# DISTRIBUTED TOTAL LEAST SQUARES ESTIMATION OVER NETWORKS

*Roberto López-Valcarce*[*]

Universidade de Vigo, Spain

*Silvana Silva Pereira*[†]*, Alba Pagès-Zamora*[†]

Universitat Politècnica de Catalunya, Spain

## ABSTRACT

We consider Total Least Squares (TLS) estimation in a network in which each node has access to a subset of equations of an overdetermined linear system. Previous distributed approaches require that the number of equations at each node be larger than the dimension $L$ of the unknown parameter. We present novel distributed TLS estimators which can handle as few as a single equation per node. In the first scheme, the network computes an extended correlation matrix via standard iterative average consensus techniques, and the TLS estimate is extracted afterwards by means of an eigenvalue decomposition (EVD). The second scheme is EVD-free, but requires that a linear system of size $L$ be solved at each iteration by each node. Replacing this step by a single Gauss-Seidel subiteration is shown to be an effective means to reduce computational cost without sacrificing performance.

***Index Terms***— Total Least Squares, distributed estimation, wireless sensor networks.

## 1. INTRODUCTION

We consider the problem of parameter estimation under a linear model, in a network with each node acquiring a small number of measurements (possibly just one) and limited to communicating with its immediate neighbors only. The goal is a distributed implementation approaching the performance of the optimal centralized estimator having access to all observations across the network. Distributed ordinary (or *linear*) least squares (OLS) estimation has been well studied, and a variety of implementations are available [1–4]. The OLS estimator is (implicitly or explicitly) based on the assumption that the observations (or *output data*) are noisy but that the regressor (or *input data*) is noise-free. However, there exist situations in which the latter assumption may not hold, and noise is also present in the regressor [5–7]. Under those circumstances, the OLS estimate will be biased in general.

Total Least Squares (TLS) estimation is a well-known alternative when dealing with noisy input and output data not only in numerical analysis or statistics [8, 9], but also in engineering fields like adaptive filtering or spectrum sensing [10, 11]. A distributed TLS scheme was proposed in [12] with proven convergence to the exact TLS solution at each node. Due to the fact that each node must solve a local TLS problem at each iteration, the scheme in [12] has important drawbacks. The first one is that the number of measurements available at each node must be strictly larger than the dimension of the unknown parameter; otherwise the local problem is underdetermined at that node, even if the *global* problem is *overdetermined.* The second drawback of [12] is complexity: to solve its local TLS problem, at each iteration each node must perform an eigenvalue decomposition (EVD) whose order is the dimension of the parameter. The low-complexity variant in [13] replaces the EVD by a single subiteration of the inverse power method, which amounts to solving a linear set of equations of the same order at each iteration. This may still be a demanding task for nodes with limited resources. In addition, the same requirement as in [12] regarding the minimum number of observations per node applies to [13].

We present two novel distributed TLS methods with the following differences with respect to [12, 13]. First and foremost, there is no minimum requirement for the number of measurements per node for either of our methods. Secondly, their complexity is much less than that of [12] and comparable to (and even less than) that of [13]. Our first method runs a number of average consensus iterative processes in parallel and, *after convergence,* each node obtains the estimate from a *single* EVD of certain matrix (in contrast, [12] requires one EVD *per iteration*). The second scheme is EVD-free, but similarly to [13], at each iteration each node must solve a set of linear equations. Replacing this step by a *single* Gauss-Seidel subiteration significantly reduces the load at the nodes, and this is achieved without sacrificing performance. Our methods accommodate a matrix-valued parameter, whereas [12, 13] consider only a vector-valued parameter.

Our framework is akin to [1–4, 12, 13]: each node collects a data snapshot and then estimation is performed based on those; i.e., the data do not 'stream in'. TLS-like estimators with streaming data do exist, usually in a stochastic, adaptive filtering framework: see e.g. [10, 14, 15] for standalone processing, and [16] for a distributed in-network approach. Extending our schemes to streaming data applications will be the subject of future work.

[*]Supported by the Spanish Government, ERDF funds (TEC2010-21245-C02-02/TCM DYNACS, CONSOLIDER-INGENIO 2010 CSD2008-00010 COMONSENS) and the Galician Government (CN 2012/260 AtlantTIC)

[†]Supported by the Spanish Government, ERDF funds (TEC2010-19171 MOSAIC, CONSOLIDER-INGENIO 2010 CSD2008-00010 COMONSENS) and the Catalan Government (2009SGR-01236 AGAUR).

## 2. PROBLEM STATEMENT

Consider the problem of fitting a linear regression model: given data matrices $\boldsymbol{Z} \in \mathbb{R}^{N\times L}$ and $\boldsymbol{Y} \in \mathbb{R}^{N\times M}$, an estimate of the parameter $\boldsymbol{X} \in \mathbb{R}^{L\times M}$ is sought in order to have $\boldsymbol{Z}\boldsymbol{X} \approx \boldsymbol{Y}$. We assume that the system is overdetermined, i.e., $N > L$, so that in general there is no exact solution for $\boldsymbol{X}$. The Ordinary Least Squares (OLS) estimator of $\boldsymbol{X}$ is

$$\hat{\boldsymbol{X}}_{\rm OLS} = \arg\min_{\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{Z}\boldsymbol{X}\|_F^2 = (\boldsymbol{Z}^T\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{Y}, \quad (1)$$

with $\|\cdot\|_F$ the Frobenius norm. In a stochastic setting, (1) provides the Best Linear Unbiased Estimator (BLUE) when $\boldsymbol{Z}$ is noise-free and $\boldsymbol{Y}$ is corrupted by zero-mean white noise; if the noise is Gaussian, then (1) is also the Maximum Likelihood (ML) estimator [17]. When $\boldsymbol{Z}$ is noisy as well, the OLS estimator is biased in general. The TLS estimator is a suitable alternative [8, 9]: it is obtained as the solution to

$$\min_{\boldsymbol{X},\boldsymbol{H}} \|\boldsymbol{Z} - \boldsymbol{H}\|_F^2 + r^2\|\boldsymbol{Y} - \boldsymbol{H}\boldsymbol{X}\|_F^2, \quad (2)$$

where $\boldsymbol{H}$ denotes a candidate noise-free regressor matrix. The scalar $r^2 > 0$ allows to assign different weights to the two terms in the cost function in (2). For small $r^2$, the minimization of (2) w.r.t. $\boldsymbol{H}$ yields $\hat{\boldsymbol{H}} \approx \boldsymbol{Z}$, so the TLS estimate of $\boldsymbol{X}$ will approach the OLS estimate (1). Thus, the noisier the regressor matrix $\boldsymbol{Z}$ is suspected, the larger $r^2$ should be. In a stochastic framework, the TLS estimator is the ML estimator when the noises corrupting $\boldsymbol{Z}$ and $\boldsymbol{Y}$ are zero-mean i.i.d. Gaussian, and the variance of the noise affecting the regressor matrix $\boldsymbol{Z}$ is $r^2$ times that of the noise affecting $\boldsymbol{Y}$.

We seek distributed schemes for solving (2) over a network, in which each node can only communicate with a small subset of nodes (neighbors), and without a "fusion center." The network is assumed *undirected* (if link $i \to j$ from node $i$ to node $j$ exists, then link $j \to i$ exists as well) and *connected* (there exist a path between any pair of nodes). Each node has access to a data subset, i.e., a subset of the rows of $\boldsymbol{Y}$ and $\boldsymbol{Z}$. It is assumed that all nodes agree on the value of the design parameter $r$. We first review the centralized case, in which $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are available at the processing entity.

## 3. CENTRALIZED TLS

The TLS estimate can be found as follows. Define the *extended data matrix* $\boldsymbol{P} \triangleq \left[\begin{array}{cc} \boldsymbol{Z} & r\boldsymbol{Y} \end{array}\right] \in \mathbb{R}^{N\times(L+M)}$, and let $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be its singular value decomposition (SVD), with $\boldsymbol{\Sigma} = \mathrm{diag}(\,\sigma_1\,\cdots\,\sigma_{L+M}\,)$ and $\sigma_1 \le \cdots \le \sigma_{L+M}$ the ordered singular values. Partition $\boldsymbol{V} \in \mathbb{R}^{(L+M)\times(L+M)}$ as

$$\boldsymbol{V} = \left[\begin{array}{cc} \boldsymbol{V}_1 & \boldsymbol{V}_3 \\ \boldsymbol{V}_2 & \boldsymbol{V}_4 \end{array}\right], \quad \boldsymbol{V}_1 \in \mathbb{R}^{L\times M}, \quad \boldsymbol{V}_2 \in \mathbb{R}^{M\times M}. \quad (3)$$

Then, provided $\boldsymbol{V}_2$ is invertible[1], one has (see, e.g., [9]):

$$\hat{\boldsymbol{X}}_{\rm TLS} = -\frac{1}{r}\boldsymbol{V}_1\boldsymbol{V}_2^{-1}. \quad (4)$$

Thus, finding the TLS estimate amounts to computing an SVD. Alternatively, note that the cost (2) is quadratic in each of the variables $\boldsymbol{X}$ and $\boldsymbol{H}$, so that a *cyclic minimization* (CM) iterative scheme can be readily applied [18]. The cost (2) is minimized w.r.t $\boldsymbol{H}$ holding $\boldsymbol{X}$ fixed; then, using the obtained value of $\boldsymbol{H}$, minimization is performed w.r.t. $\boldsymbol{X}$, and the process is repeated. Equating to zero the gradients of (2) w.r.t. $\boldsymbol{X}$ and $\boldsymbol{H}$, each variable can be found in terms of the other:

$$\begin{aligned} \boldsymbol{X} &= (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{Y}, & (5)\\ \boldsymbol{H} &= (\boldsymbol{Z} + r^2\boldsymbol{Y}\boldsymbol{X}^T)(\boldsymbol{I} + r^2\boldsymbol{X}\boldsymbol{X}^T)^{-1} \\ &= \boldsymbol{Z} + (\boldsymbol{Y} - \boldsymbol{Z}\boldsymbol{X})\boldsymbol{S}\{\boldsymbol{X}\}\boldsymbol{X}^T, & (6) \end{aligned}$$

where in (6) we have used the matrix inversion lemma, with

$$\boldsymbol{S}\{\boldsymbol{X}\} \triangleq r^2(\boldsymbol{I} + r^2\boldsymbol{X}^T\boldsymbol{X})^{-1} \in \mathbb{R}^{M\times M}. \quad (7)$$

The proposed CM iteration proceeds as follows. Starting with some initial $\hat{\boldsymbol{X}}^{(0)}$, and for $j = 1, 2, \ldots$, compute

$$\hat{\boldsymbol{H}}^{(j)} = \boldsymbol{Z} + (\boldsymbol{Y} - \boldsymbol{Z}\hat{\boldsymbol{X}}^{(j-1)})\boldsymbol{S}\{\hat{\boldsymbol{X}}^{(j-1)}\}(\hat{\boldsymbol{X}}^{(j-1)})^T, \quad (8)$$

and then solve for $\hat{\boldsymbol{X}}^{(j)}$ in the linear system

$$(\hat{\boldsymbol{H}}^{(j)})^T\hat{\boldsymbol{H}}^{(j)}\hat{\boldsymbol{X}}^{(j)} = (\hat{\boldsymbol{H}}^{(j)})^T\boldsymbol{Y}. \quad (9)$$

Incidentally, note that by choosing $\hat{\boldsymbol{X}}^{(0)} = \boldsymbol{0}$, then after the first iteration we obtain $\hat{\boldsymbol{H}}^{(1)} = \boldsymbol{Z}$ and $\hat{\boldsymbol{X}}^{(1)} = \hat{\boldsymbol{X}}_{\rm OLS}$.

Iteration (8)-(9) yields a sequence of monotonically non-increasing values of the non-negative cost (2), so that (8)-(9) is convergent in terms of cost function. However, one cannot claim global convergence: for example, if $\hat{\boldsymbol{X}}^{(0)}$ lies near a saddle point[2] of the cost (2), then the sequence $\hat{\boldsymbol{X}}^{(j)}$ may diverge, even though convergence in the cost function takes place. Nevertheless, empirical experience suggests that taking $\hat{\boldsymbol{X}}^{(0)} = \boldsymbol{0}$ (equivalently, $\hat{\boldsymbol{X}}^{(1)} = \hat{\boldsymbol{X}}_{\rm OLS}$) is enough to set the iteration within the domain of attraction of the TLS solution.

The CM iteration (8)-(9) avoids the SVD of the $N \times (L + M)$ extended data matrix, replacing it by a sequence of steps. At each of these steps, two sets of linear equations, of sizes $M \times M$ and $L \times L$ respectively, must be solved. These direct and iterative approaches to obtaining $\hat{\boldsymbol{X}}_{\rm TLS}$ will constitute the basis for the distributed TLS algorithms in the next section.

## 4. DISTRIBUTED TLS ALGORITHMS

The methods in Sec. 3 for computing the TLS estimate (either directly as per (4), or via the CM iteration (8)-(9)) are centralized, in the sense that they make use of the whole dataset

[1] Otherwise the TLS estimate does not exist [9]. It will be assumed throughout that $\boldsymbol{V}_2$ is invertible.

[2] The cost in (2), which is not convex, has no local minima or maxima, as can be shown following steps similar to those in [19].

$(\boldsymbol{Y}, \boldsymbol{Z})$. We now focus on distributed schemes in which at each of the $N$ nodes in the network only a single data point is available[3]. That is, if we partition $\boldsymbol{Y}$, $\boldsymbol{Z}$ row-wise as

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1^T \\ \vdots \\ \boldsymbol{y}_N^T \end{bmatrix}, \quad \boldsymbol{Z} = \begin{bmatrix} \boldsymbol{z}_1^T \\ \vdots \\ \boldsymbol{z}_N^T \end{bmatrix}, \tag{10}$$

then the data available at node $i$ is $(\boldsymbol{y}_i, \boldsymbol{z}_i)$. The two methods proposed below involve a symmetric *weight matrix* $\boldsymbol{W} \in \mathbb{R}^{N\times N}$ with elements $W_{ij} \neq 0$ only if nodes $i$ and $j$ are neighbors[4], and satisfying (see [2, 20])

$$\mathbf{1}^T\boldsymbol{W} = \mathbf{1}^T, \quad \boldsymbol{W}\mathbf{1} = \mathbf{1}, \quad \rho\left(\boldsymbol{W} - \tfrac{1}{N}\mathbf{1}\mathbf{1}^T\right) < 1, \tag{11}$$

with $\mathbf{1} \in \mathbb{R}^N$ the all-ones vector, and $\rho(\cdot)$ the spectral radius.

### 4.1. First implementation (DSC)

The distributed standard consensus (DSC) based scheme is similar in spirit to the method in [1] for distributed OLS estimation. Note that $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T$, i.e., the $l$-th column of $\boldsymbol{V}$ is an eigenvector of $\boldsymbol{P}^T\boldsymbol{P}$ associated to the eigenvalue $\sigma_l^2$. Since $\boldsymbol{P} = [\ \boldsymbol{Z}\ \ r\boldsymbol{Y}\ ]$, it follows that $\boldsymbol{P}^T\boldsymbol{P}$ can be written as

$$\boldsymbol{P}^T\boldsymbol{P} = \begin{bmatrix} \sum_{i=1}^N \boldsymbol{z}_i\boldsymbol{z}_i^T & r\sum_{i=1}^N \boldsymbol{z}_i\boldsymbol{y}_i^T \\ r\sum_{i=1}^N \boldsymbol{y}_i\boldsymbol{z}_i^T & r^2\sum_{i=1}^N \boldsymbol{y}_i\boldsymbol{y}_i^T \end{bmatrix}, \tag{12}$$

which can be obtained by standard distributed average consensus methods [1, 2, 20], as follows. Node $i$ updates a symmetric matrix $\boldsymbol{G}_i(k)$ which is initialized as

$$\boldsymbol{G}_i(0) = \begin{bmatrix} \boldsymbol{z}_i\boldsymbol{z}_i^T & r\boldsymbol{z}_i\boldsymbol{y}_i^T \\ r\boldsymbol{y}_i\boldsymbol{z}_i^T & r^2\boldsymbol{y}_i\boldsymbol{y}_i^T \end{bmatrix} \in \mathbb{R}^{(L+M)\times(L+M)}. \tag{13}$$

Let $g_{pq}^{(i)}(k)$ denote the $(p,q)$-element of $\boldsymbol{G}_i(k)$, and let

$$\boldsymbol{g}_{pq}(k) \triangleq \begin{bmatrix} g_{pq}^{(1)}(k) & g_{pq}^{(2)}(k) & \cdots & g_{pq}^{(N)}(k) \end{bmatrix}^T. \tag{14}$$

The consensus iteration can be written as

$$\boldsymbol{g}_{pq}(k) = \boldsymbol{W}\boldsymbol{g}_{pq}(k-1), \quad k = 1, 2\ldots, \tag{15}$$

involving communication among neighbors only. Under (11), (15) converges and $\lim_{k\to\infty} g_{pq}^{(i)}(k) = \frac{1}{N}\sum_{j=1}^N g_{pq}^{(j)}(0)$ for all $i$ [1, 2]. Hence, one has $\lim_{k\to\infty} \boldsymbol{G}_i(k) = \frac{1}{N}\boldsymbol{P}^T\boldsymbol{P}$ for all $i$. Upon convergence of these $\frac{1}{2}((L+M)^2 + (L+M))$ consensus iterations running in parallel, node $i$ extracts the eigenvectors associated to the $M$ smallest eigenvalues of $\boldsymbol{G}_i$, from which $\hat{\boldsymbol{X}}_{\text{TLS}}$ is obtained by means of (4).

[3] Generalization to settings in which some sensors may have access to several data points simultaneously is straightforward.

[4] By convention, every node is neighbor to itself.

### 4.2. Second implementation (DCM / DCM-GS)

The DSC requires, as a final step, that each node compute the EVD of an $(L+M)\times(L+M)$ matrix. Depending on the nodes' processing limitations, it may be desirable to avoid this step. Inspired by the centralized CM approach of Sec. 3, we present next an EVD-free distributed implementation. At time $k$, node $i$ keeps local copies of the following variables:

$$\hat{\boldsymbol{h}}_i(k) \in \mathbb{R}^L, \quad \hat{\boldsymbol{X}}_i(k), \boldsymbol{\Psi}_i(k) \in \mathbb{R}^{L\times M}, \quad \boldsymbol{\Phi}_i(k) \in \mathbb{R}^{L\times L}. \tag{16}$$

The row vector $\hat{\boldsymbol{h}}_i^T(k)$ constitutes a (local) estimate at node $i$ and time $k$ of the $i$-th row of the noise-free regressor matrix $\boldsymbol{H}$, whereas $\hat{\boldsymbol{X}}_i(k)$ is the corresponding local estimate of $\boldsymbol{X}$. The matrices $\boldsymbol{\Phi}_i(k)$, $\boldsymbol{\Psi}_i(k)$ are equivalent to the matrices $\hat{\boldsymbol{H}}^T\hat{\boldsymbol{H}}$ and $\hat{\boldsymbol{H}}^T\boldsymbol{Y}$ of (9) but computed locally at node $i$.

The quantities in (16) are successively updated using information gathered by local exchanges between neighbors. For $k \geq 1$, node $i$ performs the following. First, given $\hat{\boldsymbol{X}}_i(k)$, the local estimate $\hat{\boldsymbol{h}}_i(k)$ is updated:

$$\hat{\boldsymbol{h}}_i(k) = \boldsymbol{z}_i + \hat{\boldsymbol{X}}_i(k)\boldsymbol{S}\{\hat{\boldsymbol{X}}_i(k)\}(\boldsymbol{y}_i - \hat{\boldsymbol{X}}_i^T(k)\boldsymbol{z}_i). \tag{17}$$

Computing the right-hand side of (17) involves[5] solving an $M\times M$ set of linear equations. Note that (17) can be interpreted as a local version of the centralized CM step (8), applied to the $i$-th row of the regressor matrix estimate, and based on the current local estimate $\hat{\boldsymbol{X}}_i(k)$.

To implement a local version of the centralized CM step (9), note that the global quantities $\hat{\boldsymbol{H}}^T\hat{\boldsymbol{H}}$, $\hat{\boldsymbol{H}}^T\boldsymbol{Y}$ in (9) can be written as summations over local quantities. With $\boldsymbol{W}$ a symmetric weight matrix as in (11), and inspired by so-called "consensus+innovations" approaches to distributed estimation [21], this suggests the following update for the variables $\boldsymbol{\Phi}_i(k)$, $\boldsymbol{\Psi}_i(k)$ at node $i$, using information from its neighbors:

$$\boldsymbol{\Phi}_i(k) = \sum_j W_{ij}\left(\beta_k\boldsymbol{\Phi}_j(k-1) + \alpha_k\hat{\boldsymbol{h}}_j(k)\hat{\boldsymbol{h}}_j^T(k)\right), \tag{18}$$

$$\boldsymbol{\Psi}_i(k) = \sum_j W_{ij}\left(\beta_k\boldsymbol{\Psi}_j(k-1) + \alpha_k\hat{\boldsymbol{h}}_j(k)\boldsymbol{y}_j^T\right). \tag{19}$$

Finally, node $i$ solves for $\hat{\boldsymbol{X}}_i(k+1)$ in

$$\boldsymbol{\Phi}_i(k)\hat{\boldsymbol{X}}_i(k+1) = \boldsymbol{\Psi}_i(k). \tag{20}$$

The iteration is initialized with $\hat{\boldsymbol{X}}_i(1) = \mathbf{0}$ for all $i$, thus $\hat{\boldsymbol{h}}_i(1) = \boldsymbol{z}_i$, $\boldsymbol{\Phi}_i(1) = \sum_j W_{ij}\boldsymbol{z}_j\boldsymbol{z}_j^T$, $\boldsymbol{\Psi}_i(1) = \sum_j W_{ij}\boldsymbol{z}_j\boldsymbol{y}_j^T$.

The sequences $\{\beta_k\}$, $\{\alpha_k\}$ are suitable time-varying weights for the consensus and innovation terms, respectively, in (18)-(19); here, by *innovation* we mean the new information supplied by the update of the $\hat{\boldsymbol{h}}_j$'s (as opposed to that supplied by new observations, as in [21]). Possible choices for these weight factors include (see, e.g., [21, 22]):

$$\alpha_k = \frac{1}{k}, \qquad \beta_k = 1 - \frac{1}{k^\delta}, \quad 0 < \delta < 1, \quad k \geq 1. \tag{21}$$

[5] Recall the definition of the matrix-valued function $\boldsymbol{S}\{\cdot\}$ in (7).

Since typically one has $L \gg M$, the computational load of (17)-(20), referred to as DCM, is dominated by the need to solve the $L \times L$ linear system (20) at each $k$ by all nodes. One possible way to solve (20) is by means of a Gauss-Seidel (GS) iterative process; since $\boldsymbol{\Phi}_i(k)$ is symmetric positive definite, the GS method is convergent [8]. In order to reduce the computational requirements at the nodes, we propose to replace step (20) by a *single* GS subiteration, taking as initial point the estimate $\hat{\boldsymbol{X}}_i(k)$ from the previous step. Specifically, if we decompose $\boldsymbol{\Phi}_i(k) = \boldsymbol{L}_i(k) + \boldsymbol{U}_i(k)$ into a lower triangular component $\boldsymbol{L}_i(k)$ and a strictly upper triangular component $\boldsymbol{U}_i(k)$, then (20) is substituted by

$$\boldsymbol{L}_i(k)\hat{\boldsymbol{X}}_i(k+1) = \boldsymbol{\Psi}_i(k) - \boldsymbol{U}_i(k)\hat{\boldsymbol{X}}_i(k), \qquad (22)$$

in which we must solve for $\hat{\boldsymbol{X}}_i(k+1)$. This is done cheaply via forward substitution, since $\boldsymbol{L}_i(k)$ is lower triangular. Simulation results show no significant loss in performance if (20) is replaced by (22). We refer to this variant as DCM-GS.

## 5. NUMERICAL RESULTS

We consider two example networks, each with $N = 100$ nodes randomly deployed over a unit square, with two different connectivity radii $r_c = \{0.25, 0.18\}$. The distributed schemes are run over each of these networks using a Metropolis weight matrix $\boldsymbol{W}$ [1]. For DCM and DCM-GS we take $\delta = 0.8$. The parameter vector $\boldsymbol{x} \in \mathbb{R}^{L\times 1}$ with $L = 5$, $M = 1$, is randomly generated and fixed throughout the simulation. Each node has access to a single measurement; these are generated as $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}$ and $\boldsymbol{Z} = \boldsymbol{H} + \boldsymbol{E}$, with $\boldsymbol{n}$, $\boldsymbol{E}$ the corrupting noises (zero-mean i.i.d. Gaussian entries with variance $\sigma^2$). We set $r = 1$, reflecting prior knowledge about the entries of $\boldsymbol{n}$ and $\boldsymbol{E}$ having the same variance. The matrix $\boldsymbol{H}$ is randomly generated in each run with zero-mean i.i.d. Gaussian entries. Conditioned on $\boldsymbol{H}$, the signal-to-noise ratio (SNR) is

$$\text{SNR} = \frac{\boldsymbol{x}^T\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{x} + \text{tr}(\boldsymbol{H}^T\boldsymbol{H})}{N(L+1)\sigma^2} \leq \frac{(1+\|\boldsymbol{x}\|^2)\|\boldsymbol{H}\|_F^2}{N(L+1)\sigma^2} \qquad (23)$$

We take the upper bound (23) as the SNR in the simulations, as it only depends on $\|\boldsymbol{H}\|_F$ and $\|\boldsymbol{x}\|$. The performance metric considered is the normalized mean square error, defined as

$$\text{NMSE}\{\hat{\boldsymbol{x}}(k)\} = \frac{1}{N\|\boldsymbol{x}\|^2}\sum_{i=1}^{N}\mathbb{E}\left[\|\hat{\boldsymbol{x}}_i(k) - \boldsymbol{x}\|_2^2\right],$$

averaged over 500 independent realizations for each network.

Fig. 1 shows the NMSE as a function of the iteration index $k$ (up to $K = 200$ iterations) for the distributed schemes, and for both deployments, with SNR $= 10$ dB. The performance of the centralized TLS (both direct and iterative approaches) and OLS estimators is included as a reference. Note that the centralized CM scheme (run for 200 iterations) achieves the same performance of the direct TLS estimator, i.e., no convergence problems are observed. The larger NMSE of the centralized OLS estimator is mainly due to its bias.

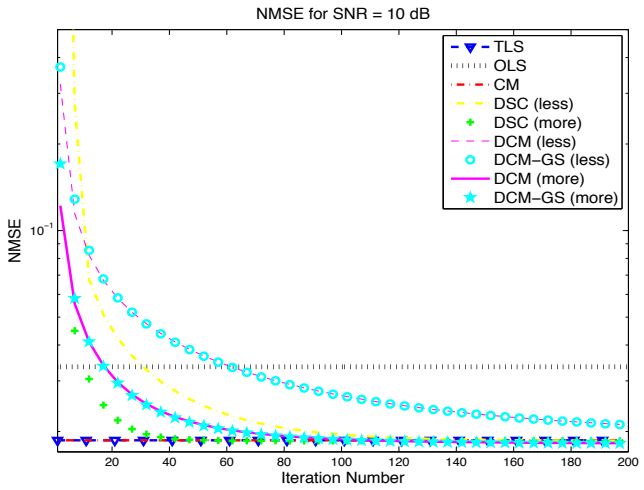


**Fig. 1**. NMSE vs. $k$ in two different deployments with $N = 100$: $r_c = 0.18$ (*"less"*) and $0.25$ (*"more"*).

The total computational cost per node of the distributed methods is the product of the number of iterations $K$ until convergence and the computational cost at each node. Assuming $L \gg M$, the total cost is $K_{\text{DSC}} \times O(L^2)$ plus an additional $O(L^3)$ EVD for DSC; $K_{\text{DCM}} \times O(L^3)$ for DCM; and $K_{\text{DCM-GS}} \times O(L)$ for DCM-GS. Note that the convergence rate of the DCM and DCM-GS methods are the same, i.e., $K_{\text{DCM-GS}} \approx K_{\text{DCM}}$. As DSC converges faster than DCM and DCM-GS, it seems to be the preferred choice whenever the computational load of $O(L^2)$ per iteration, plus the final EVD, are feasible at each node and the network is not highly connected. However, for densely connected deployments and large $L$, DCM-GS provides the lowest computational cost.

The three distributed schemes converge faster for the more densely connected deployment, as could be expected. It may seem surprising that the curves of DCM and DCM-GS for the dense deployment remain slightly below the NMSE of the centralized TLS estimator for $k > 80$. This is explained by two facts: (i) the evolution of the NMSE need not be monotonically decreasing, and (ii) the steady state has not been reached yet for $K = 200$ as in Fig. 1.

## 6. CONCLUSIONS

We have presented novel low-complexity algorithms for distributed TLS estimation which, in contrast with previous approaches, do not impose a minimum requirement on the number of available observations per node. One of them (DSC) is based on standard consensus and performs a single EVD upon convergence, whereas the other (DCM) solves a linear system at each iteration. Replacing this latter requirement by a single Gauss-Seidel subiteration, the computational load of DCM is significantly reduced without sacrificing performance.

## 7. REFERENCES


[1] L. Xiao, S. Boyd, S. Lall, "A scheme for asynchronuous distributed sensor fusion based on average consensus," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw.,* pp. 63–70, 2005.

[2] R. Olfati-Saber, J. A. Fax, R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE,* vol. 95, no. 1, pp. 215–233, Jan. 2007.

[3] H. Zhu, A. Cano, G. B. Giannakis, "Distributed consensus-based demodulation: algorithms and error analysis," *IEEE Trans. Wireless Commun.,* vol. 9, no. 6, pp. 2044–2054, Jun. 2010.

[4] H. Paul, J. Fliege, A. Dekorsky, "In-network-processing: distributed consensus-based linear estimation," *IEEE Commun. Lett.,* vol. 17, no. 1, pp. 59–62, Jan. 2013.

[5] Y. C. Eldar, A. Ben-Tal, A. Nemirovski, "Robust mean-squared error estimation in the presence of model uncertainties," *IEEE Trans. Signal Process.,* vol. 53, no. 1, pp. 168–181, Jan. 2005.

[6] A. Wiesel, Y. Eldar, A. Yeredor, "Linear regression with Gaussian model uncertainty: Algorithms and bounds," *IEEE Trans. Signal Process.,* vol. 56, no. 6, pp. 2194–2205, Jun. 2008.

[7] T. Söderström, "System identification for the errors-in-variables problem," *Trans. Inst. Meas. Control,* vol. 34, no. 7, pp. 780–792, Oct. 2012.

[8] G. H. Golub, C. Van Loan, *Matrix Computations.* The Johns Hopkins University Press, 2nd ed., 1989.

[9] I. Markovsky, S. Van Huffel, "Overview of Total Least Squares methods," *Signal Process.,* vol. 87, no. 10, pp. 2283–2302, 2007.

[10] C. E. Davila, "An efficient Total Least Squares algorithm for FIR adaptive filtering," *IEEE Trans. Signal Process.,* vol. 42, no. 2, pp. 268–280, Feb. 1994.

[11] E. Dall'Anese and G.B. Giannakis, "Distributed cognitive spectrum sensing via group sparse Total Least-Squares," *Proc. IEEE CAMSAP,* pp. 341–344, 2011.

[12] A. Bertrand, M. Moonen, "Consensus-based distributed Total Least Squares estimation in *ad hoc* wireless sensor networks," *IEEE Trans. Signal Process.,* vol. 59, no. 5, pp. 2320–2330, May 2011.

[13] A. Bertrand, M. Moonen, "Low-complexity distributed Total Least Squares estimation in ad hoc sensor networks," *IEEE Trans. Signal Process.,* vol. 60, no. 8, pp. 4321–4333, Aug. 2012.

[14] D.-Z. Feng, X.-D. Zhang, D.-X. Chang, W. X. Zheng, "A fast recursive Total Least Squares algorithm for adaptive FIR filtering," *IEEE Trans. Signal Process.,* vol. 52, no. 10, pp. 2729–2737, Oct. 2004.

[15] R. Arablouei, K. Doğançay, "Linearly-constrained Recursive Total Least-Squares algorithm," *IEEE Signal Process. Lett.,* vol. 19, no. 12, pp. 821–824, Dec. 2012.

[16] R. Arablouei, S. Werner, K. Doğançay, "Diffusion-based distributed adaptive estimation utilizing gradient-descent Total Least Squares," in *Proc. IEEE ICASSP,* pp. 5308–5312, May 2013.

[17] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[18] P. Stoica, Y. Selén, "Cyclic minimizers, majorization techniques, and the expectation–maximization algorithm: a refresher," *IEEE Signal Process. Mag.,* vol. 21, no. 1, pp. 112–114, Jan. 2004.

[19] N. Srebro, T. Jaakkola, "Weighted low-rank approximations," in *20th Int. Conf. Machine Learning,* pp. 720–727, AAAI Press, 2003.

[20] L. Xiao, S. Boyd, "Fast linear iterations for distributed averaging," in *Proc. IEEE Conf. Decision and Control*, vol. 5, pp. 4997–5002, Dec. 2003.

[21] S. Kar and J. M. F. Moura, "Consensus + innovations distributed inference over networks," *IEEE Signal Process. Mag.,* vol. 30, no. 3, pp. 99–109, May 2013.

[22] S. Silva Pereira, R. López-Valcarce, A. Pagès-Zamora, "A diffusion-based EM algorithm for distributed estimation in unreliable sensor networks", *IEEE Signal Process. Lett.*, vol. 20, pp 20, 595–598, June 2013.