A GENERAL FORMALISM FOR THE ANALYSIS OF DISTRIBUTED ALGORITHMS

Ondrej Slučiak, Thibault Hilaire, Markus Rupp

Vienna University of Technology, Austria
Institute of Communications and Radio-Frequency Engineering

{osluciak, thilaire, mrupp}@nt.tuwien.ac.at

ABSTRACT

The major contribution of this paper is the presentation of a general unifying description of distributed algorithms allowing to map local, node-based algorithms onto a single global, network-based form. As a first consequence the new description offers to analyze their learning and steady-state behavior by classical methods. A further consequence is the analysis of implementation issues as they appear due to quantization in computing and communication links. Exemplarly we apply the new method on several different averaging algorithms: the Push-Sum protocol, Consensus Propagation as well as its quantized form and furthermore examine the effects of quantization noise which is introduced by the bandwidth limited communication links and finite precision computation ability of every node. Statistical properties of these quantization noises are provided and verified by simulations.

Index Terms — distributed algorithms, unified description, implementation, quantization, distributed averaging problem

1. INTRODUCTION

Distributed computing environments, such as ad-hoc wireless sensor networks and computer clusters, provide many advantages over a centralized processing solution, larger computation power and robustness against node failures being examples.

However, comparing the ad-hoc wireless sensor networks with computer clusters, it is clear that the first mentioned solution is much more difficult to implement reliably. Since the nodes in sensor networks usually have no information about the whole network topology and have only a small computation power, and communication links are of limited bandwidth, proper distributed algorithms that always converge to the desired result need to be designed very carefully.

Well-known approaches that satisfy these requirements is the gossip-based and consensus based approach, both solving the problem of distributed averaging[1, 2]. More sophisticated algorithms also based on these basic approaches include distributed least-mean squares (LMS)[3], recursive least-squares (RLS)[4] or projection approximation subspace tracking (PAST)[5] algorithm.

Typically all these algorithms assume infinite precision computation in every node and unlimited bandwidth on communication links. It is of interest to investigate the impact of these imperfections on the performance of distributed algorithms. Such errors are caused by the following constraints:

- *Quantized measurement* – sensors have limited sensing capabilities[6, 7].
- *Quantized computation* – sensors have limited computation precision, usually with fixed-point representation.
- *Quantized communication links* – sensors are able to transmit only few bits due to power constraints and limited bandwidth [8, 9].

All these perturbations caused by quantization can dramatically change the accuracy, convergence speed and stability of the distributed algorithms. In order to propose solutions that are more robust and to deeply study the impact of the implementation, we introduce a unifying description of any distributed algorithm. It allows us to consider a wide range of algorithms with the same analysis tool.

Organization of the paper In Section 2 we introduce the notation and the general framework, which is in Section 3 applied to distributed averaging algorithms. The impact of the quantization and its statistical properties are derived in Section 4. Finally in Section 5 we show and discuss some simulation results.

2. UNIFYING DESCRIPTION

2.1. Notation

A network is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of $|\mathcal{V}|$ vertices (nodes) and $\mathcal{E}$ is a set of $|\mathcal{E}|$ edges. The graph is supposed to have no self-loops ($e = (u,u) \notin \mathcal{E}$) and each element of $\mathcal{E}$ is unique. Let us denote $Pred(v)$ and $Succ(v)$ the set of preceding and succeeding vertices of node v, respectively, i.e.

$$Pred(v) \triangleq \{u \in \mathcal{V} \mid \exists\, e = (u,v) \in \mathcal{E}\} \quad (1)$$

$$Succ(v) \triangleq \{w \in \mathcal{V} \mid \exists\, e = (v,w) \in \mathcal{E}\}. \quad (2)$$

The network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is supposed to be static. *Nevertheless* we can consider some instant k_0 when we switch from network $\mathcal{G}_1$ to network $\mathcal{G}_2$ (for example by adding or removing some edges or vertices).

We associate the graph $\mathcal{G}$ with a numbering scheme (*i.e.* a bijection that associates each node to a unique number in $\{1, \dots, |\mathcal{V}|\}$), so that a node can serve as an index of a matrix. Then we can define the *ingoing-adjacency* matrix of $\mathcal{G}$, denoted $\mathbf{A}_\mathcal{G} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, by

$$(\mathbf{A}_\mathcal{G})_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{if } (i,j) \notin \mathcal{E}. \end{cases} \quad (3)$$

2.2. Distributed algorithms

In this contribution we only consider synchronized[1] nodes. At the beginning of step k, each node receives data from its neighbors, then uses them in its algorithm, and send some data at the end of step k to its neighbors. They will receive them at the beginning of step $k+1$.

This work has been funded by the NFN SISE project (National Research Network "Signal and Information Processing in Science and Engineering").

[1]This synchronization may be performed in the higher protocol layer used in data transmission, e.g. [10].

We define the following:

- $\mathbf{z}_v(k)$ is the result of the main algorithm of node v at step k.
- $\mathbf{u}_v(k)$ is the measurement (observation) of node v at step k.
- $\mathbf{y}_{u\to v}(k)$ is the data sent from node u to node v. Node u sends them at the end of step $k-1$, and node v receives them at the beginning of step k.

In some algorithms (like in consensus propagation), the result of computation at node v, $\mathbf{z}_v(k)$, is exactly the data sent to its neighbors $\mathbf{y}_{u\to v}(k)$, but in general it is not always the case.

Note that **z**, **u** and **y** can be *scalars*, *vectors*, *matrices*, or a *collection* (list) of scalars/vectors/matrices. Without loss of generality, we consider them as **column** vectors. Their dimensions are constant in time.

We will also consider $\mathbf{y}_{v\to v}(k+1)$ the data computed at step k by node v and sent to itself (referred later in the paper as a *state*). Moreover, at each step k, nodes communicate with some (not necessary all) of their neighbors. We denote the following sets:

1. $Send_v(k) \subseteq Succ(v)$: set of nodes (in the neighborhood of v) to which the node v is going to send the messages
2. $Rec_v(k) \subseteq Pred(v)$: set of nodes (in the neighborhood of v) from which the node v received the messages at step k,

If there is no failure in the communication links, then:

$$\forall v \in \mathcal{V}, \quad Rec_v(k+1) \triangleq \{u \in \mathcal{V} \mid v \in Send_u(k)\} \tag{4}$$

otherwise the equality becomes an inclusion.

Note also that it is equivalent to define $\mathcal{E}'(k) \subseteq \mathcal{E}$ as the subset of *used* edges (in that case, $Send_v(k) = Succ_{\mathcal{E}'(k)}(v)$ and $Rec_v(k) = Pred_{\mathcal{E}'(k)}(v)$). Then $\mathcal{G}'(k) = (\mathcal{V}, \mathcal{E}'(k))$ denotes the sub-network at step k.

For every node v, at every time step k, a ***local*** *distributed algorithm* is composed of following procedures:

1. Define the set $Send_v(k)$ (for broadcast, $Send_v(k) = Succ(v)$).
2. Receive data from the communicating neighbors $\{\mathbf{y}_{u\to v}(k)\}_{u\in Rec_v(k)}$.
3. Compute the computation result $\mathbf{z}_v(k)$, the transmission data $\{\mathbf{y}_{v\to w}(k+1)\}_{w\in Send_v(k)}$ and the local data to store $\mathbf{y}_{v\to v}(k+1)$ (data to *send* to itself) from the current measurement $\mathbf{u}_v(k)$, the data received $\{\mathbf{y}_{u\to v}(k)\}_{u\in Rec_v(k)}$ and the local stored data $\mathbf{y}_{v\to v}(k)$.
4. Send the data $\{\mathbf{y}_{v\to w}(k+1)\}_{w\in Send_v(k)}$ to the selected neighbors.

2.3. Homogeneously distributed algorithms

Definition 1 (HDA). *Homogeneously distributed algorithm is a local distributed algorithm where:*

- *a node does not make any differences in its neighbors*
- *a node sends its data with the same data-type to its neighbors*

i.e., $\forall v \in \mathcal{V}, \forall w \in Succ(v)$:

$$\mathbf{y}_{v\to w}(k) = \mathbf{y}_v(k) \tag{5}$$

which can be formalized as follows:

$$\mathbf{z}_v(k) = f_v\left(\mathbf{u}_v(k), \{\mathbf{y}_u(k)\}_{u\in Rec_v(k)}, \mathbf{x}_v(k)\right) \tag{6}$$

$$\mathbf{x}_v(k+1) = g_v\left(\mathbf{u}_v(k), \{\mathbf{y}_u(k)\}_{u\in Rec_v(k)}, \mathbf{x}_v(k)\right) \tag{7}$$

$$\mathbf{y}_v(k+1) = h_v\left(\mathbf{u}_v(k), \{\mathbf{y}_u(k)\}_{u\in Rec_v(k)}, \mathbf{x}_v(k)\right) \tag{8}$$

where $\mathbf{x}_v(k) \triangleq \mathbf{y}_{v\to v}(k)$ *is consistent with a* state *notation.*

The size of the set $\{\mathbf{y}_{u\to v}(k)\}_{u\in Rec_v(k)}$ may change at each time k, but the functions f_v, g_v, h_v are the same, capable to accept three inputs of time-varying sizes. Since the size does not depend on the node, let us also denote by n_X and n_Y the size of the column vector $\mathbf{x}_v(k)$ and $\mathbf{y}_v(k)$, respectively.

2.4. Linear HDA

Definition 2 (Linear HDA). *If* update *strategy* (7) *and the* communication *strategy* (8) *are linear functions, then the algorithm is said to be **linear homogeneously distributed** and can be described as follows:*

$$\mathbf{z}_v(k) = f_v\left(\mathbf{u}_v(k), \{\mathbf{y}_u(k)\}_{u\in Rec_v(k)}, \mathbf{x}_v(k)\right) \tag{9}$$

$$\mathbf{x}_v(k+1) = \alpha_v \sum_{u\in Rec_v(k)} \mathbf{y}_u(k) + \beta_v \mathbf{x}_v(k) + \theta_v \mathbf{u}_v(k) \tag{10}$$

$$\mathbf{y}_v(k+1) = \gamma_v \sum_{u\in Rec_v(k)} \mathbf{y}_u(k) + \delta_v \mathbf{x}_v(k) + \vartheta_v \mathbf{u}_v(k) \tag{11}$$

where $\alpha_v \in \mathbb{R}^{n_X\times n_Y}$ *(receptivity),* $\beta_v \in \mathbb{R}^{n_X\times n_X}$ *(self-transmissivity),* $\theta_v \in \mathbb{R}^{n_X\times n_U}$ *(absorptivity),* $\gamma_v \in \mathbb{R}^{n_Y\times n_Y}$ *(transmissivity),* $\delta_v \in \mathbb{R}^{n_Y\times n_X}$ *(distributivity) and* $\vartheta_v \in \mathbb{R}^{n_Y\times n_U}$ *(emissivity) are constant matrices.*

Remark: We do not consider the linearization of the outputs, since this equation is not involved in the iteration loop.

It is possible to aggregate the vectors $\{\mathbf{x}_v(k)\}_{v\in\mathcal{V}}$ together in a column vector $\mathbf{x}(k) \in \mathbb{R}^{(|\mathcal{V}|n_X)\times 1}$ (and the vectors $\{\mathbf{y}_v(k)\}_{v\in\mathcal{V}}$ together in a column vector $\mathbf{y}(k) \in \mathbb{R}^{(|\mathcal{V}|n_Y)\times 1}$, respectively).

Let us denote, to make the connection between *local* and *global* algorithms:

$$\boldsymbol{\alpha} \triangleq diag(\alpha_1, \ldots, \alpha_v, \ldots, \alpha_{|\mathcal{V}|}) \in \mathbb{R}^{(|\mathcal{V}|n_X)\times(|\mathcal{V}|n_Y)} \tag{12}$$

$$\boldsymbol{\beta} \triangleq diag(\beta_1, \ldots, \beta_v, \ldots, \beta_{|\mathcal{V}|}) \in \mathbb{R}^{(|\mathcal{V}|n_X)\times(|\mathcal{V}|n_X)} \tag{13}$$

$$\boldsymbol{\theta} \triangleq diag(\theta_1, \ldots, \theta_v, \ldots, \theta_{|\mathcal{V}|}) \in \mathbb{R}^{(|\mathcal{V}|n_X)\times(|\mathcal{V}|n_U)} \tag{14}$$

$$\boldsymbol{\gamma} \triangleq diag(\gamma_1, \ldots, \gamma_v, \ldots, \gamma_{|\mathcal{V}|}) \in \mathbb{R}^{(|\mathcal{V}|n_Y)\times(|\mathcal{V}|n_Y)} \tag{15}$$

$$\boldsymbol{\delta} \triangleq diag(\delta_1, \ldots, \delta_v, \ldots, \delta_{|\mathcal{V}|}) \in \mathbb{R}^{(|\mathcal{V}|n_Y)\times(|\mathcal{V}|n_X)} \tag{16}$$

$$\boldsymbol{\vartheta} \triangleq diag(\vartheta_1, \ldots, \vartheta_v, \ldots, \vartheta_{|\mathcal{V}|}) \in \mathbb{R}^{(|\mathcal{V}|n_Y)\times(|\mathcal{V}|n_U)}. \tag{17}$$

Proposition 1 (Global algorithm). *Using the set of equations (12-17), a global algorithm (the aggregation of the algorithms of all the nodes) can be formulated by:*

$$\mathbf{z}_v(k) = f\left(\mathbf{u}_v(k), \{\mathbf{y}_u(k)\}_{u\in Rec_v(k)}, \mathbf{x}_v(k)\right) \tag{18}$$

$$\mathbf{x}(k+1) = \boldsymbol{\alpha}\left(\mathbf{A}_{\mathcal{G}'(k)} \otimes \mathbf{I}_{n_Y}\right)\mathbf{y}(k) + \boldsymbol{\beta}\mathbf{x}(k) + \boldsymbol{\theta}\mathbf{u}(k) \tag{19}$$

$$\mathbf{y}(k+1) = \boldsymbol{\gamma}\left(\mathbf{A}_{\mathcal{G}'(k)} \otimes \mathbf{I}_{n_Y}\right)\mathbf{y}(k) + \boldsymbol{\delta}\mathbf{x}(k) + \boldsymbol{\vartheta}\mathbf{u}(k). \tag{20}$$

3. EXAMPLES

3.1. Push-Sum protocol

The push-sum protocol[11] is a simple algorithm for distributed averaging. It belongs to the class of gossip-based algorithms. According to our formalism, this is a *linear homogenously distributed algorithm* with following properties:

$$n_X = n_Y = 2, \quad \mathbf{u}(k) = \mathbf{0} \quad \forall k > 0,$$

$$\mathbf{x}_v(0) = \begin{pmatrix} \mathbf{u}_v(0) \\ 1 \end{pmatrix}$$

$\mathbf{x}(k)$ and $\mathbf{y}(k)$ both represent $\begin{pmatrix} sum \\ weight \end{pmatrix}$ (there is no difference between data kept and data sent). At time step k, each node randomly selects only one of his neighbors. Therefore $\mathbf{A}_{\mathcal{G}'(k)}$ contains at most one 1 in every row. We then find

$$\boldsymbol{\alpha} = \boldsymbol{\beta} = \boldsymbol{\gamma} = \boldsymbol{\delta} = \tfrac{1}{2}\mathbf{I}_{2|\mathcal{V}|} \tag{21}$$

$$\mathbf{z}_v(k) = \frac{\mathbf{x}^{(1)}(k)}{\mathbf{x}^{(2)}(k)} \tag{22}$$

where $\mathbf{x}^{(1)}(k)$ and $\mathbf{x}^{(2)}(k)$ are the two components of $\mathbf{x}_v(k)$.

3.2. Consensus propagation

Consensus propagation (e.g. [12]) is also a distributed algorithm for solving distributed averaging problems. It can be formalized by

$$n_X = n_Y = 1,\ \mathbf{u}(k) = \mathbf{0} \quad \forall k > 0,$$

$$\mathbf{z}(k) = \mathbf{x}(k+1) = \mathbf{y}(k+1)$$

and the nodes are initialized by $\mathbf{x}(0) = \mathbf{u}(0)$.
The nodes broadcast their data, therefore $\mathbf{A}_{\mathcal{G}'(k)} = \mathbf{A}_{\mathcal{G}}, \forall k \geq 0$, and

$$\boldsymbol{\beta} = \boldsymbol{\delta} = \mathbf{I}_{|\mathcal{V}|} - \epsilon\mathbf{D}_{\mathcal{G}} \tag{23}$$

$$\boldsymbol{\alpha} = \boldsymbol{\gamma} = \epsilon\mathbf{I}_{|\mathcal{V}|} \tag{24}$$

where $\epsilon > 0$ is the step-size, $\mathbf{D}_{\mathcal{G}}$ is the degree matrix of the graph $\mathcal{G}$.

3.3. Real-valued average consensus over noisy quantized channels [8, Censi]

In [8], Censi and Murray proposed a new strategy to deal with average consensus with quantized communication and preserve the convergence (not necessary to the true average).

Their algorithm is based on an integration of the quantization communication error to be re-injected into the system. Denoting $\mathbf{x}^\top(k) \triangleq \left(\mathbf{x}_1^\top(k), \mathbf{x}_2^\top(k), \mathbf{c}^\top(k)\right)$, and using the notation from [8] (for η; Δ; $\mathbf{D}$; $\mathbf{x}_1(k) \equiv x(k)$; $\mathbf{x}_2(k) \equiv y(k)$; $\mathbf{c}(k) \equiv c(k)$), the algorithm can be written as follows:

$$\boldsymbol{\alpha} = \mathbf{I}_{|\mathcal{V}|} \otimes \begin{pmatrix} \frac{\eta}{\Delta} \\ \frac{\eta}{\Delta} \\ 0 \end{pmatrix} \tag{25}$$

$$\boldsymbol{\beta} = \tfrac{\eta}{\Delta}\mathbf{D} \otimes \begin{pmatrix} -1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \mathbf{I}_{|\mathcal{V}|} \otimes \begin{pmatrix} 1 & 0 & 0 \\ 2 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix} \tag{26}$$

$$\boldsymbol{\gamma} = \tfrac{\eta}{\Delta}\mathbf{D} \otimes \begin{pmatrix} -1 & 0 & 0 \end{pmatrix} + \mathbf{I}_{|\mathcal{V}|} \otimes \begin{pmatrix} 2 & -1 & -1 \end{pmatrix} \tag{27}$$

$$\boldsymbol{\delta} = \tfrac{\eta}{\Delta}\mathbf{I}_{|\mathcal{V}|}. \tag{28}$$

Nodes are initialized by $\mathbf{x}^\top(0) = \left(\mathbf{u}^\top(0),\ \mathbf{0}_{|\mathcal{V}|}^\top,\ \mathbf{0}_{|\mathcal{V}|}^\top\right)$ and we communicate with all neighbors, i.e. $\mathbf{A}_{\mathcal{G}'(k)} = \mathbf{A}_{\mathcal{G}}, \forall k \geq 0$.

4. IMPACT OF THE QUANTIZATION

Let us define the first (μ) and second (σ^2, $\boldsymbol{\Psi}$) order moments of a noise vector ξ by:

$$\mu_\xi \triangleq E\left\{\xi(k)\right\} \tag{29}$$

$$\boldsymbol{\Psi}_\xi \triangleq E\left\{(\xi(k) - \mu_\xi)(\xi(k) - \mu_\xi)^\top\right\} \tag{30}$$

$$\sigma_\xi^2 \triangleq E\left\{(\xi(k) - \mu_\xi)^\top(\xi(k) - \mu_\xi)\right\} = tr(\boldsymbol{\Psi}_\xi) \tag{31}$$

where $E\{\cdot\}$ and $tr(\cdot)$ are the *mean* and the *trace* operator, respectively. Let us denote

$$\boldsymbol{\Gamma}(k) \triangleq \begin{pmatrix} \mathbf{x}(k) \\ \mathbf{y}(k) \end{pmatrix}. \tag{32}$$

Then the algorithm can be seen as a state-space system

$$\boldsymbol{\Gamma}(k+1) = \underbrace{\begin{pmatrix} \boldsymbol{\beta} & \boldsymbol{\alpha}(\mathbf{A}_{\mathcal{G}'(k)} \otimes \mathbf{I}_{n_Y}) \\ \boldsymbol{\gamma} & \boldsymbol{\delta}(\mathbf{A}_{\mathcal{G}'(k)} \otimes \mathbf{I}_{n_Y}) \end{pmatrix}}_{\mathbf{P}(k)} \boldsymbol{\Gamma}(k) + \underbrace{\begin{pmatrix} \boldsymbol{\theta} \\ \boldsymbol{\vartheta} \end{pmatrix}}_{\mathbf{Q}} \mathbf{u}(k). \tag{33}$$

Due to the real implementation, two terms need to be added:

- $\boldsymbol{\xi}'(k)$ - noise due to the computations
- $\boldsymbol{\xi}(k)$ - noise due to the quantization of the sent data (apply only to $\mathbf{y}(k)$).

The distortion in measurement is implicitly contained in $\mathbf{u}(k)$.

Remark: These noises are merely determined by the used implementation scheme. In a fixed-point scheme, these noises are independent white Gaussian noises with given moments defined by the word-lengths used in the algorithm, and the algorithm itself. For more details see [13, 14].

The implemented system is then given by

$$\boldsymbol{\Gamma}^\star(k+1) = \mathbf{P}(k)\boldsymbol{\Gamma}^\star(k) + \mathbf{Q}\mathbf{u}(k) + \mathbf{R}\boldsymbol{\zeta}(k), \tag{34}$$

where $\mathbf{R}$ is any linear transformation of the noise term $\boldsymbol{\zeta}(k)$ (note that $\boldsymbol{\zeta}(k)$ contains $\boldsymbol{\xi}(k)$ and $\boldsymbol{\xi}'(k)$). For example, for "Censi algorithm" (see Section 3.3), by definition of the algorithm, we have

$$\mathbf{R} = \begin{pmatrix} \mathbf{I}_{|\mathcal{V}|} \otimes \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\ \mathbf{I}_{|\mathcal{V}|} \end{pmatrix} \tag{35}$$

because the same communication noise applies to $\mathbf{x}_2(k)$ and $\mathbf{y}(k)$.

Proposition 2. *Considering a case where* $\mathbf{P}$ *is constant in time. Then, the term* $\Delta\boldsymbol{\Gamma}^\star(k) \triangleq \boldsymbol{\Gamma}^\star(k) - \boldsymbol{\Gamma}(k)$ *is the noise added to* $\boldsymbol{\Gamma}(k)$ *and satisfies*

$$\Delta\boldsymbol{\Gamma}^\star(k+1) = \mathbf{P}\Delta\boldsymbol{\Gamma}^\star(k) + \mathbf{R}\boldsymbol{\zeta}(k). \tag{36}$$

Thus, the first and second moment order of $\Delta\boldsymbol{\Gamma}^\star$ *are given by*

$$\mu_{\Delta\boldsymbol{\Gamma}^\star} = \mu_\zeta(\mathbf{I} - \mathbf{P})^{-1}\mathbf{R}, \quad \sigma^2_{\Delta\boldsymbol{\Gamma}^\star} = tr(\mathbf{W}), \tag{37}$$

where $\mathbf{W}$ *is the solution of the Lyapunov equation* $\mathbf{W} = \mathbf{P}\mathbf{W}\mathbf{P}^\top + \mathbf{R}\boldsymbol{\Psi}_\zeta\mathbf{R}^\top$.

Proof: The noise $\boldsymbol{\zeta}$ is added to $\Delta\boldsymbol{\Gamma}$ through the state-space system $(\mathbf{P}, \mathbf{R}, \mathbf{I}, \mathbf{0})$. Classical result on noises through systems applies here. See [13, 15] for more details. ■

Remark: For distributed algorithms solving averaging problems, the matrix $\mathbf{P}$ always contains one eigenvalue of value 1. In that case, $(\mathbf{I} - \mathbf{P})$ is not invertible, and we can conclude that the added noise $\boldsymbol{\zeta}$ must be of zero-mean for the algorithm to converge. However, the term $(\mathbf{I} - \mathbf{P})^{-1}\mathbf{R}$ can be computed when $\mathbf{R}$ works as a stabilizing term, *i.e.* when $\lim\limits_{n\to\infty} \sum_{i=0}^{n} \left(\mathbf{P}^i\mathbf{R}\right)$ exists. This is also the case for the "Censi algorithm" with $\mathbf{R}$ definied as (35).

Thus, using this approach we can exactly compute the *"drift from the mean"* ($\mu_{\Delta\boldsymbol{\Gamma}^\star}$) and the *"average disagreement"* ($\sigma^2_{\Delta\boldsymbol{\Gamma}^\star}$) for which Censi, though from different point of view, set only loose bounds (see Tab. 1).

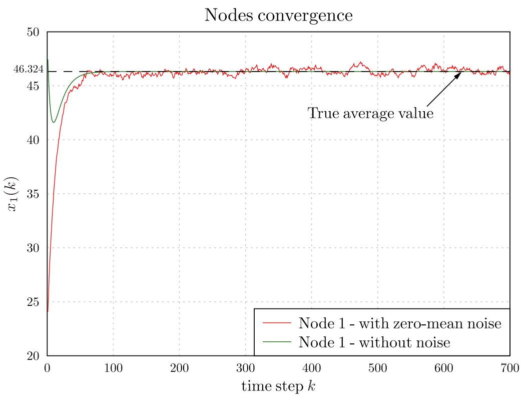
5. SIMULATIONS

To verify the above mentioned theoretical relations for consensus propagation and quantized consensus propagation, simulations with several networks with different topologies were performed.

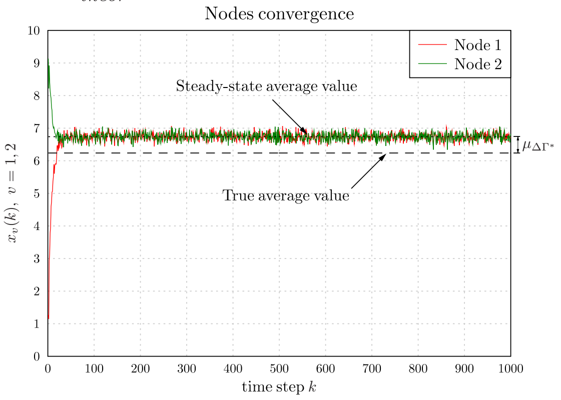
For unquantized consensus propagation the convergence with added zero-mean noise was studied and it was shown that the theoretical bounds hold (see Fig. 1(a)).

For consensus propagation with quantized communication (see Fig. 1(b)) the convergence of the algorithm even for non-zero mean noise was verified. We observe that this additive noise can be of any type (not only e.g. *round-off* noise). The theoretical results of Proposition 2 were confirmed by simulations and compared with bounds derived by Censi (see Tab. 1). It must be noted, however, that (μ_ζ, $\mathbf{\Psi}_\zeta$) were computed from true error noise added, while Censi's bounds are predicted. Nevertheless, the estimation of (μ_ζ, $\mathbf{\Psi}_\zeta$) of quantization noise (see [13]), as well as any additive noise, can be calculated in advance, thus the equation (37) still holds.

The equivalence of the proposed formalism with the original algorithms was examined and was proved to be exactly, value by value, identical.



(a) Behaviour of node 1 - consensus propagation (see Section 3.2): $\mu_{\Delta\mathbf{\Gamma}^*_{sim}} = 0$, $\mu_{\Delta\mathbf{\Gamma}^*_{theor}} = 0$, $\sigma^2_{\Delta\mathbf{\Gamma}^*_{sim}} = 3.5$, $\sigma^2_{\Delta\mathbf{\Gamma}^*_{theor}} = 3.53$.



(b) Behaviour of the first 2 nodes – "Censi algorithm" (see Section 3.3): $\mu_{\Delta\mathbf{\Gamma}^*_{sim}} = 0.50$, $\mu_{\Delta\mathbf{\Gamma}^*_{theor}} = 0.49$, $\sigma^2_{\Delta\mathbf{\Gamma}^*_{sim}} = 82.86$, $\sigma^2_{\Delta\mathbf{\Gamma}^*_{theor}} = 82.86$.

Fig. 1. Simulation for a regular undirected network with random initial values. $|\mathcal{V}| = 10,\ \Delta = 6$.

6. CONCLUSION

In this paper we have proposed a general formalism for the analysis of distributed algorithms. In this formalism we have defined a class of algorithms (*linear HDA*) having equivalent *global* behaviour and we have analyzed some well-known algorithms belonging to that class. Although these first analyses have revealed some of the capabilities of this framework, a deeper insight into this formalism which could help us to better analyze the behaviour of implemented distributed algorithms (e.g. fixed-point implementation), is still needed. Novel robust algorithms based on bounds, set by this formalism, seem to be also a potential goal of future research.

network topology $\lvert\mathcal{V}\rvert = 10$	$\mu_{\Delta\mathbf{\Gamma}^*_{theor}}$	Censi's bound on the mean drift[8]	$\sigma^2_{\Delta\mathbf{\Gamma}^*_{theor}}$	Censi's average disagreement[8]
star	0.0032	0.05	0.0000585	1.2247
complete	0.0121	0.05	0.00000908	1.2247

Table 1. Comparison of Censi's bounds (see [8]) vs. our exact theoretical approach.

7. REFERENCES

[1] C. C. Moallemi and B. Van Roy, "Consensus Propagation," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4753–4766, 2006.

[2] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[3] A.H.Sayed and C.G. Lopes, "Distributed processing over adaptive networks," in *9th International Symposium on Signal Processing and Its Applications (ISSPA 2007)*, Feb. 2007, pp. 1–3.

[4] A.H. Sayed and C.G. Lopes, "Distributed recursive least-squares strategies over adaptive networks," in *Fortieth Asilomar Conference on Signals, Systems and Computers (ACSSC '06)*, Nov. 2006, pp. 233–237.

[5] C. Reyes, T. Hilaire, and C. F. Mecklenbräuker, "Distributed Projection Approximation Subspace Tracking Based on Consensus Propagation," *The Third International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP '09)*, 2009.

[6] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in Ad Hoc WSNs With Noisy Links– Part I: Distributed Estimation of Deterministic Signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350 – 364, 2008.

[7] A. Ribeiro, *Distributed Quantization-Estimation for Wireless Sensor Networks*, Ph.D. thesis, Faculty of the graduate school of the university of Minnesota, 2005.

[8] A. Censi and R.M. Murray, "Real-valued average consensus over noisy quantized channels," in *American Control Conference (ACC '09)*, June 2009, pp. 4361–4366.

[9] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," in *IEEE International Symposium on Information Theory*, July 2006, pp. 635–639.

[10] K. Römer, "Time synchronization in ad hoc networks," in *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 2001, pp. 173–182.

[11] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of 44th Annual IEEE Symposium on Foundations of Computer Science*, Oct. 2003, pp. 482–491.

[12] P. Frasca, R. Carli, F. Fagnani, and S. Zampieri, "Average consensus on networks with quantized communication," in *Int. Journal of Robust and Nonlinear Control*, 2008.

[13] T. Hilaire, D. Ménard, and O. Sentieys, "Bit accurate roundoff noise analysis of fixed-point linear controllers," in *Proc. IEEE Int. Symposium on Computer-Aided Control System Design (CACSD'08)*, Sept. 2008.

[14] B. Widrow, "Statistical analysis of amplitude quantized sampled-data systems," *Trans AIEE*, vol. 2, no. 79, pp. 555–568, 1960.

[15] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, Mc Graw Hill, 1991.