i)} + d_k^{(i)*}\boldsymbol{x}_k^{(i)} \tag{16}$$

where λ_f is the forgetting factor.

B. Incremental Distributed CG - Based Solutions

In the incremental distributed model of our algorithm, each node is only allowed to communicate with its direct neighbor at each time instant. To describe the whole process, we define a cycle where each node in this network could only access its immediate neighbor in this cycle [1]. The quantity $\boldsymbol{\psi}_k^{(i)}$ is defined as a local estimate of $\boldsymbol{\omega}^o$ at time i. As a result, we assume that node k has access to an estimate of $\boldsymbol{\omega}^o$ at its immediate neighbor node $k-1$ which is $\boldsymbol{\psi}_{k-1}^{(i)}$ in the defined cycle. Fig.1 shows its processing.

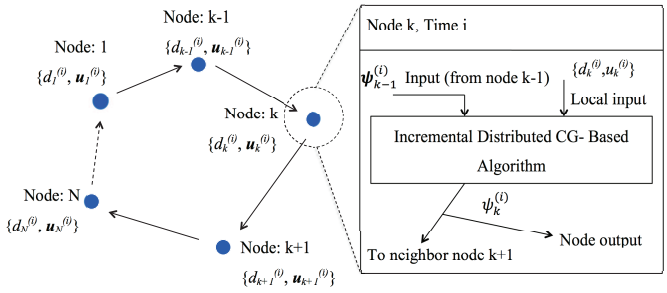

Fig. 1. Incremental distributed CG-based network processing

Based on the main steps of the CG algorithm, we propose two distributed adaptive filtering algorithms, namely, the CCG and the MCG for distributed estimating over sensor networks. The difference between these two strategies is that the CCG needs to run k iterations while the MCG only needs one iteration. The implementation of incremental distributed CCG solution (IDCCG) is shown in Table I. Similarly to the CCG algorithm,the incremental distributed MCG solution (IDMCG)

TABLE I
IDCG SOLUTIONS

IDCCG Solution	IDMCG Solution
Initialization:	Initialization:
$\boldsymbol{\omega}_0 = 0, \boldsymbol{g}(0) = \boldsymbol{b}, \boldsymbol{p}(1) = \boldsymbol{g}(0)$	$\boldsymbol{\omega}_0 = 0, \boldsymbol{g}_0 = \boldsymbol{b}, \boldsymbol{p}_1 = \boldsymbol{g}_0$
For each time instant i=1,2, …, n	For each time instant i=1,2, …, n
For each node k=1,2, …, N	For each node k=1,2, …, N
$\boldsymbol{\psi}_0^{(i)}(1) = \boldsymbol{\omega}_{i-1}$	$\boldsymbol{R}_k^{(i)} = \lambda_f \boldsymbol{R}_{k-1}^{(i)} + \boldsymbol{x}_k^{(i)} \boldsymbol{x}_k^{(i)H}$
$\boldsymbol{R}_k^{(i)} = \lambda_f \boldsymbol{R}_{k-1}^{(i)} + \boldsymbol{x}_k^{(i)} \boldsymbol{x}_k^{(i)H}$	$\boldsymbol{b}_k^{(i)} = \lambda_f \boldsymbol{b}_{k-1}^{(i)} + d_k^{(i)} \boldsymbol{x}_k^{(i)}$
$\boldsymbol{b}_k^{(i)} = \lambda_f \boldsymbol{b}_{k-1}^{(i)} + d_k^{(i)} \boldsymbol{x}_k^{(i)}$	$\boldsymbol{\psi}_0^{(i)} = \boldsymbol{\omega}_{i-1}$
For iterations j=1,2, …, J	$\alpha_k^{(i)} = \eta[\boldsymbol{p}_k^{(i)H} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}]^{-1} [\boldsymbol{p}_k^{(i)H} \boldsymbol{g}_{k-1}^{(i)}]$
$\alpha_k^{(i)}(j) = \eta[\boldsymbol{p}_k^{(i)}(j)^H \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}(j)]^{-1} [\boldsymbol{p}_k^{(i)}(j)^H \boldsymbol{g}_{k-1}^{(i)}(j)]$	where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$
where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$	$\boldsymbol{\psi}_k^{(i)} = \boldsymbol{\psi}_{k-1}^{(i)} + \alpha_k^{(i)} \boldsymbol{p}_k^{(i)}$
$\boldsymbol{\psi}_k^{(i)}(j) = \boldsymbol{\psi}_{k-1}^{(i)}(j) + \alpha_k^{(i)}(j) \boldsymbol{p}_k^{(i)}(j)$	$\boldsymbol{g}_k^{(i)} = \lambda_f \boldsymbol{g}_{k-1}^{(i)} - \alpha_k^{(i)} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}$
$\boldsymbol{g}_k^{(i)}(j) = \boldsymbol{g}_k^{(i)}(j-1) - \alpha_k^{(i)}(j) \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}(j)$	$\beta_k^{(i)} = [\boldsymbol{g}_{k-1}^{(i)H} \boldsymbol{g}_{k-1}^{(i)}]^{-1} [(\boldsymbol{g}_k^{(i)} - \boldsymbol{g}_{k-1}^{(i)})^H \boldsymbol{g}_k^{(i)}]$
$\beta_k^{(i)}(j) = [\boldsymbol{g}_k^{(i)}(j-1)^H \boldsymbol{g}_k^{(i)}(j-1)]^{-1} [(\boldsymbol{g}_k^{(i)}(j))^H \boldsymbol{g}_k^{(i)}(j)]$	$\boldsymbol{p}_{k+1}^{(i)} = \boldsymbol{g}_k^{(i)} + \beta_k^{(i)} \boldsymbol{p}_k^{(i)} + \boldsymbol{x}_k^{(i)} [d_k^{(i)} - \boldsymbol{x}_k^{(i)H} \boldsymbol{\psi}_{k-1}^{(i)}]$
$\boldsymbol{p}_k^{(i)}(j+1) = \boldsymbol{g}_k^{(i)}(j) + \beta_k^{(i)}(j) \boldsymbol{p}_k^{(i)}(j)$	$\boldsymbol{\omega}_i = \boldsymbol{\psi}_N^{(i)}$
$j = j + 1$	$k = k + 1$
After J iterations	After N iterations
$k = k + 1$	$i = i + 1$
After N iterations	
$\boldsymbol{\omega}_i = \boldsymbol{\psi}_N^{(i)}$, $i = i + 1$	

only needs one iteration per time instant and the details are shown in Table I. These two incremental distributed CG- based solutions can be summarized as:

1) assess local error
2) update its weight vector
3) pass the updated weight estimate $\boldsymbol{\psi}_k^{(i)}$ to its neighbor node

The idea of the MCG algorithm comes from the CCG algorithm. Instead of equation (10), we redefine the negative gradient vector with a recursive expression [9]:

$$\begin{aligned} \boldsymbol{g}_k^{(i)} &= \boldsymbol{b}_k^{(i)} - \boldsymbol{R}_k^{(i)} \boldsymbol{\omega}_k^{(i)} \\ &= \lambda_f \boldsymbol{g}_{k-1}^{(i)} - \alpha_k^{(i)} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)} \\ &\quad + \boldsymbol{x}_k^{(i)} [d_k^{(i)} - \boldsymbol{x}_k^{(i)H} \boldsymbol{\omega}_{k-1}^{(i)}]. \end{aligned} \tag{17}$$

Premultiplying (17) by $\boldsymbol{p}_k^H$ and considering $\boldsymbol{p}_k$ uncorrelated with $\boldsymbol{x}_k$, d_k and $\boldsymbol{\omega}_{k-1}$ and then taking the expectation, we get:

$$\begin{aligned} E[\boldsymbol{p}_k^{(i)H} \boldsymbol{g}_k^{(i)}] &= \lambda_f E[\boldsymbol{p}_k^{(i)H} \boldsymbol{g}_{k-1}^{(i)}] \\ &\quad - E[\alpha_k^{(i)}] E[\boldsymbol{p}_k^{(i)H} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k{}^{(i)}] \\ &\quad + E[\boldsymbol{p}_k^{(i)H} \boldsymbol{x}_k^{(i)} [d_k^{(i)} - \boldsymbol{x}_k^{(i)H} \boldsymbol{\omega}_{k-1}^{(i)}]]. \end{aligned} \tag{18}$$

Assuming the algorithm converges, then the last term of (18) could be neglected and we will get:

$$E[\alpha_k{}^{(i)}] = \frac{E[\boldsymbol{p}_k^{(i)H} \boldsymbol{g}_k{}^{(i)}] - \lambda_f E[\boldsymbol{p}_k^{(i)H} \boldsymbol{g}_{k-1}^{(i)}]}{E[\boldsymbol{p}_k^{(i)H} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}]}$$

and

$$(\lambda_f - 0.5) \frac{E[\boldsymbol{p}_k^{(i)H} \boldsymbol{g}_{k-1}^{(i)}]}{E[\boldsymbol{p}_k^{(i)H} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}]} \leq E[\alpha_k^{(i)}] \leq \frac{E[\boldsymbol{p}_k^{(i)H} \boldsymbol{g}_{k-1}^{(i)}]}{E[\boldsymbol{p}_k^{(i)H} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}]} \tag{19}$$

The inequalities in (19) are satisfied if we define:

$$\alpha_k^{(i)} = \eta \frac{\boldsymbol{p}_k^{(i)H} \boldsymbol{g}_{k-1}^{(i)}}{\boldsymbol{p}_k^{(i)H} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}}, \tag{20}$$

TABLE II
COMPUTATIONAL COMPLEXITY OF ALGORITHMS.

Algorithm	Additions	Multiplications
IDCCG	$m^2 + m$ $+J(m^2 + 6m - 4)$	$2m^2 + 2m$ $J(m^2 + 7m + 3)$
IDMCG	$2m^2 + 10m - 4$	$3m^2 + 12m + 3$
IDLMS [1]	$4m - 1$	$3m + 1$
IDRLS[3]	$4m^2 + 12m + 1$	$4m^2 + 12m - 1$

where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$. The direction vector $\boldsymbol{p}_k$ is defined by:

$$\boldsymbol{p}_{k+1}^{(i)} = \boldsymbol{g}_k^{(i)} + \beta_k^{(i)} \boldsymbol{p}_k^{(i)} \tag{21}$$

where β_k is computed to avoid the residue produced by using the Polak-Ribiere approach [8] which is given by:

$$\beta_k^{(i)} = \frac{(\boldsymbol{g}_k^{(i)} - \boldsymbol{g}_{k-1}^{(i)})^H \boldsymbol{g}_k^{(i)}}{\boldsymbol{g}_{k-1}^{(i)H} \boldsymbol{g}_{k-1}^{(i)}}. \tag{22}$$

C. Computational Complexity

To analyse the proposed incremental distributed CG algorithms, we detail the computational complexity. Additions and multiplications are used to measure the complexity and listed in Table II. It is obvious that the complexity of the incremental distributed CCG algorithm depends on the iteration number j.

IV. PROPOSED DIFFUSION DISTRIBUTED CG - BASED ALGORITHMS

A. Network Structure

For the diffusion distributed CG- based strategy, we consider a network structure where each node from the same neighborhood could exchange information with each other at every iteration. For each node in the network, it can collect information from all its neighbors and itself, and then convey all the information to its local adaptive filter and update the estimation of the weight vector through our algorithms. Specifically, at any time instant $i-1$, we define that node k has access to a set of unbiased estimates $\{\boldsymbol{\psi}_k^{(i-1)}\}_{k \in N_k}$ from its neighborhood N_k including itself. Then, these local estimates are combined at node k as

$$\boldsymbol{\phi}_k^{(i-1)} = \sum_{l \in N_{k,i-1}} c_{kl} \boldsymbol{\psi}_l^{(i-1)} \tag{23}$$

TABLE IV
COMPUTATIONAL COMPLEXITY OF ALGORITHMS

Algorithm	Additions	Multiplications
DDCCG	m^2+m $+J(m^2+6m$ $+Lm-4)$	$2m^2+2m$ $+J(m^2+7m$ $+Lm+3)$
DDMCG	$2m^2+10m-4$ $+Lm$	$3m^2+12m+3$ $+Lm$
DDLMS [2]	$4m-1+Lm$	$3m+1+Lm$
DDRLS [4]	$4m^2+16m+1+Lm$	$4m^2+12m-1+Lm$

where c_{kl} should be satisfied

$$\sum_l c_{kl} = 1, l \in N_{k,i-1} \forall k \tag{24}$$

Among the strategies to choose the combiner C are the Metropolis, the Laplacian and the nearest neighbor rules [2]. For our proposed diffusion distributed CG- based algorithm, we choose the Metropolis whose processing is shown in Fig. 2 and operates as follows:

$$\begin{cases} c_{kl} = \frac{1}{max(n_k, n_l)}, & \text{if } k \neq l \text{ are linked} \\ c_{kl} = 0, & \text{for } k \text{ and } l \text{ not linked} \\ c_{kk} = 1 - \sum_{l \in N_k/k} c_{kl}, & \text{for } k = l \end{cases} \tag{25}$$

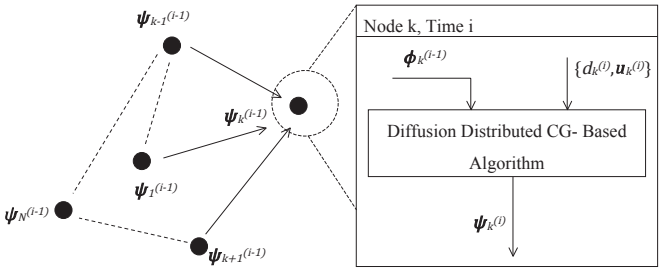

Fig. 2. Diffusion Distributed CG- Based Network Processing

B. Diffusion Distributed CG-Based Solutions

The CCG and MCG algorithms are also developed for the diffusion distributed CG - based solutions, the details for these two algorithms are shown in Table III. To derive these two algorithms, we first use equation (23) to get the unbiased estimates $\boldsymbol{\phi}_k^{(i-1)}$ and substitute them into equation (11), which results in:

$$\boldsymbol{\psi}_k^{(i)}(j) = \boldsymbol{\phi}_k^{(i-1)}(j) + \alpha_k^{(i)}(j)\boldsymbol{p}_k^{(i)}(j) \tag{26}$$

The rest of the derivation is similar to the incremental CG-based solutions.

C. Computational Complexity

The computational complexity is used to analyse the proposed diffusion distributed CG - based algorithms where additions and multiplications are measured. The details are listed in Table IV. Similarly to incremental distributed CG - based algorithms, it is clear that the complexity of the incremental distributed CCG algorithm depends on the iteration number j and the number of linked nodes l.

V. SIMULATION RESULTS

In this part, we test the proposed incremental and diffusion distributed CG - based algorithms in a sensor network and compare the results with LMS, RLS and AP [3] algorithms based on the performance of excess MSE (EMSE). For each test, the number of repetitions is set to 1000, and we assume there are 20 nodes in the network. The number of taps of the adaptive filter is 10, the variance for the input signal and the noise are 1 and 0.001, respectively. Besides, the noise samples are modeled as complex Gaussian noise.

A. Performance of Proposed IDCG Algorithms

First, we give out the definitions of the parameters of our test for each algorithms and the network. After 1000 iterations, the performance of each algorithm has been shown in Fig. 3. We can see that, the performance of the IDMCG and IDCCG algorithm is better than IDLMS, while IDMCG is very close to the IDRLS algorithm's curve. The reason why the proposed IDMCG algorithm has a better performance is IDMCG defined the negative gradient vector $\boldsymbol{g}_k$ with a recursive expression and the β_k is computed using the Polak-Ribiere approach. Comparing with the IDCCG algorithm, the IDMCG is a non-reset and low complexity algorithm with one iteration per time instant. Since the frequency which the algorithm resets influences the performance, the IDMCG algorithm introduces the non-reset method together with the Polak- Ribiere approach which are used to improve the performance [8].

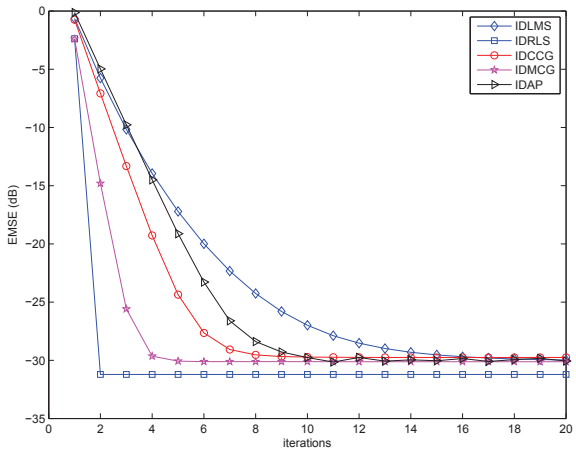

Fig. 3. Output EMSE against the number of iterations for the incremental strategy with α_{IDLMS}=0.005, λ=0.2, $\lambda_{f-IDCCG}$=0.3,$\lambda_{f-IDMCG}$=0.25, $\eta_{f-IDCCG}$=$\eta_{f-IDMCG}$=0.15, j=5, α_{IDAP}=0.06, K=2

B. Performance of Proposed DDCG Algorithms

In order to assess the proposed DDCG algorithms, we use some similar definitions of parameters as in the last part. For the diffusion strategy, we build the link between each node randomly, and for the combiner C, we calculate it following the Metropolis rule. Fig. 4 shows the network structure. After 1000 iterations, the results are shown in Fig. 5. We can see that, the proposed DDMCG and DDCCG still have a better performance than the DDLMS algorithm and DDMCG is closer to the DDRLS's curve. For the diffusion strategy, the network structure has a significant influence on the performance of our proposed DDCG Algorithms.

TABLE III
DDCG SOLUTIONS

DDCCG Solution	DDMCG Solution
Initialization:	Initialization:
$\boldsymbol{\omega}_0 = 0, \boldsymbol{g}(0) = \boldsymbol{b}, \boldsymbol{p}(1) = \boldsymbol{g}(0)$	$\boldsymbol{\omega}_0 = 0, \boldsymbol{g}_0 = \boldsymbol{b}, \boldsymbol{p}_1 = \boldsymbol{g}_0$
For each time instant i=1,2, …, n	For each time instant i=1,2, …, n
For each node k=1,2, …, N	For each node k=1,2, …, N
$\boldsymbol{\phi}_k^{(-1)}(1) = 0$	$\boldsymbol{R}_k^{(i)} = \lambda_f \boldsymbol{R}_{k-1}^{(i)} + \boldsymbol{x}_k^{(i)} \boldsymbol{x}_k^{(i)H}$
$\boldsymbol{R}_k^{(i)} = \lambda_f \boldsymbol{R}_{k-1}^{(i)} + \boldsymbol{x}_k^{(i)} \boldsymbol{x}_k^{(i)H}$	$\boldsymbol{b}_k^{(i)} = \lambda_f \boldsymbol{b}_{k-1}^{(i)} + d_k^{(i)} \boldsymbol{x}_k^{(i)}$
$\boldsymbol{b}_k^{(i)} = \lambda_f \boldsymbol{b}_{k-1}^{(i)} + d_k^{(i)} \boldsymbol{x}_k^{(i)}$	$\boldsymbol{\phi}_k^{(-1)}(1) = 0$
For iterations j=1,2, …, J	$\alpha_k^{(i)} = \eta[\boldsymbol{p}_k^{(i)H} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}]^{-1} [\boldsymbol{p}_k^{(i)H} \boldsymbol{g}_{k-1}^{(i)}]$
$\alpha_k^{(i)}(j) = \eta[\boldsymbol{p}_k^{(i)}(j)^H \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}(j)]^{-1} [\boldsymbol{p}_k^{(i)}(j)^H \boldsymbol{g}_k^{(i)}(j-1)]$	where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$
where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$	$\boldsymbol{\phi}_k^{(i-1)} = \sum_{l \in N_{k,i-1}} c_{kl} \boldsymbol{\psi}_l^{(i-1)}$
$\boldsymbol{\phi}_k^{(i-1)}(j) = \sum_{l \in N_{k,i-1}} c_{kl} \boldsymbol{\psi}_l^{(i-1)}(j)$	$\boldsymbol{\psi}_k^{(i)} = \boldsymbol{\phi}_k^{(i-1)} + \alpha_k^{(i)} \boldsymbol{p}_k^{(i)}$
$\boldsymbol{\psi}_k^{(i)}(j) = \boldsymbol{\phi}_k^{(i-1)}(j) + \alpha_k^{(i)}(j) \boldsymbol{p}_k^{(i)}(j)$	$\boldsymbol{g}_k^{(i)} = \lambda_f \boldsymbol{g}_{k-1}^{(i)} - \alpha_k^{(i)} \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)} + \boldsymbol{x}_k^{(i)} [d_k^{(i)} - \boldsymbol{x}_k^{(i)H} \boldsymbol{\phi}_k^{(i-1)}]$
$\boldsymbol{g}_k^{(i)}(j) = \boldsymbol{g}_k^{(i)}(j-1) - \alpha_k^{(i)}(j) \boldsymbol{R}_k^{(i)} \boldsymbol{p}_k^{(i)}(j)$	$\beta_k^{(i)} = [\boldsymbol{g}_{k-1}^{(i)H} \boldsymbol{g}_{k-1}^{(i)}]^{-1} [(\boldsymbol{g}_k^{(i)} - \boldsymbol{g}_{k-1}^{(i)})^H \boldsymbol{g}_k^{(i)}]$
$\beta_k^{(i)}(j) = [\boldsymbol{g}_k^{(i)}(j-1)^H \boldsymbol{g}_k^{(i)}(j-1)]^{-1} [(\boldsymbol{g}_k^{(i)}(j))^H \boldsymbol{g}_k^{(i)}(j)]$	$\boldsymbol{p}_{k+1}^{(i)} = \boldsymbol{g}_k^{(i)} + \beta_k^{(i)} \boldsymbol{p}_k^{(i)}$
$\boldsymbol{p}_k^{(i)}(j+1) = \boldsymbol{g}_k^{(i)}(j) + \beta_k^{(i)}(j) \boldsymbol{p}_k^{(i)}(j)$	$k = k + 1$
$j = j + 1$	After N iterations
After J iterations	$\boldsymbol{\omega}_i = \boldsymbol{\psi}_N^{(i)}$
$k = k + 1$	$i = i + 1$
After N iterations	
$\boldsymbol{\omega}_i = \boldsymbol{\psi}_N^{(i)}, i = i + 1$	

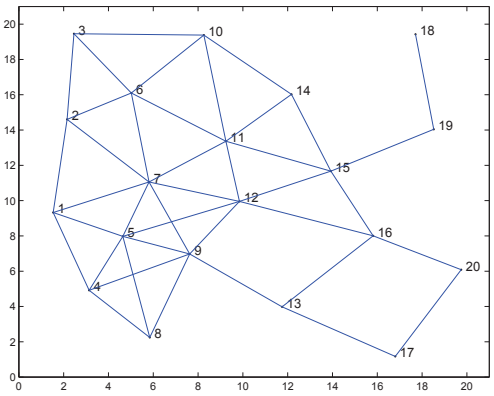

Fig. 4. Network structure.

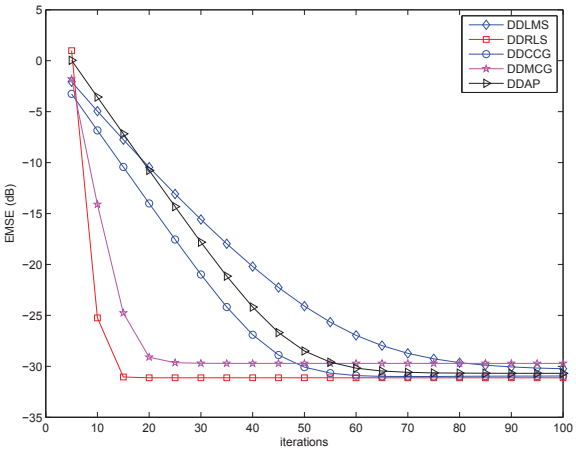

Fig. 5. Output EMSE against the number of iterations for the diffusion strategy with α_{DDLMS}=0.0075, λ=0.998, $\lambda_{f-DDCCG}$=0.25, $\eta_{f-DDCCG}$=0.25, j=5, $\lambda_{f-DDMCG}$=0.46, $\eta_{f-DDMCG}$=0.45, α_{DDAP}=0.075, K=2.

VI. CONCLUSIONS

We have developed distributed CG algorithms for incremental and diffusion type estimation over sensor networks. The CG- based strategies avoid the matrix inversion and numerical instability of RLS algorithms and have a faster convergence than LMS and AP algorithms. Simulation results have shown that the proposed IDMCG and DDMCG algorithms have a better performance than the LMS and AP algorithm, and a close performance to the RLS algorithm.

REFERENCES

[1] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks", *IEEE Trans. Sig. Proc.*, vol. 55, no. 8, pp. 4064-4077, August 2007.

[2] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis", *IEEE Trans. Sig. Proc.*, vol. 56, no. 7, pp. 3122-3136, July 2008.

[3] L.L. Li, J.A. Chambers, C.G. Lopes,and A.H. Sayed, "Distributed Estimation Over an Adaptive Incremental Network Based on the Affine Projection Algorithm", *IEEE Trans. Sig. Proc.*, vol. 58, issue. 1, pp. 151-164, Jan. 2010.

[4] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," IEEE Trans. Sig. Proc., vol. 56, no. 5, pp. 1865-1877, May 2008.

[5] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Distributed Recursive Least-Squares for Consensus-Based In-Network Adaptive Estimation," IEEE Trans. Sig. Proc., vol. 57, no. 11, pp. 4583-4588,November 2009.

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

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

[8] 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, pp. 409-418, Febrary 2000.

[9] L. Wang, and R.C.de Lamare , "Constrained adaptive filtering algorithms based on conjugate gradient techniques for beamforming ", *IET Signal Processing*, vol. 4, issue. 6, pp. 686-697, Feb. 2010.

[10] R. Fa, R. C. de Lamare, and L. Wang, "Reduced-Rank STAP Schemes for Airborne Radar Based on Switched Joint Interpolation, Decimation and Filtering Algorithm", *IEEE Trans. Sig. Proc.*, vol. 58, no. 8, pp. 4182-4194, Aug. 2010.

[11] R. C. de Lamare and R. Sampaio-Neto, "Adaptive Reduced-Rank Processing Based on Joint and Iterative Interpolation, Decimation, and Filtering", *IEEE Trans. Sig. Proc.*, vol. 57, no. 7, pp. 2503 - 2514, Juy. 2009.