DISTRIBUTED DATA DIMENSIONALITY REDUCTION AND DENOISING

by

ABIODUN T. ADUROJA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2013

To my parents Engr. and Mrs. Aduroja

who sacrificed so much for me.

ACKNOWLEDGEMENTS

A lot of people have contributed to the success and completion of my thesis, from whom I have learnt a lot of lessons which will be quite invaluable for me in my future pursuits.

Firstly, I would like to thank my supervising Professor; Dr. I. D. Schizas for his mentorship, invaluable pieces of advice during the course of my research. Dr. Schizas was there to encourage and guide me through sharpening my research skills. I also would like to thank Dr. Schizas for giving me the opportunity to conduct this research work under him and more importantly for the financial assistance that I also received. It really went a long way in reducing my financial burdens.

I wish to also thank Dr. Qilian Liang and Dr. Saibun Tjuatja and my graduate advisor Dr. W. Alan Davis for taking out time out of their busy schedule to serve on my thesis defense committee.

I would also like to extend my appreciation to my other research group members; Guohua Ren and Jia Chen for your support, encouragement and even the helpful discussions we had about the research work.

Finally, I would like to acknowledge my brother; Tobi Aduroja's support and words of inspiration during the course of my graduate studies. I say thank you.

November 22, 2013

ABSTRACT

DISTRIBUTED DATA DIMENSIONALITY REDUCTION AND DENOISING

ABIODUN T. ADUROJA, M. Sc.

The University of Texas at Arlington, 2013

Supervising Professor: Ioannis D. Schizas

With the recent upsurge in the amount of data transmission and use, giving rise to the era of big data, research has put a lot of emphasis on designing efficient methods to extract useful information by reducing the size of a large set of data to a much smaller set. This is the main idea in data dimensionality reduction which is a critical step in applications that involve a constantly increasing data size, and limited computational and communication resources. Such a situation appears in sensor networks where there is a large number of sensors acquiring data. However, the informative part of the sensor data in practice is created by a small number of sources/phenomena of interest and the dimension of data can be effectively reduced.

Data dimensionality reduction by employing principal component analysis has proven to be extremely useful. Traditional principal component analysis involves principal subspace estimation obtained by a decomposition of the data covariance matrix in a centralized fashion. However, such centralized approaches do not account for the fact that sensors are spatially scattered while they may have to communication through noisy links. This thesis puts forth a distributed algorithmic framework for estimating the principal eigenspace of the sensor data without the need of a central fusion center. Each sensor is only responsible

for estimating only a small part of the principal subspace, and it only communicates with sensors in its near vicinity, the so called single-hop neighbors. Toward this end, the standard principal component analysis framework is reformulated as a separable constrained minimization problem which is solved by utilizing coordinate descent techniques combined with the alternating direction method of multipliers. Computationally simple local updating recursions are obtained that involve only single-hop inter-sensor communications and allow sensors to estimate the principal covariance eigenspace in a distributed fashion. Two different type of algorithms are obtained that can applied in different sensor network architectures.

The proposed distributed principal eigenspace estimation framework is used in a denoising application, where the sensor data are projected onto the estimated principal space to reduce the noise and improve overall data quality. To this end, a novel distributed denoising scheme is presented in this thesis. Extensive numerical tests using both synthetic and real data demonstrate that the proposed framework has the potential to achieve a considerably faster convergence rate and better steady-state estimation performance compared to existing alternatives. Further, the proposed distributed algorithms exhibit robustness in the presence of inter-sensor communication noise whereas alternative techniques do not converge in the presence of random perturbations.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

Data acquired by sensors that sense a particular field are usually confined into a subspace of dimensionality. Efficient in-network processing of the sensor data collected is a major strategy that can enable sensor networks to extract useful information from the acquired data. Since not all data acquired is usually needed, estimation of the *principal* signal subspace of the sensor data is crucial in reducing the memory and computational requirements imposed by a huge amount of data. Apart from this, sensors have limited resources in terms of energy, computational power and data throughput. Radio communication in particular is an energy consuming process and it is one of the primary reasons in sensor node's battery depletion [1]. The reduction in the quantity of the data being transmitted is therefore essential in designing sensor networks [2]. Sensor data in practice tend to exhibit temporal and spatial correlations that can be exploited to reduce their dimensionality. The workhorse of recovering the low dimensional signal subspace corresponding to a large set of data is known as Principal Component Analysis (PCA), see e.g., [3]. This method involves estimating the eigenvectors and eigenvalues of the data covariance matrix which can be estimated via sample-averaging of data acquired over a time period. PCA aims to keep estimate a few covariance eigenvectors, the principal eigenvectors, that describe in a compact way the low-dimensional signal subspace where the sensor data lie on. The data acquired are then projected onto the principal eigenspace to obtain a low-dimensional representation. This projection, which is of reduced dimension, finds application in data dimensionality reduction, image denoising and sensor data denoising on an mean square

error (MSE) optimal basis [4–6]. The topic of this thesis is the design and testing of improved distributed algorithms that have the ability to estimate the principal eigenspace over a network of spatially scattered sensors. Combining statistics with message passing techniques an computationally efficient and effective distributed algorithmic framework is put forth that has the ability to outperform existing alternatives when it comes to distributed dimensionality and denoising applications.

1.1.1 Sensor Networks

Recent technological improvements in wireless communications have made the development of small, inexpensive, low-power, distributed devices, which are capable of local processing and wireless communication, possible. Such nodes known as sensors are capable of only a limited amount of processing. Networks of sensors offer spatial diversity when acquiring data in the sense that sensing is carried out at multiple locations, thus the danger of having severely degraded by shadowing measurements when sensing at a single location is eliminated. Thus, multiple sensing nodes that occupy different locations can really lead to improved accuracy and robustness. When coordination/collaboration takes place among multiple sensors that share their information, networks of sensors have the ability to measure a given physical environment in great detail. Thus, a sensor network can be described as a collection of sensor nodes which collaborate to perform some specific action [7]. Figure 1.1 shows a graphical representation of ad-hoc (structure-less) sensor network with 13 sensors/nodes.

The are different basic network topologies,namely; fully connected, mesh, star, ring, tree, as well as ad-hoc networks with such structure as the one provided in 1.1. A single network of sensors may consist of the following different interconnected topologies [8].

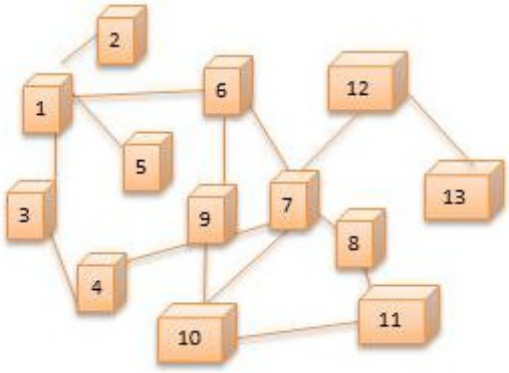

Figure 1.1. A ad-hoc sensor network.

1. Fully connected networks basically face complexity problems because as additional nodes are added, the number of links increases exponentially making the routing problem computationally complex.
2. Mesh networks have nodes that are identical permitting communication to their nearest neighbors. They are also called peer-to-peer nets. This topology is robust to both node and link failures. An advantage of mesh nets is that, although all nodes may be identical and have the same computing and transmission capabilities, certain nodes can be designated as bridges that take on additional functions. Similar properties are shared by ad hoc networks too.
3. Star topology involves connection of all sensors to a fusion center that gathers the sensor data. The fusion center does the information processing. If there is a failure in a communication link, only one node is affected. However, if the fusion center breaks down, then the whole network fails.

4. Ring topology involves all nodes performing the same function without a central or a 'bridge' sensor. Messages generally travel around the ring in a single direction. If there is a failure along one of the communication links, the whole network fails.

1.1.2 Distributed Sensing

Traditional sensor networks consisted of small number of sensor nodes that were wired to a central fusion center. However, the focus recently has been on wireless, distributed, ad hoc sensor networks. Sensing in a distributed fashion allows for closer placement to the parameter being measured than a single sensor would permit especially when the exact location of that particular deterministic quantity is unknown. Multiple sensor nodes are also required to overcome environmental obstacles like obstructions, line of sight constraints etc [9]. Typically, the environment to be monitored does not have an existing infrastructure for either energy or communication. As such, factors such as energy,communication cost and memory capacity becomes integral to any network of sensors. In terms of distributed processing, since communication consumes a lot of energy, centralized sensing systems would require huge communication costs to communicate over long distances resulting in energy depletion.This also suffers from communication latency, in which case is the time it takes information to be transferred to a central processing node. This makes localized computation important as well. [7]

Sensor Networks have found huge applications in environmental monitoring - involving monitoring quantities such as air soil and water, condition based maintenance, habitat monitoring, seismic detection, military surveillance, inventory tracking etc.It is worth noting that, because of the pervasive nature of micro-sensors, sensor networks have the potential to revolutionize the ways data collection and model development are used to construct complex physical systems [10]. Despite its applications, sensor network poses some challenges in terms of ease of ad hoc deployment, performance in an unattended operation,

sensor outage detection and compensation. Hence, the task of optimizing sensor network communication have been have extensively studied with regards to energy efficiency, computation localization, and minimizing routing [7, 11].

1.1.3 Principal Component Analysis (PCA)

PCA is a statistical technique for simplifying a dataset. It was first introduced by Pearson(1901) [12] as a method of linear regression through geometric derivation. Later, it was developed and got its current name through Hotelling(1933) [13, 14] when it was first presented in algebraic form. Hotelling derived it as a technique for analysis of correlation between many random variables. PCA aims to reduce the dimensionality of a set of data whilst preserving as much of the relevant information as possible. It also relies entirely on the input data without reference to the corresponding target data [15] (the criterion to be minimized is the reconstruction mean-square error) [16].

This is achieved by transforming to a new set of variables; the principal components(PC) that are uncorrelated, linear functions of the original variables, and the greatest variance by any projection of the data comes to lie on the first coordinate, the second greatest variance on the second coordinate, and so on. This is done practically by computing the covariance matrix for the full data set. Next, the eigenvectors and eigenvalues of the covariance matrix are computed, and sorted according to decreasing eigenvalue [15]. These eigenvalues give the variances of their respective principal components, and the ratio of the sum of the first say p eigenvalues to the sum of the variances of all say r original variables represents the proportion of the total variance in the original dataset, accounted for by the first p principal components.

PCA has been extended to non-linear applications such as approach to multivariate analysis [17] and also implemented in neural networks [18].

1.1.4 Previous work

PCA has been developed for different settings that include the aggregation of data to a central processing unit [5, 6, 19]. But sensor data are generally scattered among sensors in a sensor network,some other approaches were developed such as the fusion center-based algorithm in [20] where all sensors are assumed to be fully connected and each node is able to transmit its measurements to a FC. Now the FC performs all computation including computing the covariance matrix, performing the singular value decomposition(SVD), and subsequently extracting the principal components. However, performing such computation requires a lot of computational capabilities and communication bandwidth. Also, there is potentially a single point of failure in the sensor network, i.e. once the fusion center is down, the whole network is down. Consequently, decentralized methods have been developed. Partially decentralized PCA algorithms have been proposed either by local computation [20–23] or by data aggregation. However, these still required the presence of a fusion center for merging measurements. The work in [24] puts forth a distributed Karhunen-Loeve transform considering distributed compression and source coding, but convergence is generally not guaranteed while again a fusion center is needed. The work in [25] developed a distributed algorithm, but this is dependent on the assumption that the covariance matrix is decomposable. Distributed PCA algorithm with in-network processing capabilities were proposed in [26, 27]. However, the approach in [27] relies on the fact that the sensors are either fully connected or there is a special tree structure. The approach in [26] relies on consensus averaging techniques, see e.g., [28], but it is well know that these diverge in the presence of noise in inter-sensor links. In this thesis, an efficient, distributed PCA setting is considered that has the potential to converge faster than [26] while demonstrating better steady-state performance. Also, this algorithm is robust both to communication and observation noise in contrast to [26] which is not as resilient and may diverge since it relies on consensus techniques.

1.1.5 Contributions of the thesis

The contributions of this thesis include:

1) Developing a distributed algorithm for dimensionality reduction in sensor networks with a few sensors more powerful than others robust to noise across communication links.

2) Developing a distributed algorithm for dimensionality reduction in sensor networks with all sensors having the same capabilities robust to noise across communication links.

3) Introducing a distributed algorithm for denoising data acquired during sensor measurements. 4) The proposed algorithmic framework has the potential to achieve better convergence rates and lower steady-state estimation errors than existing alternatives. 5) It exhibits robustness in the presence of inter-sensor communication noise, while other approaches do not exhibit robustness and diverge.

1.1.6 Outline of Work

This research work proposes different distributed techniques for dimensionality reduction and denoising. Before designing these algorithms, the sensor network, the motivation for the distributive framework as well as the PCA framework are introduced. Chapter 2 focuses on dimensionality reduction in sensor networks having a few computationally strong sensors-called bridges and a distributed PCA framework is put forth. Chapter 3 focuses on dimensionality reduction in those sensor networks where all sensors have the same capabilities. Chapter 4 illustrates how the proposed algorithmic framework proposed in Chapters 2 and 3 can be used for data denoising to remove as much sensing noise as possible after acquiring data across sensors. Chapter 5 provides extensive numerical tests using both real and synthetic data, and how compare the performance of our framework with existing alternatives. Chapter 6 concludes by showing the extent of work done, while highlighting future research directions that we are pursuing.

CHAPTER 2

A DISTRIBUTED FRAMEWORK FOR PERFORMING PRINCIPAL COMPONENT ANALYSIS

Estimation of the principal eigenspace of a data covariance matrix is instrumental in applications such as data dimensionality reduction and denoising. In sensor networks the acquired data are spatially scattered which further calls for the development of distributed principal subspace estimation algorithms. Toward this end, the standard principal component analysis framework is reformulated as a separable constrained minimization problem which is solved by utilizing coordinate descent techniques combined with the alternating direction method of multipliers. Computationally simple local updating recursions are obtained that involve only single-hop inter-sensor communications and allow sensors to estimate the principal covariance eigenspace in a distributed fashion.

2.1 Problem Statement

Consider an ad hoc sensor network with p sensors in which each sensor is capable to exchange messages only with other sensors within its tranmission/reception range via single-hop communications. The single-hop neighborhood of a sensors j is denoted by $\mathcal{N}_j$, and its cardinality (number of single-hop neighbors for sensor j) is denoted by $|\mathcal{N}_j|$. Assuming links in the sensor network are symmetric, we model the sensor network as an undirected connected graph whose vertices correspond to the sensors and the edges represent single-hop communication links. We characterize this network by an adjacency matrix $\mathbf{E} \in \mathbb{R}^{p\times p}$ where $\mathbf{E}_{ij} = \mathbf{E}_{ji} = 1$ for $\mathbf{i} \in \mathcal{N}_j$, and $\mathbf{E}_{ji} = 0$ if $\mathbf{i} \notin \mathcal{N}_j$. Each sensor acquires a set of zero-mean measurements $\{x_\tau(j)\}_{j=1}^{p}$ of a particular deterministic quantity

at discrete time instances $\tau = 0, 1, 2, \ldots, t$. Let $\mathbf{x}_\tau := [x_\tau(1), \ldots, x_\tau(p)]^T$ contain all the sensor data at a given time instant. Note that $\mathbf{x}_t$ is not stored somewhere and its given here for notational purposes. The ensemble covariance of the sensor data $\mathbf{x}_t$ is denoted by:

$$\mathbf{\Sigma}_x = \mathbb{E}[\mathbf{x}_\tau \mathbf{x}_\tau^T]. \tag{2.1}$$

The ensemble covariance in practice is not available and we rely on the sensor observations to approximate this as

$$\mathbf{\Sigma}_x \approx \frac{1}{t} \sum_{\tau=0}^{t} [\mathbf{x}_\tau \mathbf{x}_\tau^T]. \tag{2.2}$$

Assuming temporally white data and using the law of large numbers [29], it follows that the right hand side sample-average estimate in (2.2) will converge as $t \to \infty$ to the ensemble covariance $\mathbf{\Sigma}_x$.

Let $\mathbf{\Sigma}_x = \mathbf{U}_x \mathbf{\Lambda}_x \mathbf{U}_x^T$ denote the eigenvalue decomposition of the covariance matrix, where $\mathbf{U}_x \in \mathbb{R}^{p \times p}$ is the eigenvector matrix, which is the eigenspace, and the diagonal matrix $\mathbf{\Lambda}_x$ contains the corresponding eigenvalues. Using principal component analysis, the principal eigenspace $\mathbf{U}_{x,r}$ can be found by estimating the $r \leq p$ principal eigenvectors of the covariance matrix $\mathbf{\Sigma}_x$ using the gathered sensor data. These estimates can be used in data compression, dimensionality reduction, image processing, denoising and so on [30–32]. The principal components constituted by the eigenspace $\mathbf{U}_{x,r}$ can be found, see e.g., [5], by minimizing

$$\hat{\mathbf{C}} = \arg\min (t+1)^{-1} \sum_{\tau=0}^{t} \|\mathbf{x}_\tau - \mathbf{C}^T \mathbf{C} \mathbf{x}_\tau\|_2^2, \tag{2.3}$$

with respect to $\mathbf{C} \in \mathbb{R}^{r \times p}$. The following equation follows from setting $\mathbf{y}_\tau = \mathbf{C}\mathbf{x}_\tau$ in (2.3)

$$(\hat{\mathbf{C}}, \{\hat{\mathbf{y}}_\tau\}_{\tau=0}^t) = \arg \min_{\mathbf{C}, \mathbf{y}_\tau} (t+1)^{-1} \sum_{\tau=0}^{t} \|\mathbf{x}_\tau - \mathbf{C}^T \mathbf{y}_\tau|_2^2, \tag{2.4}$$

The minimization framework given in (2.4) pertains to a centralized PCA formulation that requires all data to be available at a central location (fusion center) in order to be able to estimate the principal eigenspace of $\mathbf{\Sigma}_x$.

By estimating the covariance using $\hat{\mathbf{\Sigma}}_{x,t} = (t+1)^{-1}\sum_{\tau=0}^{t}\mathbf{x}_\tau\mathbf{x}_\tau^T$ and obtaining the r principal eigenvectors for which $\hat{\mathbf{C}} = \hat{\mathbf{U}}_{x,r}$, we form a minimizer for (2.3) which forms a separable PCA cost function that is amenable to distributed minimization. By setting $\mathbf{C}^T(\hat{\mathbf{C}}\hat{\mathbf{C}}^T)^{-1}\hat{\mathbf{C}} = \hat{\mathbf{U}}_{x,r}\hat{\mathbf{U}}_{x,r}^T$, both (2.4) and (2.3) reach the same minima even though (after applying first order optimality conditions) (2.3) yields $\hat{\mathbf{y}}_\tau = (\hat{\mathbf{C}}\hat{\mathbf{C}}^T)^{-1}\hat{\mathbf{C}}\mathbf{x}_\tau$ which is different from $\hat{\mathbf{y}}_\tau = \hat{\mathbf{C}}\mathbf{x}_\tau$. Now let $\hat{\mathbf{C}} = \mathbf{U}_c\mathbf{S}_c\mathbf{V}_c^T$ denote the singular value decomposition of the minimizer in (2.4), where $\mathbf{U}_c \in \mathbb{R}^{r\times r}$ and $\mathbf{V}_c \in \mathbb{R}^{p\times p}$ contain the left and right singular vectors of $\hat{\mathbf{C}}$, while diagonal matrix $\mathbf{S}_c$ contains the singular values. The last two equations indicate that r left singular vectors of $\hat{\mathbf{C}}$ corresponding to the largest singular values, say $\mathbf{U}_{c,r}$, satisfy $\mathbf{U}_{c,r} = \hat{\mathbf{U}}_{x,r}\mathbf{W}$ where $\mathbf{W}$ is an arbitrary $r \times r$ unitary matrix. Thus, $\mathbf{U}_{x,r}$ can be estimated from $\hat{\mathbf{C}}$ in (2.4) up to a unitary matrix ambiguity. A distributed PCA algorithm is derived next for a wireless sensor network equipped with bridge sensors [33].

2.2 Distributed Implementation

In this section, the centralized PCA cost in (2.4) is re-written in a separable way, after which the alternating method of multipliers (ADMM) [34] combined with block coordinate descent iteration [34, 35] yields a distributed estimation algorithm. The method of multipliers actually exploits the decomposable structure of the associated Lagrangian function [36]. Hence, (2.4) is easily decomposed into smaller subtasks that can be carried out in parallel across all sensors. Our aim is to form a cost function in which each sensor j forms updates for $\mathbf{C}_{:j}$, jth column of $\mathbf{C}$ for $j = 1, \ldots, p$. Starting from the cost function in (2.4) we have

$$J(\mathbf{C}, \{\mathbf{y}_\tau\}_{\tau=0}^t) = (t+1)^{-1}\sum_{j=1}^{p}\sum_{\tau=0}^{t}(x_\tau(j) - \mathbf{C}_{:j}^T\mathbf{y}_\tau)^2. \tag{2.5}$$

To form a separable PCA formulation, we introduce auxiliary vectors $\mathbf{y}_{\tau,j}$ for each sensor j, responsible for estimating the principal components in $\mathbf{y}_\tau$, subject to the appropriately

set consensus equality constraints $\mathbf{y}_{\tau,j} = \mathbf{y}_{\tau,b}$ that will ensure that the local estimates $\mathbf{y}_{\tau,j}$ are the same across all sensors. The following *separable* constrained optimization problem is then formed

$$(\hat{\mathbf{C}}, \{\check{\mathbf{y}}_{\tau,j}\}_{j=1,\tau=0}^{p,t}) = \arg\min (t+1)^{-1} \sum_{j=1}^{p} \sum_{\tau=0}^{t} (x_\tau(j) - \mathbf{C}_{:j}^T \mathbf{y}_{\tau,j})^2,$$

$$\text{s. to } \mathbf{y}_{\tau,j} = \mathbf{y}_{\tau,b}, \quad b \in \mathcal{B}_j \text{ and } \tau = 0, \ldots, t \tag{2.6}$$

where $\mathcal{B}_j := \mathcal{N}_j \cap \mathcal{B}$ represent a set of 'bridge' neighbors of the jth sensor and its cardinality represented by $|\mathcal{B}_j|$ for $j = 1, \ldots, p$. The set $\mathcal{B} \subseteq [1, p]$ is a subset of bridge sensors maintaining local vectors $\mathbf{y}_{\tau,b}$ that are utilized to impose consensus among all local variables $\mathbf{y}_{\tau,j}$ across all sensors. If, for example, $\mathcal{B} \equiv [1, p]$, then the communication graph is connected, i.e. there exists a path connecting any two sensors, and the constraints $\mathbf{y}_{\tau,j} = \mathbf{y}_{\tau,b}$, $\mathbf{y}_{\tau,j} = \mathbf{y}_{\tau,b}$, $b \in \mathcal{B}_j$ will render $\mathbf{y}_{\tau,j} = \mathbf{y}_{\tau,i} \forall\ i, j \in \{1, \ldots, p\}$. This further implies that the PCA cost in (2.4) is equivalent to the constrained separable formulation given in (2.6).

The set of bridge sensors $\mathcal{B}$ is selected such that, see e.g., [33]:

(a1) $\forall j \in [1, p]$ there exists at least one $b \in \mathcal{B}_j$ so that $b \in \mathcal{N}_j$; and

(b1) If j_1 and j_2 are single-hop neighboring sensors, there must exist a bridge sensor b such that $b \in \mathcal{N}_{j,1} \cap \mathcal{N}_{j,2}$.

Note that (a1) ensures that every node has a bridge sensor neighbor. Further, (b1), guarantees that that all bridge variables $\{\mathbf{y}_{\tau,b}\}_{b\in\mathcal{B}}$, as well as the local principal component estimates $\{\mathbf{y}_{\tau,j}\}_{\tau=0}^{t}$ can reach consensus i.e. become equal.

2.2.1 Alternating-Direction Method of Multipliers (ADMM)

To solve (2.6) in a distributed fashion we will exploit the decomposable structure of the so called augmented Lagrangian function (see e.g., [34]) and update the local variables

$\hat{\mathbf{y}}_{\tau,j}$, $\mathbf{y}_{\tau,b}$ and $\mathbf{C}_{:j}$ by combining the method of multipliers with block coordinate descent iterations. This approach will yield a distributed adaptive estimation algorithm.

Let us define the $\{\mathbf{v}_{\tau,j}^b\}_{b\in\mathcal{B}_j}$ that correspond to the Lagrange multipliers associated with the constraints $\mathbf{y}_{\tau,j} = \mathbf{y}_{\tau,b}$ for $b \in \mathcal{B}_j$ and updated at sensor j. Then, the augmented Lagrangian function corresponding to (2.6) can therefore be written as

$$\mathcal{L}[\mathbf{C}, \{\mathbf{y}_{\tau,j}\}_{\tau=0,j=1}^{t,p}, \{\mathbf{y}_{\tau,b}\}_{\tau=0,b\in\mathcal{B}}^{t}, \mathbf{v}] = (t+1)^{-1}\sum_{j=1}^{p}\sum_{\tau=0}^{t}(x_\tau(j) - \mathbf{C}_{:j}^T\mathbf{y}_{\tau,j})^2$$

$$+\sum_{\tau=0}^{t}\sum_{j=1}^{p}\sum_{b\in\mathcal{B}_j}\left[(\mathbf{v}_{\tau,j}^b)^T(\mathbf{y}_{\tau,j} - \mathbf{y}_{\tau,b})\right] + 0.5c\sum_{\tau=0}^{t}\sum_{j=1}^{p}\sum_{b\in\mathcal{B}_j}\left[\|\mathbf{y}_{\tau,j} - \mathbf{y}_{\tau,b}\|_2^2\right], \qquad (2.7)$$

where $c > 0$ are penalty coefficients that introduce the constraints-related second-order terms to make the PCA formulation in (2.6) strictly convex without affecting its optimal solution. The vector $\mathbf{v}$ in the left hand side of (2.7) contains the multipliers $\mathbf{v}_{\tau,j}^b$ for $\tau = 0, \ldots, t$, $b \in \mathcal{B}_j$ and $j = 1, \ldots, p$.

2.2.2 Batch Distributed PCA Algorithm

Let $\kappa = 0, 1, \ldots,$ denote the index for a coordinate descent cycle and $k = 1, \ldots, K$ indicate the consensus iteration index within a coordinate cycle. Then $\kappa \cdot K + k$ enumerates the total number of ADMM (consensus) iterations from the beginning and after k consensus iterations have been completed during the κth coordinate descent cycle. One coordinate descent cycle will entail one iteration per sensor to update $\mathbf{C}_{:j}$'s, and K consensus iterations associated with ADMM to respect the equality constraints. Hence, we let $\mathbf{y}_{\tau,j}(\kappa K + K)$ indicate the most recent updates for $\mathbf{y}_{\tau,j}$, after K consensus iterations have been completed during coordinate cycle κ. We now tackle (2.6) first by using the block coordinate descent method. First, (2.6) is minimized with respect to C after fixing $\mathbf{y}_{\tau,j}$ to their most up-to-date values $\mathbf{y}_{\tau,j}(\kappa K + K)$. Due to the separability of (2.6) with respect to the columns of $\mathbf{C}$, it turns out that sensor j has to tackle during coordinate descent cycle $\kappa + 1$ the minimization

task

$$\mathbf{C}_{:j}(\kappa+1) = \arg\min_{\mathbf{C}_{:j}} \left[\sum_{\tau=0}^{t}(x_\tau(j) - \mathbf{C}_{:j}{}^T\mathbf{y}_{\tau,j}((\kappa+1)K))^2\right], \tag{2.8}$$

via which sensor j can update the jth column of matrix $\mathbf{C}$. After calculating the first-order derivative of the cost in (2.8) and setting it equal to zero (first-order optimality conditions) it turns out that:

$$-2\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K)x_\tau(j) + 2\left[\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K))(\mathbf{y}_{\tau,j}((\kappa+1)K)))^T\right]\mathbf{C}_{:j}(k+1) = 0,$$

from which it follows that

$$\mathbf{C}_{:j}(\kappa+1) = \left[\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K)(\mathbf{y}_{\tau,j}((\kappa+1)K))^T\right]^{-1} \times \sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K)x_\tau(j). \tag{2.9}$$

For simplicity in exposition, let $\mathbf{y}_{\tau,j}((\kappa+1)K) := \mathbf{y}_{\tau,j}(k)$ in the notation used from now on. Note than in (2.9), a single iteration will be enough since $\hat{\mathbf{C}}$ will be evaluated in closed form. Next, the local variables $\mathbf{y}$'s and $\mathbf{v}$'s multipliers are updated across sensors by applying ADMM. The first step in ADMM involves updating the Lagrange multipliers using gradient ascent at coordinate iteration $\kappa+1$ as follows

$$\mathbf{v}_{\tau,j}^{\kappa+1,b}(k) = \mathbf{v}_{\tau,j}^{\kappa+1,b}(k-1) + c[\mathbf{y}_{\tau,j}^{\kappa+1}(k) - \mathbf{y}_{\tau,b}^{\kappa+1}(k)] \tag{2.10}$$

for $b \in \mathcal{B}_j$, $k = 1,\ldots,K$, $\kappa = -1, 0, \ldots$, and $\tau = 0,\ldots,t$. Here, $\mathbf{v}_{\tau,j}^{\kappa+1,b}(0) = \mathbf{v}_{\tau,j}^{\kappa,b}(K)$ which is the multiplier update after K consensus iterations, so coordinate cycle $\kappa+1$ starts using the most up-to-date $\mathbf{v}$'s and $\mathbf{y}$'s from cycle κ.

Next, we form updates for $\mathbf{y}_{\tau,j}$ by minimizing (2.7) with respect to $\mathbf{y}_{\tau,j}$ after fixing $\mathbf{C}$ and the multipliers $\mathbf{v}$'s to their most-up-date values. Multiple iterations K will however be needed for the local variables $\mathbf{y}_{\tau,j}$ to reach consensus.

$$\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) = \arg\min_{\mathbf{y}_{\tau,j}} \left[\mathcal{L}(\mathbf{y}_j, \mathbf{y}_b(k), \mathbf{v}(k))\right] \tag{2.11}$$

$$= \arg\min_{\mathbf{y}_{\tau,j}} [\sum_{\tau=0}^{t}\sum_{j=1}^{p}(x_\tau(j) - \mathbf{C}_{:j}^T(k+1)\mathbf{y}_{\tau,j})^2$$

$$+ \sum_{b\in\mathcal{B}_j}\sum_{\tau=0}^{t}\left[(\mathbf{v}_{\tau,j}^b(\kappa))^T(\mathbf{y}_{\tau,j} - \mathbf{y}_{\tau,b}(\kappa))\right] + 0.5c\sum_{b\in\mathcal{B}_j}\sum_{\tau=0}^{t}\left[\|\mathbf{y}_{\tau,j} - \mathbf{y}_{\tau,b}(\kappa)\|_2^2\right]. \tag{2.12}$$

Applying the first-order optimality conditions to find the optimal solution of the strictly convex problem in (2.11) it turns out that

$$-2(\mathbf{C}_{:j}(\kappa+1))\left[x_\tau(j) - (\mathbf{C}_{:j}(\kappa+1))^T\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)\right]$$

$$+ \sum_{b\in\mathcal{B}_j}\sum_{\tau=0}^{t}(\mathbf{v}_{\tau,j}^{\kappa+1,b}(\kappa)) + 0.5c\sum_{b\in\mathcal{B}_j}\sum_{\tau=0}^{t}\left[2\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) - 2\mathbf{y}_{\tau,b}^{\kappa+1}(k+1)\right] = 0. \tag{2.13}$$

Solving with respect to $\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$ it follows that

$$\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) = \left[2\mathbf{C}_{:j}(\kappa+1)(\mathbf{C}_{:j}(\kappa+1))^T + c|\mathcal{B}_j|\mathbf{I}\right]^{-1}$$

$$\times \left[2\mathbf{C}_{:j}(\kappa+1)x_\tau(j) - \textstyle\sum_{b\in\mathcal{B}_j}\mathbf{v}_{\tau,j}^{\kappa+1,b}(k) + c\sum_{b\in\mathcal{B}_j}\mathbf{y}_{\tau,b}^{\kappa+1}(k)\right]. \tag{2.14}$$

where $\kappa = 0, 1, \ldots,$, $k = 1, \ldots, K$ and $\tau = 0, \ldots, t$ for $b \in \mathcal{B}$.

The final step in ADMM involves updating $\mathbf{y}_{\tau,b}$ by minimizing (2.6) with respect to $\mathbf{y}_{\tau,b}$ after fixing $\mathbf{C}$ and $\mathbf{y}_{\tau,j}$ to their most up-to-date values. The associated cost that has to be minimized at bridge sensor b can be obtained from (2.7) after isolating the terms involving $\mathbf{y}_{\tau,b}$, i.e.,

$$\mathbf{y}_{\tau,b}^{\kappa+1}(k+1) = \arg\min_{\mathbf{y}_{\tau,b}} \left[\mathcal{L}(\mathbf{y}_j(k+1), \mathbf{y}_b, \mathbf{v}(k))\right] \tag{2.15}$$

$$= \arg\min_{\mathbf{y}_{\tau,b}} [\sum_{j=1}^{p}\sum_{\tau=0}^{t}(x_\tau(j) - \mathbf{C}_{:j}^T(k+1)\mathbf{y}_{\tau,j})^2$$

$$+\sum_{j\in\mathcal{N}_b}\sum_{\tau=0}^{t}\left[(\mathbf{v}_{\tau,j}^{b}(k))^T(\mathbf{y}_{\tau,j}(k+1)-\mathbf{y}_{\tau,b})\right]+0.5c\sum_{j\in\mathcal{N}_b}\sum_{\tau=0}^{t}\left[\|\mathbf{y}_{\tau,j}(k+1)-\mathbf{y}_{\tau,b}\|_2^2\right], \tag{2.16}$$

After applying first-order optimality conditions in the convex cost of (2.15) it follows readily that

$$\mathbf{y}_{\tau,b}^{\kappa+1}(k+1)=\sum_{j\in\mathcal{N}_b}\frac{1}{(\sum_{\beta\in\mathcal{N}_b}c_\beta)}\left[\mathbf{v}_{\tau,j}^{\kappa+1,b}(k)+c\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)\right] \tag{2.17}$$

where $\kappa=0,1,\ldots,$, $k=1,\ldots,K$ and $\tau=0,\ldots,t$ and $b\in\mathcal{B}$.

Let the Lagrange multipliers $\{\mathbf{v}_{\tau,j}^{b}(-1)\}_{b\in\mathcal{B}_j}$, local estimates $\{\mathbf{y}_{\tau,j}(0)\}_{j=1}^{p}$ and consensus variables $\{\mathbf{y}_{\tau,b}(0)\}_{b\in\mathcal{B}_j}$ be arbitrary [33] with the WSN reaching consensus as $k\to\infty$. It turns out that if an infinite number of consensus iterations ($K\to\infty$) is applied during coordinate cycle $\kappa+1$ then consensus is reached such that $\lim_{k\to\infty}\mathbf{y}_{\tau,j}^{\kappa+1}(k)=(\mathbf{C}(\kappa+1)\mathbf{C}(\kappa+1)^T)^{-1}\mathbf{C}(\kappa+1)\mathbf{x}_\tau$ which constitute the principal component vectors; for all sensors $j=1,\ldots,p$, while $\mathbf{C}(\kappa+1):=[\mathbf{C}_{:1}(\kappa+1)\ldots\mathbf{C}_{:p}(\kappa+1)]$. The latter convergence claim can be established using similar arguments as the ones given in [33].

Equations (2.10), (2.14) and (2.17) constitute the recursions that are executed for K consensus iterations during each cycle κ. Together with (2.9), these form our *batch* D-PCA approach. The proposed algorithm is tabulated below as Algorithm 1.

All sensors $j\in[1,p]$ keep track of:

i) $\mathbf{C}_{:j}(k+1)\in\mathbb{R}^{r\times 1}$, that estimates the jth row of $\mathbf{U}_{x,r}$;

ii) the multipliers $\{\mathbf{v}_{\tau,j}^{\kappa,j'}(k)\}_{j'\in\mathcal{N}_j}$; and

iii) the principal components in $\mathbf{y}_{\tau,j}^{\kappa}(k+1)$ for $\tau=0,\ldots,t$.

All sensors belonging to $\mathcal{B}$ also keep track of $\mathbf{y}_{\tau,b}^{\kappa}(k+1)$ for $\tau=0,\ldots,t$.

Algorithm 1 Batch Distributed Principal Component Analysis

1: Lagrange multipliers $\{\mathbf{v}_{\tau,j}^{\kappa,b}(-1)\}_{b\in\mathcal{B}_j}$,sensor local estimates $\mathbf{y}_{\tau,j}^{\kappa+1}(0)$, consensus enforcing variables $\mathbf{y}_{\tau,b}^{\kappa+1}(0)$ are randomly initialized

2: Every sensor j gathers $t+1$ measurements $\{x_j(\tau)\}_{\tau=0}^{t}$.

3: **for** $\kappa = 1, 2, \ldots$ **do**

4: Each sensor j the jth row of the principal component matrix $\mathbf{C}$, estimating $\mathbf{U}_{x,r}^T$, and forms $\mathbf{C}_{:j}(\kappa+1) \in \mathbb{R}^{r\times 1}$ via (2.9).

5: **for** $k = 1, 2, \ldots, K$ **do**

6: Sensor j updating the multipliers $\{\mathbf{v}_{\tau,j}^{\kappa,b}(k)\}_{b\in\mathcal{B}_j}$ using (2.10).

7: Bridge sensor b belonging to $\mathcal{B}$ updating the consensus-enforcing variables $\mathbf{y}_{\tau,b}^{\kappa+1}(k+1)$ using (2.17).

8: Sensor j receiving the consensus variables from all its neighbors in the subset $\mathcal{B}$ ($b \in \mathcal{B}_j$) and estimates $\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$ using (2.14).

9: **end for**

10: If $\max_{j=1,\ldots,p}(\|\hat{\mathbf{C}}_{:j}^{\kappa+1} - \hat{\mathbf{C}}_{:j}^{\kappa}\|_2) \leq \epsilon$ then stop (for desired tolerance ϵ).

11: **end for**

In the batch process, sensors first gather $t+1$ measurements (for fixed t) i.e. $x_0, ..., x_t$. The D-PCA algorithm then takes effect. During coordinate descent cycle $\kappa+1$, sensor j forms $\mathbf{C}_{:j}(\kappa+1) \in \mathbb{R}^{r\times 1}$ via (2.9), using local principal components $\mathbf{y}_{\tau,j}^{\kappa}(K)$ for $\tau = 0, ..., t$. Each sensor now runs K consensus iterations comprising the following steps:

i) At consensus iteration $k+1$ still during coordinate descent cycle $\kappa+1$,each sensor j receives consensus variables $\mathbf{y}_{\tau,b}^{\kappa+1}(k)$ from all its neighbors in the subset $\mathcal{B}$, namely all $b \in \mathcal{B}_j$. Based on these consensus variables, it updates the Lagrange multipliers $\{\mathbf{v}_{\tau,j}^{\kappa+1,b}(k)\}_{b\in\mathcal{B}_j}$ using (2.10).

ii)Sensor j then combines $\{\mathbf{v}_{\tau,j}^{\kappa+1,b}(k)\}_{b\in\mathcal{B}_j}$ which is used along with $\{\mathbf{y}_{\tau,b}^{\kappa+1}(k)\}_{b\in\mathcal{B}_j}$ to form $\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$ via (2.14). After determining $\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$, sensor j transmits to each of its

neighbors $b \in \mathcal{B}_j$ the vector $\mathbf{v}_{\tau,j}^{\kappa+1,b}(k) + c\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$.

iii) Each sensor $b \in \mathcal{B}_j$ receives $\mathbf{v}_{\tau,j}^{\kappa+1,b}(k) + c\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$ from all its neighbors $j \in \mathcal{N}_b$ which then initializes the $(k+1)$th consensus iteration.

For an increasing number of consensus iterations $K \to \infty$ and coordinate descent cycles ($\kappa \to \infty$), $\mathbf{C}(\kappa)$ and $\mathbf{y}_{\tau,j}(\kappa)$ converge at least to a stationary point of the PCA cost in (2.5). This can be established using the convergence properties of block coordinate descent and ADMM in [33, 34, 36].

2.2.3 Adaptive D-PCA

In situations where real time processing of data is of utmost importance as compared to estimation performance, a batch implementation will not be suitable. This particularly comes to play in cases where the number of measurements $t+1$ is not fixed. This is because every sensor has to carry out $(t+1)K$ consensus iterations for updating $\{\mathbf{y}_{\tau,j}^{\kappa}\}_{\tau=0}^{t}$. For an increasingly large t, i.e. with sensors collecting a large amount of data, a batch implementation will have a large computational complexity, as well as high communication and memory costs. Building on batch D-PCA we derive an adaptive D-PCA scheme that is capable to process data online, while having a manageable computational, communication and memory cost [37]. In batch D-PCA the separable PCA cost in (2.6) is time-invariant. For a constant t, the batch D-PCA consisted of multiple coordinate descent cycles that could be run until, e.g., the the updates for matrix $\mathbf{C}$ (or PCA cost) do not decrease below a desired limit.

In an adaptive setting sensors acquire new data at every time instant t. Thus, for an increasing t the cost (2.6) will be augmented with new data terms at every time instant. Therefore, it becomes important at t to apply a small number of coordinate cycles; here one coordinate cycle per t is employed to update the $\mathbf{C}$ matrix. Thus, the time t and coordinate cycle

indices κ coincide. It should be noted that K consensus iterations are still performed using (2.10), (2.14) and (2.17). Specifically, at time t the following steps are taking place

$$\mathbf{C}_{:j}(t+1) = \left[\sum_{\tau=0}^{t} \mathbf{y}_{\tau,j}(K)(\mathbf{y}_{\tau,j}(K))^T\right]^{-1} \times \sum_{\tau=0}^{t} \mathbf{y}_{\tau,j}(K)x_\tau(j), \tag{2.18}$$

$$\begin{aligned}\mathbf{y}_{t+1,j}(k+1) = &\left[2\mathbf{C}_{:j}(t+1)(\mathbf{C}_{:j}(t+1))^T + 2c|\mathcal{B}_j|\mathbf{I}\right]^{-1} \\ &\times \left[2\mathbf{C}_{:j}(k+1)x_{t+1}(j) - \sum_{b\in\mathcal{B}_j} \mathbf{v}_{t+1,j}^b(k) + c\sum_{b\in\mathcal{B}_j} \mathbf{y}_{t+1,b}(k)\right],\end{aligned} \tag{2.19}$$

$$\mathbf{y}_{t+1,b}(k+1) = \sum_{j\in\mathcal{N}_b} \frac{1}{\left(\sum_{\beta\in\mathcal{N}_b} c_\beta\right)} \left[\mathbf{v}_{t+1,j}^b(k) + c\mathbf{y}_{t+1,j}(k+1)\right], \tag{2.20}$$

$$\mathbf{v}_{t+1,j}^b(k) = \mathbf{v}_{t+1,j}^b(k-1) + 0.5c[\mathbf{y}_{t+1,j}(k) - \mathbf{y}_{t+1,b}(k)], k = 1,\ldots,K \tag{2.21}$$

which is a single recursion that updates C, and then updates $\mathbf{y}_{t,j}$, $\mathbf{y}_{t,b}$, ${\mathbf{v}_{t,j}}^b$ over K consensus iterations.(where $k = 0, 1, \ldots, K-1$ in (2.19)). Note that coordinate cycle superscripts 'κ' have been removed since one cycle takes place per t. Note that in contrast to the batch approach, in the adaptive implementation ADMM (consensus iterations) is applied only for the $\mathbf{y}_{t,j}$ and $\mathbf{y}_{t,b}$ that contain the most recent data and not for all $\{\mathbf{y}_{\tau,j}, \mathbf{y}_{\tau,b}\}_{\tau=0}^{t}$. The multipliers at $t = 0$, namely $\mathbf{v}_{j,0}(0)$, are initialized randomly, whereas at time instant $t > 0$ warm-starts are employed to set $\mathbf{v}_{t,j}(0) = \mathbf{v}_{t-1,j}(K)$. At time instant t variables $\mathbf{y}_{t,j}$ are initialized as $\mathbf{y}_{t,j}(0) = \mathbf{C}_{:j}(t)x_t(j)$.

As stated earlier, what results a smaller computational complexity compared to the batch algorithm, is that at time t the ADMM updating recursions (2.10), (2.14) and (2.17) will only be performed the multipliers $\mathbf{v}_{t,j}$, the principal vectors $\mathbf{y}_{t,j}$ and the consensus enforcing variables $\mathbf{y}_{t,b}$ that contain the most recent data at time instant t. On the contrary, in the batch algorithm the ADMM is applied for all $\{\mathbf{v}_{\tau,j}, \mathbf{y}_{\tau,b}, \mathbf{y}_{\tau,j}\}$ for $\tau = 0, \ldots, t$.

Let us define now the following quantities

$$\mathbf{M}_{x,t} := \sum_{\tau=0}^{t} \mathbf{y}_{\tau,j}(K)(\mathbf{y}_{\tau,j}(K))^T \tag{2.22}$$

and

$$\mathbf{m}_{xy,t} := \sum_{\tau=0}^{t} \mathbf{y}_{\tau,j}(K) x_\tau(j), \tag{2.23}$$

which are necessary in implementing updating $\mathbf{C}_{:j}(t+1)$ using (2.18).

At time $\tau = t+1$, $\mathbf{M}_{x,t}$ and $\mathbf{m}_{xy,t}$ are updated in an adaptive fashion as

$$\mathbf{M}_{x,t+1} = \mathbf{M}_{x,t} + \mathbf{y}_{t+1,j}(K)(\mathbf{y}_{t+1,j}(K))^T \tag{2.24}$$

and

$$\mathbf{m}_{xy,t+1} = \mathbf{m}_{xy,t} + \mathbf{y}_{t+1,j}(K) x_t(j) \tag{2.25}$$

such that the $\mathbf{M}_{x,\tau}$ and $\mathbf{m}_{xy,\tau}$ for $\tau = 0, \ldots, t$ remains constant. Results from (2.24) and (2.25) are used to update $\mathbf{C}_{:j}(t+1)$ using (2.18). The ADMM process kicks in by initializing as

$$\mathbf{y}_{t+1,j}(0) = \mathbf{C}_{:j}(t+1) x_{t+1}(j) \tag{2.26}$$

$$\mathbf{y}_{t+1,b}(0) = \mathbf{y}_{t,b}(K) \tag{2.27}$$

$$\mathbf{v}_{t+1,j}(0) = \mathbf{v}_{t,j}(K) \tag{2.28}$$

During the kth consensus iteration, each sensor j receives updates $\mathbf{y}_{t+1,b}$ from its bridge neighbors $b \in \mathcal{B}_j$ and the multipliers $\mathbf{v}^b_{t+1,j}$ are updated using (2.21). Sensor j then updates $\mathbf{y}_{t+1,j}$ at time $k+1$ for $k \in K$, using (2.19) with $\mathbf{v}^b_{t+1,j}$ and $\mathbf{y}_{t+1,j}$ obtained from the bridge neighbors $b \in \mathcal{B}_j$ in the previous iteration. Updates $\mathbf{y}_{t+1,b}$ are then formed at the bridge sensors $b \in \mathcal{B}_j$ using the $\mathbf{y}_{t+1,j}$ update and the multiplier update $\mathbf{v}^b_{t+1,j}$. This process is repeated for K ADMM iterations.

Further, during time instant $\tau = t+2$, $\mathbf{M}_{x,t+1}$ and $\mathbf{m}_{xy,t+1}$ are updated as

$$\mathbf{M}_{x,t+2} = \mathbf{M}_{x,t+1} + \mathbf{y}_{t+2,j}(K)(\mathbf{y}_{t+2,j}(K))^T \tag{2.29}$$

and

$$\mathbf{m}_{xy,t+2} = \mathbf{m}_{xy,t+1} + \mathbf{y}_{t+2,j}(K)x_t(j) \tag{2.30}$$

Results from (2.29) and (2.30) are used to update $\mathbf{C}_{:j}(t+2)$ using (2.18). ADMM is applied for K iterations, and the whole process continues for $\tau = t+3, t+4, \ldots$. Notice that for all $\tau = 0, \ldots, t$, updates $\mathbf{v}^b_{\tau,j}(K)$,$\mathbf{y}_{t,j}(K)$ and $\mathbf{y}_{t,b}(K)$ remain constant for the time instances $\tau+1, \tau+2, \ldots$ for $\tau < t$ since consensus is only applied for $\mathbf{y}_{t,j}$, $\mathbf{y}_{t,b}$ and $\mathbf{v}^b_{t,j}$.

2.3 Communication cost

For the communication costs associated with a D-PCA network setting with bridge sensors, *simple* sensor j in D-PCA has to transmit $r(|\mathcal{B}_j|)$ scalars per consensus iteration corresponding to the entries of the multipliers $\mathbf{v}^b_{t,j}(k)$ and the local estimate $\mathbf{y}_{t,j}(k)$. Given that at each time instant t there are K consensus iterations taking place, the total transmission load per sensor during time instant t is $rK(|\mathcal{B}_j|)$. For the bridge sensors, a bridge sensor b has to transmit $r(|\mathcal{N}_j|+1)$ scalars per consensus iteration corresponding to the entries of the consensus enforcing variables $\mathbf{y}_{t,b}(k)$. Similarly, for K consensus iterations,the total transmission load per sensor during time instant t is $rK(|\mathcal{N}_j|+1)$. Generally, since $|\mathcal{B}_j| < |\mathcal{N}_j|$,then $rK|\mathcal{B}_j| < rK(|\mathcal{N}_j|+1)$ and hence a bridge sensor has a higher transmission cost than a simple sensor j. In S-PCA each sensor has to transmit $rK(r+1)$ to carry out consensus-iterations involving $r \times r$ matrices and $r \times 1$ vectors. Comparing with the bridge sensors, if $r < (|\mathcal{B}_j| - 1)$ is smaller than the size of the single-hop neighborhoods then S-PCA has a smaller transmission cost, whereas if $r > (|\mathcal{B}_j| - 1)$ D-PCA will have an advantage. When compared with the simple sensors, if $r < |\mathcal{N}_j|$ is smaller than the size of the single-hop neighborhoods then S-PCA has a smaller transmission cost, whereas if $r > |\mathcal{N}_j|$ D-PCA will have an advantage.

When considering the reception cost it can be seen that in D-PCA each *simple* sensor receives $2rK|\mathcal{B}_j|$ scalars during K consensus iterations pertaining to $\{\mathbf{y}_{t,b}(k)\}_{b\in\mathcal{B}_j}$. Considering the bridge sensors, each bridge sensor receives $2rK(|\mathcal{N}_j|)$ scalars per consensus iteration corresponding to the entries of the multipliers $\mathbf{v}_{t,j}^b(k)$ and the local estimate $\mathbf{y}_{t,j}(k)$. Again, since $|\mathcal{B}_j| < |\mathcal{N}_j|$, the bridge sensors have higher receiving cost than the *simple* sensors. However, in S-PCA sensor j receives $(r+1)rK|\mathcal{N}_j| \geq 2rK|\mathcal{N}_j| > 2rK|\mathcal{B}_j|$ due to the reception of $\mathcal{N}_j$ $r\times r$ matrices and $r\times 1$ vectors per consensus iteration. Thus, D-PCA has a smaller reception cost for $r > 1$. Even though the transmission cost of D-PCA may be greater than the one in S-PCA, it will be corroborated via numerical examples that the higher transmission cost in D-RLS pays off in improved convergence rates and steady-state performance.

The use of bridge sensors is ideal for networks in which a few sensors (the bridges) have higher computational and memory capabilities than the rest. However, in a situation where all sensors have equal capability, and want to avoid the extra coordination required for setting and updating the bridge sensor set an alternative separable formulation for PCA is set forth that does not require the utilization of bridge sensors.

CHAPTER 3

DISTRIBUTED PCA WITHOUT BRIDGE SENSORS

In the previous chapter to enable task parallelization of PCA via ADMM was achieved by relying on the so called bridge sensor subset. Not only setting up, but also readjusting the bridge sensor set, e.g., when sensors inevitably run out of battery resources, requires additional coordination among sensors. To allow a distributed implementation in sensor networks where all sensors are the same and therefore there are not special bridge sensors a different formulation is proposed here. As in Chpt. 2 the machinery utilized will be ADMM. However, a simpler algorithm that can be applied in sensor networks with no hierarchical structure. On the other hand the approach proposed in Chapter 2, has a smaller communication cost for the no-bridge sensors while it exhibits faster convergence rates compared to the framework put here. Thus, selecting between the bridge-based D-PCA and no-bridge D-PCA depends on the available resources, as well as the desired convergence speed.

3.1 Problem Statement

Next, we consider an ad hoc sensor network with J sensors in which each sensor j is only capable of single-hop communication. Here, a sensor can only communicate with its directly connected neighbour in $\mathcal{N}_j$ with cardinality $|\mathcal{N}_j|$. Assuming links in the sensor network are symmetric, we model the sensor network as an undirected connected graph whose vertices are sensors and edges represent communication links. We characterize this network by an adjacency matrix $\mathbf{E} \in \mathbb{R}^{J\times J}$ where $\mathbf{E}_{ij} = \mathbf{E}_{ji} = 1$ for $\mathbf{i} \in \mathcal{N}_j$ and $\mathbf{E}_{ji} = 0$ if $\mathbf{i} \notin \mathcal{N}_j$. Each sensor acquires a set of zero-mean measurements $\{x_\tau(j)\}_{j=1}^{J}$ of a particular

deterministic quantity at discrete time instances $\tau = 0, 1, 2, \ldots, t$. As in Chpt. 2 these measurements are stacked in $\mathbf{x}_\tau := [x_\tau(1), \ldots, x_\tau(J)]^T$, and the covariance becomes

$$\mathbf{\Sigma}_x = \mathbb{E}[\mathbf{x}_\tau \mathbf{x}_\tau^T]. \tag{3.1}$$

For a very large number of observations t, the covariance can be approximated as

$$\mathbf{\Sigma}_x \approx \frac{1}{t} \sum_{\tau=0}^{t} [\mathbf{x}_\tau \mathbf{x}_\tau^T], \tag{3.2}$$

where equality holds for an infinite number of observations.

Let $\mathbf{\Sigma}_x = \mathbf{U}_x \mathbf{\Lambda}_x \mathbf{U}_x^T$ denote the eigenvalue decomposition of the covariance matrix, where $\mathbf{U}_x \in \mathbb{R}^{J \times J}$ is the eigenvector matrix, which is the eigenspace, and the diagonal matrix $\mathbf{\Lambda}_x$ contains the corresponding eigenvalues. Using principal component analysis, the principal eigenspace $\mathbf{U}_{x,r}$ can be found by estimating the r principal eigenvectors of the covariance matrix $\mathbf{\Sigma}_x$ using the gathered sensor data. The principal components constituted by the eigenspace $\mathbf{U}_{x,r}$ can be found, see [5], by minimizing

$$\hat{\mathbf{C}} = \arg\min (t+1)^{-1} \sum_{\tau=0}^{t} \|\mathbf{x}_\tau - \mathbf{C}^T \mathbf{C} \mathbf{x}_\tau\|_2^2, \tag{3.3}$$

with respect to $\mathbf{C} \in \mathbb{R}^{r \times J}$. The following equation follows from setting $\mathbf{y}_\tau = \mathbf{C}\mathbf{x}_\tau$ in (3.3)

$$(\hat{\mathbf{C}}, \{\hat{\mathbf{y}}_\tau\}_{\tau=0}^t) = \arg \min_{\mathbf{C}, \mathbf{y}_\tau} (t+1)^{-1} \sum_{\tau=0}^{t} \|\mathbf{x}_\tau - \mathbf{C}^T \mathbf{y}_\tau|_2^2, \tag{3.4}$$

The above equation in the centralized PCA cost function. By estimating the covariance using $\hat{\mathbf{\Sigma}}_{x,t} = (t+1)^{-1} \sum_{\tau=0}^{t} \mathbf{x}_\tau \mathbf{x}_\tau^T$ and obtaining the r principal eigenvectors for which $\hat{\mathbf{C}} = \hat{\mathbf{U}}_{x,r}$, we form a minimizer for (3.3) which forms a separable PCA cost function that is amenable to distributed minimization.

Now let $\hat{\mathbf{C}} = \mathbf{U}_c \mathbf{S}_c \mathbf{V}_c^T$ denote the singular value decomposition of the minimizer in (3.4), where $\mathbf{U}_c \in \mathbb{R}^{r \times r}$ and $\mathbf{V}_c \in \mathbb{R}^{J \times J}$ contain the left and right singular vectors of $\hat{\mathbf{C}}$, while

diagonal matrix $\mathbf{S}_c$ contains the singular values. The last two equations indicate that r left singular vectors of $\check{\mathbf{C}}$ corresponding to the largest singular values, say $\mathbf{U}_{c,r}$, satisfy $\mathbf{U}_{c,r} = \hat{\mathbf{U}}_{x,r}\mathbf{W}$ where $\mathbf{W}$ is an arbitrary $r \times r$ unitary matrix. Thus, $\mathbf{U}_{x,r}$ can be estimated from $\check{\mathbf{C}}$ in (3.4) up to a unitary matrix ambiguity. A distributed PCA setting without bridge sensors is now considered in the next section.

3.2 Distributed Implementation of PCA

In order to develop a distributed (D-) PCA algorithm without bridge sensors, we will rewrite the centralized PCA cost in (3.4) in a separable way and then employ the alternating direction method of multipliers (ADMM) [33–35] combined with block coordinate descent techniques, see e.g., [36], to split the optimization problem into smaller subtasks that can be implemented in parallel across sensors. Towards this end, the cost in (3.4) can be rewritten as

$$J(\mathbf{C}, \{\mathbf{y}_\tau\}_{\tau=0}^t) = (t+1)^{-1} \sum_{j=1}^J \sum_{\tau=0}^t (x_\tau(j) - \mathbf{C}_{:j}^T \mathbf{y}_\tau)^2, \tag{3.5}$$

where $\mathbf{C}_{:j}$ denotes the jth column of $\mathbf{C}$ for $j = 1, \ldots, J$. Sensor j is responsible for forming updates for $\mathbf{C}_{:j}$ that estimates the jth row of $\mathbf{U}_{x,r}$. Since summands in (3.5) are coupled through the vectors $\mathbf{y}_\tau$, separate minimization of $\sum_{\tau=0}^t (x_\tau(j) - \mathbf{C}_{:j}^T \mathbf{y}_\tau)^2$ at sensor j will not return the minimizer of (3.4). A separable PCA formulation that is equivalent to the centralized minimization problem in (3.4) is obtained by introducing the auxiliary vectors $\mathbf{y}_{\tau,j}$ for each sensor j and impose the consensus constraint $\mathbf{y}_{\tau,1} = \mathbf{y}_{\tau,2} = \ldots = \mathbf{y}_{\tau,J}$. The following *separable* constrained optimization problem is obtained

$$\begin{aligned} (\check{\mathbf{C}}, \{\check{\mathbf{y}}_{\tau,j}\}_{j=1,\tau=0}^{J,t}) = \arg\min (t+1)^{-1} \sum_{j=1}^J \sum_{\tau=0}^t (x_\tau(j) - \mathbf{C}_{:j}^T \mathbf{y}_{\tau,j})^2, \\ \text{s. to } \mathbf{y}_{\tau,j} = \mathbf{y}_{\tau,j'}, \quad j' \in \mathcal{N}_j \text{ and } \tau = 0, \ldots, t \end{aligned} \tag{3.6}$$

Since the network is connected it follows readily that (3.6), (3.5) and (3.4) are equivalent in the sense that $\{\check{\mathbf{y}}_{\tau,j} = \check{\mathbf{y}}_\tau\}_{j=1}^J$.

3.2.1 Application of the ADMM

In order to solve (3.6) in a distributed implementation, we employ ADMM which will allow sensor j to obtain iteratively estimates for the jth row of $\mathbf{U}_x$ via $\mathbf{C}_{:j}$. To facilitate utilization of ADMM, consider the auxiliary variables $\mathbf{z}_{\tau,j}^{j'}$ for $j' \in \mathcal{N}_j$ and $\tau = 0, \ldots, t$. Then, substitute the constraints in (3.6) with the equivalent ones

$$\mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j}^{j'} \text{ and } \mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j'}^{j} \text{ for } j' \in \mathcal{N}_j \text{ and } j \neq j'. \tag{3.7}$$

The variables $\mathbf{z}_{\tau,j}^{j'}$ are just used to derive the local recursions run across sensors to find $\check{\mathbf{C}}_{:j}$, and eventually these variables are eliminated. Next, let $\mathbf{v}_{\tau,j}^{j'}$ and $\mathbf{w}_{\tau,j}^{j'}$ denote the multipliers associated with the constraints $\mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j}^{j'}$ and $\mathbf{y}_{\tau,j} = \mathbf{z}_{\tau,j'}^{j}$ respectively. ADMM exploits the decomposable structure of the augmented Lagrangian function [36, Ch. 3] which for (3.6) is written as

$$\begin{aligned}\mathcal{L}[\mathbf{C}, \{\mathbf{y}_{\tau,j}\}_{\tau=0,j=1}^{t,J}, \mathbf{v}, \mathbf{w}] &= (t+1)^{-1} \sum_{j=1}^{J} \sum_{\tau=0}^{t} (x_\tau(j) - \mathbf{C}_{:j}^T \mathbf{y}_{\tau,j})^2 \\ &+ \sum_{j=1}^{J} \sum_{j' \in \mathcal{N}_j} \sum_{\tau=0}^{t} \left[(\mathbf{v}_{\tau,j}^{j'})^T (\mathbf{y}_{\tau,j} - \mathbf{z}_{\tau,j}^{j'}) + (\mathbf{w}_{\tau,j}^{j'})^T (\mathbf{y}_{\tau,j} - \mathbf{z}_{\tau,j'}^{j}) \right] \\ &+ 0.5c \sum_{j=1}^{J} \sum_{j' \in \mathcal{N}_j} \sum_{\tau=0}^{t} \left[\|\mathbf{y}_{\tau,j} - \mathbf{z}_{\tau,j}^{j'}\|_2^2 + \|\mathbf{y}_{\tau,j} - \mathbf{z}_{\tau,j'}^{j}\|_2^2 \right], \end{aligned} \tag{3.8}$$

where c is a positive penalty coefficient, while $\mathbf{v}$ and $\mathbf{w}$ contain the multipliers $\mathbf{v}_{\tau,j}^{j'}$ and $\mathbf{w}_{\tau,j}^{j'}$, respectively for $\tau = 0, \ldots, t$, $j' \in \mathcal{N}_j$ and $j = 1, \ldots, J$. In order to tackle (3.6) we first employ a block coordinate descent where we first minimize wrt $\mathbf{C}$ while treating the $\mathbf{y}_{\tau,j}$'s fixed and vice versa. ADMM will be utilized when minimizing (3.6) wrt to $\mathbf{y}_{\tau,j}$, while

respecting the consensus constraints in (3.7). When minimizing wrt $\mathbf{C}$ assuming fixed $\mathbf{y}_{\tau,j}$ one iteration will be enough since $\check{\mathbf{C}}$ will be found in closed form. However, application of ADMM to determine $\mathbf{y}_{\tau,j}$ for a fixed $\mathbf{C}$, while respecting the equality constraints in (3.6), will involve multiple iterations denoted as K. Multiple recursions (ideally $K \to \infty$) in ADMM will be needed to enforce the consensus requirement across the $\mathbf{y}_{\tau,j}$ variables [33].

3.2.1.1 Batch Algorithm

Now let $\kappa = 0, 1, \ldots,$ denote the index for a coordinate descent cycle and $k = 1, \ldots, K$ indicate the consensus iteration index within a coordinate cycle. Then $\kappa \cdot K + k$ enumerates the total number of consensus iterations from the beginning after k consensus iterations have been completed during the κth coordinate descent cycle. One coordinate descent cycle will entail one iteration per sensor to update $\mathbf{C}_{:j}$'s, and K consensus iterations associated with ADMM. Specifically, let $\mathbf{y}_{\tau,j}(\kappa K + K)$ and $\mathbf{z}^{j'}_{\tau,j}(\kappa K + K)$ indicate the most recent updates for $\mathbf{y}_{\tau,j}$ and $\mathbf{z}^{j'}_{\tau,j}$ respectively, after K consensus iterations have been completed during coordinate cycle κ. Minimization of (3.6) wrt to $\mathbf{C}_{:j}$ during coordinate descent cycle $\kappa + 1$, while treating $\mathbf{y}_{\tau,j}$ as fixed and equal to $\mathbf{y}_{\tau,j}(\kappa K + K)$. From the augmented Lagrangian function in (3.8) it follows that matrix $\mathbf{C}$ can be obtained as the minimizer of

$$(\check{\mathbf{C}}) = \arg\min (t+1)^{-1} \sum_{j=1}^{J} \sum_{\tau=0}^{t} (x_\tau(j) - \mathbf{C}_{:j}^T \mathbf{y}_{\tau,j}((\kappa+1)K))^2. \tag{3.9}$$

Exploiting the separability of (3.9) with respect to the sensor index j, it follows that the update for the jth column of $\mathbf{C}$ can be found at sensor j as

$$\check{\mathbf{C}}(\kappa+1) = \arg\min (t+1)^{-1} \sum_{\tau=0}^{t} (x_\tau(j) - \mathbf{C}_{:j}^T \mathbf{y}_{\tau,j}((\kappa+1)K).))^2 \tag{3.10}$$

Applying first-order optimality conditions in (3.10) it gives

$$-2\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K)x_\tau(j) \\ +2\left[\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K))(\mathbf{y}_{\tau,j}((\kappa+1)K)))^T\right]\mathbf{C}_{:j}(\kappa+1)=0, \tag{3.11}$$

from which it follows readily that

$$\mathbf{C}_{:j}(\kappa+1)=\left[\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K)(\mathbf{y}_{\tau,j}((\kappa+1)K))^T\right]^{-1}\times\sum_{\tau=0}^{t}\mathbf{y}_{\tau,j}((\kappa+1)K)x_\tau(j). \tag{3.12}$$

To make the notation more compact let $\mathbf{y}_{\tau,j}^{\kappa}(k):=\mathbf{y}_{\tau,j}(\kappa K+k)$, $\mathbf{v}_{\tau,j}^{\kappa,j'}(k)=\mathbf{v}_{\tau,j}^{j'}(\kappa K+k)$, $\mathbf{w}_{\tau,j}^{\kappa,j'}(k)=\mathbf{w}_{\tau,j}^{j'}(\kappa K+k)$ and $\mathbf{z}_{\tau,j}^{\kappa,j'}(k)=\mathbf{z}_{\tau,j}^{j'}(\kappa K+k)$. Then, updates $\mathbf{y}_{\tau,j}^{\kappa+1}(k)$ will be formed at sensor j for $k=1,\ldots,K$ by employing the ADMM.

The first step in ADMM, during coordinate descent cycle $\kappa+1$, updates the Lagrange multipliers using the gradient ascent iterations

$$\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k)=\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k-1)+c[\mathbf{y}_{\tau,j}^{\kappa+1}(k)-\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)], \tag{3.13}$$

$$\mathbf{w}_{\tau,j}^{\kappa+1,j'}(k)=\mathbf{w}_{\tau,j}^{\kappa+1,j'}(k-1)+c[\mathbf{y}_{\tau,j}^{\kappa+1}(k)-\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)], \tag{3.14}$$

for $j'\in\mathcal{N}_j$, $k=1,\ldots,K$ and $\tau=0,\ldots,t$. Note that $\mathbf{v}_{\tau,j}^{\kappa+1,j'}(0)=\mathbf{v}_{\tau,j}^{\kappa,j'}(K)$, thus coordinate cycle $\kappa+1$ starts using the most up-to-date $\mathbf{v}$'s from cycle κ. The same holds for the $\mathbf{y}$'s, $\mathbf{w}$'s and $\mathbf{z}$'s.

The second step involves a minimization of (3.8) wrt $\mathbf{y}_{\tau,j}$ while treating the rest optimization variables as fixed to their most up-to-date value.

$$\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)=\arg\min_{\mathbf{y}_{\tau,j}}\left[\mathcal{L}_a(\mathbf{y}_j,\mathbf{y}_{j'},\mathbf{v},\mathbf{w})\right] \\ =\arg\min_{\mathbf{y}_{\tau,j}}[\sum_{\tau=0}^{t}(x_\tau(j)-\mathbf{C}_{:j}^T(\kappa+1)\mathbf{y}_{\tau,j})^2 \tag{3.15}$$

$$+\sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^{t}\left[(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k))^T(\mathbf{y}_{\tau,j}-\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k))+(\mathbf{w}_{\tau,j}^{\kappa+1,j'}(k))^T(\mathbf{y}_{\tau,j}-\mathbf{z}_{\tau,j'}^{\kappa+1,j}(k))\right]$$

$$+0.5c\sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^{t}\left[\|\mathbf{y}_{\tau,j}-\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)\|_2^2+\|\mathbf{y}_{\tau,j}-\mathbf{z}_{\tau,j'}^{\kappa+1,j}(k)\|_2^2\right]. \quad (3.16)$$

After evaluating the gradient of the augmented Lagrangian with respect to $\mathbf{y}_{\tau,j}$ and setting it equal to zero (first-order optimality conditions) it follows

$$-2\mathbf{C}_{:j}(\kappa+1)\left[x_\tau(j)-\mathbf{C}_{:j}{}^T(\kappa+1)\mathbf{y}_{\tau,j}\right]+\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{\tau,j}^{\kappa+1,j'}+\mathbf{w}_{\tau,j}^{\kappa+1,j'})$$

$$+0.5c\textstyle\sum_{j'\in\mathcal{N}_j}2(\mathbf{y}_{\tau,j}-\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)+\mathbf{y}_{\tau,j}-\mathbf{z}_{\tau,j'}^{\kappa+1,j}(k))=0. \quad (3.17)$$

Then, from (3.17) it follows that for $j=1,\ldots,J$

$$\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)=\left[2\mathbf{C}_{:j}(\kappa+1)(\mathbf{C}_{:j}(\kappa+1))^T+2c|\mathcal{N}_j|\mathbf{I}\right]^{-1}$$

$$\times\left[2\mathbf{C}_{:j}(\kappa+1)x_\tau(j)-\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k)+\mathbf{w}_{\tau,j}^{\kappa+1,j'}(k))\right.$$

$$\left.+c\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)+\mathbf{z}_{\tau,j'}^{\kappa+1,j}(k))\right]. \quad (3.18)$$

where $\kappa=0,1,\ldots,$, $k=1,\ldots,K$ and $\tau=0,\ldots,t$ for $j'\in\mathcal{N}_j$.

The third step in ADMM involves forming the updates $\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)$. This is done by minimizing (3.8) with respect to $\mathbf{z}_{\tau,j}^{\kappa+1,j'}$, while fixing the other variables to their most up-to-date values

$$\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k+1)=\arg\min_{\mathbf{z}_{\tau,j}{}^{j'}}\left[\mathcal{L}_a(\mathbf{y}_{j'},\mathbf{v},\mathbf{w},\mathbf{y}_j(k+1))\right] \quad (3.19)$$

$$=\arg\min_{\mathbf{z}_{\tau,j}{}^{j'}}[\sum_{j=1}^{p}\sum_{\tau=0}^{t}(x_\tau(j)-\mathbf{C}_{:j}^T\mathbf{y}_{\tau,j}(k+1))^2$$

$$+\sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^{t}\left[(\mathbf{v}_{\tau,j}^{\kappa+1,j'})^T(\mathbf{y}_{\tau,j}(k+1)-\mathbf{z}_{\tau,j}^{j'})+(\mathbf{w}_{\tau,j}^{\kappa+1,j'})^T(\mathbf{y}_{\tau,j}(k+1)-\mathbf{z}_{\tau,j'}^{j})\right]$$

$$+0.5c\sum_{j'\in\mathcal{N}_j}\sum_{\tau=0}^{t}\left[\|\mathbf{y}_{\tau,j}(k+1)-\mathbf{z}_{\tau,j}^{j'}\|_2^2+\|\mathbf{y}_{\tau,j}(k+1)-\mathbf{z}_{\tau,j'}^{j}\|_2^2\right], \quad (3.20)$$

from which first-order optimality conditions result

$$
\begin{aligned}
&- (\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{w}_{\tau,j'}^{\kappa+1,j}(k)) \\
&\qquad - c(\mathbf{y}_{\tau,j}(k+1) - \mathbf{z}_{\tau,j}^{\kappa+1,j'}(k+1) + \mathbf{y}_{\tau,j'}(k+1) - \mathbf{z}_{\tau,j}^{\kappa+1,j'}(k)) = 0 \\
&\Leftrightarrow 2c\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k) - c(\mathbf{y}_{\tau,j'}(k+1) + \mathbf{y}_{\tau,j}(k+1)) = \mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{w}_{\tau,j'}^{\kappa+1,j}(k). \qquad (3.21)
\end{aligned}
$$

Then, it follows

$$
\mathbf{z}_{\tau,j}^{\kappa+1,j'}(k+1) = 0.5[\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) + \mathbf{y}_{\tau,j'}^{\kappa+1}(k+1)] + 0.5c^{-1}[\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) + \mathbf{w}_{\tau,j}^{\kappa+1,j'}(k)], \qquad (3.22)
$$

where $j = 1, \ldots, J$ and $j' \in \mathcal{N}_j$. Substituting (3.22) into the two recursions in (3.13), it follows that if the Lagrange multipliers are initialized such that $\mathbf{v}_{\tau,j}^{0,j'}(0) = -\mathbf{w}_{\tau,j'}^{0,j}(0)$, then $\mathbf{v}_{\tau,j}^{\kappa,j'}(k) = -\mathbf{w}_{\tau,j'}^{\kappa,j}(k)$ for all τ, κ and k, while

$$
\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) = \mathbf{v}_{\tau,j}^{\kappa+1,j'}(k-1) + 0.5c(\mathbf{y}_{\tau,j}^{\kappa+1}(k) - \mathbf{y}_{\tau,j'}^{\kappa+1}(k)), \qquad (3.23)
$$

for $j' \in \mathcal{N}_j$. Thus, sensor j only has to keep track of $\{\mathbf{v}_{\tau,j}^{\kappa,j'}(k)\}_{j' \in \mathcal{N}_j}$ since $\mathbf{w}_{\tau,j}^{\kappa,j'}(k) = -\mathbf{v}_{\tau,j'}^{\kappa,j}(k)$ becomes redundant.

Then, using (3.22) and $\mathbf{v}_{\tau,j}^{\kappa,j'}(k) = -\mathbf{w}_{\tau,j'}^{\kappa,j}(k)$ recursion (3.18) becomes

$$
\begin{aligned}
\mathbf{y}_{\tau,j}^{\kappa+1}(k+1) = &\left[2\mathbf{C}_{:j}(\kappa+1)(\mathbf{C}_{:j}(\kappa+1))^T + c|\mathcal{N}_j|\mathbf{I}\right]^{-1} \\
&\times \Big[2\mathbf{C}_{:j}(\kappa+1)x_\tau(j) - \textstyle\sum_{j' \in \mathcal{N}_j}(\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k) - \mathbf{v}_{\tau,j'}^{\kappa+1,j}(k)) \\
&\qquad + c\textstyle\sum_{j' \in \mathcal{N}_j}(\mathbf{y}_{\tau,j}^{\kappa+1}(k) + \mathbf{y}_{\tau,j'}^{\kappa+1}(k))\Big], \qquad (3.24)
\end{aligned}
$$

where the $\mathbf{z}_{\tau,j}^{\kappa,j'}(k)$ variables have been eliminated. Using the convergence claims in [33] turns out that if an infinite number of consensus iterations ($K \to \infty$) is applied during coordinate cycle $\kappa + 1$ then consensus is reached in the sense that $\lim_{k\to\infty} \mathbf{y}_{\tau,j}^{\kappa+1}(k) = (\mathbf{C}(\kappa+1)\mathbf{C}(\kappa+1)^T)^{-1}\mathbf{C}(\kappa+1)\mathbf{x}_\tau$ (the principal component vectors) for all sensors $j = 1, \ldots, J$, while $\mathbf{C}(\kappa+1) := [\mathbf{C}_{:1}(\kappa+1) \ldots \mathbf{C}_{:J}(\kappa+1)]$.

Recursions (3.12), (3.23) and (3.24) constitute a *batch* D-PCA approach, whereby sensor j keeps track of

i) $\mathbf{C}_{:j}(k+1) \in \mathbb{R}^{r\times 1}$, that estimates the jth row of $\mathbf{U}_{x,r}$;

ii) the multipliers $\{\mathbf{v}_{\tau,j}^{\kappa,j'}(k)\}_{j'\in\mathcal{N}_j}$; and

iii) the principal components vectors $\mathbf{y}_{\tau,j}^{\kappa}(k+1)$ for $\tau = 0,\ldots,t$.

In a batch setting, sensors first gather $t+1$ measurements (t is fixed), namely $\mathbf{x}_0,\ldots,\mathbf{x}_t$, and then employ the D-PCA algorithm. During coordinate descent cycle $\kappa+1$ sensor j first forms $\mathbf{C}_{:j}(\kappa+1)$ via (3.12), using its local principal components vector updates $\mathbf{y}_{\tau,j}^{\kappa}(K)$ for $\tau = 0,\ldots,t$. Then, sensor j runs K consensus iterations by carrying out (3.23) and (3.24). Specifically, during consensus iteration $k+1$ and coordinate cycle $\kappa+1$, it receives from its neighbors $j'\in\mathcal{N}_j$ the $r\times 1$ vectors $\mathbf{y}_{\tau,j'}^{\kappa+1}(k)$ and updates its multipliers $\mathbf{v}_{\tau,j}^{\kappa+1,j'}(k)$ via (3.23). Then, sensor j receives $\{\mathbf{v}_{\tau,j'}^{\kappa+1,j}(k)\}_{j'\in\mathcal{N}_j}$ which are used along with $\{\mathbf{y}_{\tau,j'}^{\kappa+1}(k)\}_{j'\in\mathcal{N}_j}$ to form $\mathbf{y}_{\tau,j}^{\kappa+1}(k+1)$ via (3.24). For an increasing number of consensus iterations $K\to\infty$ and coordinate descent cycles ($\kappa\to\infty$), $\mathbf{C}(\kappa)$ and $\mathbf{y}_{\tau,j}(\kappa)$ converge at least to a stationary point of the PCA cost in (3.5). This can be established using the convergence properties of block coordinate descent and ADMM in [36] and [33, 34], respectively.

3.2.2 Adaptive D-PCA

In situations where real time processing of data is of utmost importance as compared to estimation performance, a batch implementation will not be suitable. This particularly comes to play in cases where the number of measurements $t+1$ is not fixed. This is because every sensor has to carry out $(t+1)K$ consensus iterations for updating $\{\mathbf{y}_{\tau,j}^{\kappa}\}_{\tau=0}^{t}$. For an increasingly large t, i.e., with sensors collecting a large amount of data, there comes an increasing demand of computational, communication and memory capabilities across sensors for a batch implementation. Building on batch D-PCA we derive an adaptive D-PCA scheme that is capable to process data online, while having a manageable computational,

communication and memory cost [37]. In batch D-PCA the separable PCA cost in (3.6) is time-invariant. For a constant t, the batch D-PCA consisted of multiple coordinate descent cycles that could be run until, e.g., the PCA cost does not decrease below a desired limit. In batch D-PCA the separable PCA cost in (3.6) is time-invariant, i.e. t is fixed. For a constant t batch D-PCA consisted of multiple coordinate descent cycles that could be run until, e.g., the PCA cost does not decrease below a desired threshold. In an adaptive setting sensors acquire new data at every time instant t, thus t is increasing and the cost (3.6) will be augmented with new data terms. Thus, it is pertinent at t to apply a small number of coordinate cycles; here one coordinate cycle per t is employed. Thus, the time and coordinate cycle indices coincide, i.e., $\kappa = t$. During t there will be K nested consensus iterations carrying out (3.23) and (3.24). Different from batch D-PCA, (3.23) and (3.24) will be carried out only for the most recent multipliers and principal vectors $\{\mathbf{v}_{t,j}^{j'}, \mathbf{y}_{t,j}\}_{j=1}^{p}$, and not for all $\tau = 0, \ldots, t$ as in the batch implementation.

During $t+1$, adaptive D-PCA updates $\mathbf{C}_{:j}$ as

$$\mathbf{C}_{:j}(t+1) = \left[\sum_{\tau=0}^{t} \mathbf{y}_{\tau,j}(K)\mathbf{y}_{\tau,j}^T(K)\right]^{-1} \sum_{\tau=0}^{t} \mathbf{y}_{\tau,j}(K)x_\tau(j), \tag{3.25}$$

and employs K consensus iterations for updating $\mathbf{y}_{t,j}$ and $\mathbf{v}_{t,j}$ as

$$\mathbf{v}_{t+1,j}^{j'}(k) = \mathbf{v}_{t+1,j}^{j'}(k-1) + 0.5c[\mathbf{y}_{t+1,j}(k) - \mathbf{y}_{t+1,j'}(k)], \tag{3.26}$$

$$\begin{aligned}\mathbf{y}_{t+1,j}(k+1) = &\left[2\mathbf{C}_{:j}(t+1)(\mathbf{C}_{:j}(t+1))^T + c|\mathcal{N}_j|\mathbf{I}\right]^{-1} \\ &\times \Big[2\mathbf{C}_{:j}(t+1)x_{t+1}(j) - \textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{t+1,j}^{j'}(k) - \mathbf{v}_{t+1,j'}^{j}(k)) \\ &+c\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{y}_{t+1,j}(k) + \mathbf{y}_{t+1,j'}(k))\Big],\end{aligned} \tag{3.27}$$

where $k = 1, \ldots, K$ for (3.26), while coordinate cycle superscripts have been removed since one cycle takes place per t. The multipliers at $t = 0$, namely $\mathbf{v}_{j,0}(0)$, are initialized randomly, whereas at time instant $t > 0$ warm-starts are employed to set $\mathbf{v}_{t,j}(0) = \mathbf{v}_{t-1,j}(K)$. Further, during instant t the $\mathbf{y}_{t,j}$ variables are initialized as $\mathbf{y}_{t,j}(0) = \mathbf{C}_{:j}(t)x_t(j)$.

Let $\mathbf{M}_{x,t}$ denote the matrix inverted in (3.25), and $\mathbf{m}_{xy,t} := \sum_{\tau=0}^{t} \mathbf{y}_{\tau,j}(K)x_\tau(j)$. Since consensus is applied only for $\mathbf{y}_{t,j}$ and $\mathbf{v}_{t,j}^{j'}$, the updates $\mathbf{y}_{\tau,j}(K)$ and $\mathbf{v}_{\tau,j}^{j'}(K)$ for $\tau < t$ remain constant for the time instances $\tau+1, \tau+2, \ldots$. Thus, $\mathbf{M}_{x,t}$ and $\mathbf{m}_{xy,t}$ can be adaptively updated at sensor j as $\mathbf{M}_{x,t} = \mathbf{M}_{x,t-1} + \mathbf{y}_{t,j}(K)\mathbf{y}_{t,j}^T(K)$ and $\mathbf{m}_{xy,t} = \mathbf{m}_{xy,t-1} + \mathbf{y}_{t,j}(K)x_t(j)$, respectively.

Summarizing, adaptive D-PCA will involve the following steps during $t+1$: Sensor j updates recursively $\mathbf{M}_{x,t}$ and $\mathbf{m}_{xy,t}$ and uses them to update $\mathbf{C}_{:j}(t+1)$ via (3.25). Then, K consensus recursions are employed in order to obtain $\mathbf{y}_{t+1,j}(K)$. To this end, the initialization $\mathbf{y}_{t+1,j}(0) = \mathbf{C}_{:j}(t+1)x_{t+1}(j)$ and $\mathbf{v}_{t+1,j}(0) = \mathbf{v}_{t,j}(K)$ takes place. During consensus iteration k sensor j receives from its neighbors $j' \in \mathcal{N}_j$ vectors $\mathbf{y}_{t+1,j'}(k)$ and updates its multipliers $\mathbf{v}_{t+1,j}^{j'}(k)$ via (15). Then, sensor j receives the multipliers $\{\mathbf{v}_{t+1,j'}^{j}(k)\}_{j'\in\mathcal{N}_j}$ which are used along with $\{\mathbf{y}_{t+1,j'}(k)\}_{j'\in\mathcal{N}_j}$ to form $\mathbf{y}_{t,j}(k+1)$ via (16). Once $\mathbf{y}_{t+1,j}(K)$ are formed across sensors, the process is repeated.

3.3 Inter-Sensor Communication Noise

Thus far, no inter-sensor communication noise was considered in the information exchanges between neighboring sensors. The inter-sensor links so far have been assumed to be ideal. However, in practice data communication between sensors suffer from noise that will contaminate the quantities being exchanged between say a sensor j and another sensor j' for $j \neq j'$, and $j' \in \mathcal{N}_j$. The presence of additive noise in the inter-sensor links will affect the quantities $\mathbf{y}_{t,j}$ and $\mathbf{v}_{t,j}$ being exchanged between neighboring sensors. Specifically, if a sensor j transmits the $r \times 1$ vectors $\mathbf{y}_{t+1,j'}(k)$ and $\mathbf{v}_{t+1,j}^{j'}(k)$, then a neighboring sensor $j' \in \mathcal{N}_j$ will receive the 'noisy' vectors $\mathbf{y}_{t+1,j'}(k) + \boldsymbol{\eta}_j^{t+1,j'}(k)$ and $\mathbf{v}_{t+1,j'}^{j}(k) + \boldsymbol{\zeta}_j^{t+1,j'}(k)$, where $\boldsymbol{\eta}_j^{t+1,j'}(k)$ and $\boldsymbol{\zeta}_j^{t+1,j'}(k)$ correspond to the zero-mean communication noise affecting the

link from sensor j to sensor j' during transmission of the $\mathbf{y}$ and $\mathbf{v}$ vectors, respectively. Then, the updating formulas in (3.26) and (3.27) will be adjusted as

$$\mathbf{v}_{t+1,j}^{j'}(k) = \mathbf{v}_{t+1,j}^{j'}(k-1) + 0.5c[\mathbf{y}_{t+1,j}(k) - (\mathbf{y}_{t+1,j'}(k) + \boldsymbol{\eta}_j{}^{t+1,j'}(k))], \tag{3.28}$$

$$\begin{aligned}\mathbf{y}_{t+1,j}(k+1) &= \left[2\mathbf{C}_{:j}(t+1)(\mathbf{C}_{:j}(t+1))^T + c|\mathcal{N}_j|\mathbf{I}\right]^{-1} \\ &\times \Big[2\mathbf{C}_{:j}(t+1)x_{t+1}(j) - \textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{v}_{t+1,j}^{j'}(k) - (\mathbf{v}_{t+1,j'}^{j}(k) + \boldsymbol{\zeta}_j{}^{t+1,j'}(k)) \\ &+c\textstyle\sum_{j'\in\mathcal{N}_j}(\mathbf{y}_{t+1,j}(k) + (\mathbf{y}_{t+1,j'}(k)) + \boldsymbol{\eta}_j{}^{t+1,j'}(k))\Big]\end{aligned} \tag{3.29}$$

As will be corroborated by numerical tests, the D-PCA algorithm comprising of recursions (3.25) and (3.28) is resilient to communication noise. On the contrary existing distributed PCA approaches [38] do not exhibit robustness in the presence of noise and diverge as will be demonstrated by simulations. Moreover, in the absence of noise (ideal links) D-PCA will outperform [38] both in terms of convergence speed, as well as steady-state principal eigenspace estimation error.

3.4 Communication and Computational Costs

The communication and computational costs associated with the adaptive D-PCA scheme is now evaluated, summarized in (3.25), (3.26) and (3.29) , and compare it with the related approach in [26]. The computational complexity for carrying out (3.25) is $\mathcal{O}(r^2)$ which is dictated by the inversion of $\mathbf{M}_{x,t}$ which can be done by employing the matrix inversion lemma [39, pg. 571]. Updating of $\mathbf{v}_{t,j}^{j'}$ and $\mathbf{m}_{xy,t}$ has a complexity of the order of $\mathcal{O}(r)$, while the associated complexity for forming $\mathbf{y}_{t,j}$ is $\mathcal{O}(r^2)$ which is imposed by the matrix inversion that again can be carried out using the matrix inversion lemma. Thus, the computational complexity per time instant t and consensus iteration in adaptive D-PCA is $\mathcal{O}(r^2)$. The computational complexity of the algorithm proposed in [26], abbreviated as S-PCA, is also $\mathcal{O}(r^2)$.

For the communication costs associated with D-PCA and S-PCA, sensor j in D-PCA has to transmit $r(|\mathcal{N}_j|+1)$ scalars per consensus iteration corresponding to the entries of the multipliers $\{\mathbf{v}_{t,j}^{j'}(k)\}_{j\in\mathcal{N}_j}$ and the local estimate $\mathbf{y}_{t,j}(k)$. Given that at each time instant t there are K consensus iterations taking place, the total transmission load per sensor during time instant t is $rK(|\mathcal{N}_j|+1)$. In S-PCA each sensor has to transmit $rK(r+1)$ to carry out consensus-iterations involving $r\times r$ matrices and $r\times 1$ vectors. If $r<|\mathcal{N}_j|$ is smaller than the size of the single-hop neighborhoods then S-PCA has a smaller transmission cost, whereas if $r>|\mathcal{N}_j|$ D-PCA will have an advantage. When considering the reception cost it can be seen that in D-PCA each sensor receives $2rK|\mathcal{N}_j|$ scalars during K consensus iterations pertaining to the vectors $\{\mathbf{v}_{t,j'}^{j}(k),\mathbf{y}_{t,j'}(k)\}_{j'\in\mathcal{N}_j}$. However, in S-PCA sensor j receives $(r+1)rK|\mathcal{N}_j|\geq 2rK|\mathcal{N}_j|$ due to the reception of $\mathcal{N}_j$ $r\times r$ matrices and $r\times 1$ vectors per consensus iteration. Thus, D-PCA has a smaller reception cost for $r>1$. Even though the transmission cost of D-PCA may be greater than the one in S-PCA, it will be corroborated via numerical examples that the higher transmission cost in D-RLS pays off in improved convergence rates and steady-state performance. The work in [33] thoroughly studied why ADMM exhibits better convergence properties over consensus-averaging [40] when it comes to least-squares estimation [33]. Similar arguments can be carried over the present PCA setting supporting the better convergence properties of ADMM observed via simulations. In the next chapter we apply the D-PCA algorithm in denoising in a distributed fashion data gathered across spatially scattered sensors.

3.5 Comparison of bridge/ *bridgeless* sensor network communication costs

Here, we compare the sensor network communication costs of both a sensor network with a bridge and that without a bridge and the motivation behind each of the two schemes. In both cases, we consider the K consensus iterations and rank r.

Considering the transmission cost, for a network with bridges, each bridge has a transmission cost corresponding to $rK < |\mathcal{N}_j + 1|$ scalars. The simple sensors in the same network have transmission cost corresponding to $rK|\mathcal{B}_j|$ scalars. Whereas, in the network without bridges, each sensor has a transmission cost corresponding to $rK < |\mathcal{N}_j + 1|$ scalars. So, in terms of transmission cost, the bridge sensors in a network with bridges have the same cost as compared to the sensors of a network without bridges, at the expense of the simple sensors which have a lower transmission cost.

In terms of the reception cost, for a network with bridges, each bridge has a reception cost corresponding to $2rK < |\mathcal{N}_j|$ scalars. The simple sensors in the same network have reception cost corresponding to $2rK|\mathcal{B}_j|$ scalars. Whereas, in the network without bridges, each sensor has a transmission cost corresponding to $2rK < |\mathcal{N}_j|$ scalars. Therefore, in terms of reception cost, the bridge sensors in a network with bridges have the same communication cost as compared to the sensors of a network without bridges, at the expense of the simple sensors which have a lower reception cost.

Based on the analysis above, a network with bridges is ideal for networks in which a few sensors have higher memory and computation capabilities than other sensors, while network without bridges are ideal when sensors have the same computational power and memory.

CHAPTER 4

DISTRIBUTED DATA DENOISING

The amount of useful information extracted from measurement data is often hampered by the presence of noise. Even the effectiveness of statistical analysis methods such as PCA sometimes degrade fast in very low signal-to-noise ratio (SNR) settings. Denoising (or noise removal) is therefore a crucial step to improve the the information content of the low dimension data obtained from these statistical methods. Although not much work has been done on this, especially in a distributed setting, a few denoising algorithms have been put forward. The work in [41] suggested the use of wavelets as a tool for brain electrical activity analysis and denoising. A thresholding algorithm was introduced in [42] to reduce noise in the wavelet domain. However, these methods require high signal amplitudes to be able to effectively distinguish between noise and signal-related wavelet coefficients in single trials. The approach in [43] modified the original a posteriori Wiener filter to find accurate filter settings. The work in [44] proposed a non-linear approach to transient signal denoising with better filter settings. Other denoising approaches considered other statistical methods for determining a particular matrix of data and its delayed version for denoising, e.g., see [45]. [46] developed a batch based denoising approach where each node had to have access to a specific sensor group and not just single hop communications. A distributed PCA denoising algorithm is proposed in this chapter. The principal signal eigenspace is estimated in a distributed fashion and the sensor data are projected on the estimated signal space to remove noise and obtain better signal estimates. Denoising is then performed by projecting the sensor data onto a subset of the principal eigenvectors of

data covariance matrix. The denoising task is performed whilst maintaining the distributed setting being considered in earlier chapters.

4.1 Problem Statement

Consider an ad hoc sensor network consisting of J sensors taking measurements of a particular deterministic quantity. Each sensor j takes measurements at time instant τ, where $\tau = 1, \ldots, t$ and as mentioned on previous chapters is able to communicate only with its single-hop neighbors, i.e. *ad-hoc* neighbors j' for $\mathcal{N}_j$. The measurements acquired across sensors are stacked in the vector $\mathbf{x}_\tau^n := [x_\tau^n(1), \ldots, x_\tau^n(p)]^T$, for $\tau = 1, \ldots, t$ are assumed to be zero-mean. Similarly, let the vector $\mathbf{x}_\tau := [x_\tau(1), \ldots, x_\tau(p)]^T$ denote the information part of the sensor data for $\tau = 1, \ldots, t$ such that

$$\mathbf{x}_\tau^n = \mathbf{x}_\tau + \mathbf{n}_\tau, \tag{4.1}$$

where $\mathbf{n}_\tau := [n_\tau(1), \ldots, n_\tau(p)]^T$ is a dataset containing the noise variables. The sensor data covariance matrix, namely $\mathbf{\Sigma}_{x^n}$, can then be written as

$$\mathbf{\Sigma}_{x^n} = \mathbb{E}[\mathbf{x}_\tau^n (\mathbf{x}_\tau^n)^T] = \mathbb{E}[\mathbf{x}_\tau \mathbf{x}_\tau^T + \mathbf{x}_\tau \mathbf{n}_\tau^T + \mathbf{n}_\tau \mathbf{x}_\tau^T + \mathbf{n}_\tau \mathbf{n}_\tau^T] \tag{4.2}$$

Since the *information* signal $\mathbf{x}_\tau$ in $\mathbf{x}_\tau^n$ is uncorrelated with the noise $\mathbf{n}_\tau$ (for $\tau = 1, .., t$), from (4.2) it follows that

$$\mathbf{\Sigma}_{x^n} = \mathbf{\Sigma}_x + \mathbf{\Sigma}_n, \tag{4.3}$$

where $\mathbf{\Sigma}_n$ denotes the measurement noise covariance matrix. Here we assume that the sensing noise across sensors is spatially uncorrelated, i.e., $\mathbf{\Sigma}_n = \sigma_n^2 \mathbf{I}$ where σ_n^2 denotes the sensing noise variance. Let $\mathbf{\Sigma}_x = \mathbf{H} \mathbf{\Sigma}_s \mathbf{H}^T$ denote the singular value decomposition (svd) of information covariance matrix $\mathbf{\Sigma}_x$, where $\mathbf{H} \in \mathbb{R}^{J \times J}$ is the orthonormal matrix that

contains the eigenvector in its columns, while the diagonal $\mathbf{\Sigma}_s$ contains the corresponding eigenvalues. Thus, (4.3) becomes

$$\mathbf{\Sigma}_{x^n} = \mathbf{H}\mathbf{\Sigma}_s\mathbf{H}^T + \sigma_n^2\mathbf{I}. \tag{4.4}$$

Specifically, assume that the sensed field is formed by r zero-mean uncorrelated stationary sources represented by the signals $s_\rho(t)$. In detail, the informative part of the sensor data ($\mathbf{x}_\tau^n$) is formed as

$$\mathbf{x}_\tau = \sum_{\rho=1}^{r} \mathbf{h}_\rho s_\rho(\tau) = \mathbf{H}_{:,1:r}\mathbf{s}_\tau, \tag{4.5}$$

where $s_\rho(\tau)$ corresponds to the signal emitted by the ρth source, while the different sources are uncorrelated. As assumed before, the sensor data follow the noisy linear model $\mathbf{x}_\tau^n = \mathbf{x}_\tau + \mathbf{n}_\tau$. Further, let the singular value decomposition of $\mathbf{H} := [\mathbf{h}_1 \dots \mathbf{h}_r] \in \mathbb{R}^{p\times r}$ be written as $\mathbf{H} = \mathbf{U}_h\mathbf{S}_h\mathbf{V}_h^T$, where $\mathbf{S}_h \in \mathbb{R}^{p\times r}$ is diagonal and $\mathbf{U}_h \in \mathbb{R}^{p\times p}$ and $\mathbf{V}_h \in \mathbb{R}^{r\times r}$ orthonormal matrices.

The information signal covariance matrix $\mathbf{\Sigma}_x = \mathbf{U}_h\mathbf{\Lambda}_s\mathbf{U}_h^T$, where $\mathbf{U}_h$ corresponds to the eigenvector matrix and $\mathbf{\Lambda}_s := \mathbf{S}_h\mathbf{D}_s\mathbf{S}_h^T$ the corresponding eigenvalue matrix where $\mathbf{D}_s$ is diagonal matrix whose diagonal entries correspond to the variance of source signals $s_\rho(\tau)$. This further implies that $\mathbf{\Sigma}_x = \mathbf{U}_h(\mathbf{\Lambda}_s + \sigma_n^2\mathbf{I})\mathbf{U}_h^T$, where $\mathbf{\Lambda}_s = \text{diag}(\mathbf{\Lambda}_{s,r}, \mathbf{0}_{p-r})$ since only r eigenvalues in covariance $\mathbf{\Sigma}_x$ will be strictly positive given that there are r field sources. Here we consider subspace-based denoising which performs projection of the data onto the principal signal subspace, i.e.,

$$\mathbf{U}_{h,r}\mathbf{U}_{h,r}^T\mathbf{x}_\tau^n = \mathbf{H}\mathbf{s}_\tau + \mathbf{U}_{h,r}\mathbf{U}_{h,r}^T\mathbf{n}_\tau, \tag{4.6}$$

where $\mathbf{U}_{h,r}$ corresponds to the r principal eigenvectors of $\mathbf{\Sigma}_x$, i.e., $\mathbf{U}_h = [\mathbf{U}_{h,r} \quad \mathbf{U}_{h,r-p}]$ where $\mathbf{U}_{h,r-p}$ corresponds to the eigenvectors multiplied by zero eigenvalues. It follows readily that the covariance of the projected noise $\mathbf{U}_{h,r}\mathbf{U}_{h,r}^T$ has rank equal to r, where $r < p$ since the number of field sources in many practical settings is much smaller than the

number of sensors. The total noise variance of $\mathbf{U}_{h,r}\mathbf{U}_{h,r}^T$ becomes $r\sigma_n^2$, which is small than $p\sigma^2$ (the total noise variance before data projection in (4.6)). Thus, the overall effect of the noise is reduced on the projected signal while the information part $\mathbf{H}\mathbf{s}_\tau$ in (4.5) remains unchanged in (4.6).

It becomes apparent that the main task in denoising via projection in (4.6) is to estimate the principal eigenspace $\mathbf{U}_{h,r}$. This is where the D-PCA approach proposed in Chapter 3 will be utilized to come up with a distributed way to estimate $\mathbf{U}_{x,r}$ across sensors and denoise the sensor data. Different techniques will be compared with D-PCA in terms of the SNR achieved in the processed data after performing projection (post-SNR comparisons).

4.2 Distributed Implementation

In order to estimate the principal eigenspace $\mathbf{U}_{h,r}$ for data denoising we start with the centralized cost function

$$\hat{\mathbf{C}} = \arg\min (t+1)^{-1} \sum_{\tau=0}^{t} \|\mathbf{x}_{\tau^n} - \mathbf{C}^T\mathbf{C}\mathbf{x}_{\tau^n}\|_2^2, \tag{4.7}$$

where $\hat{\mathbf{C}}$ corresponds to the estimate for $\mathbf{U}_{h,r}$. Again by setting $\mathbf{y}_\tau = \mathbf{C}\mathbf{x}_{\tau^n}$ in (4.7), we have

$$(\hat{\mathbf{C}}, \{\hat{\mathbf{y}}_\tau\}_{\tau=0}^t) = \arg\min_{\mathbf{C},\mathbf{y}_\tau} (t+1)^{-1} \sum_{\tau=0}^{t} \|\mathbf{x}_{\tau^n} - \mathbf{C}^T\mathbf{y}_\tau|_2^2, \tag{4.8}$$

Here, we consider the adaptive setting and therefore still use equations (3.25), (3.26) and (3.27) to estimate the eigenspace $\mathbf{U}_{h,r}$.

The performance metrics that will be used to compared the different techniques for estimating $\mathbf{U}_{h,r}$ and performing denoising will be the SNR before and after data projection. Specifically, the SNR will be evaluated as

$$\text{SNR} = 10log_{10}\frac{tr(\mathbf{S}_x)}{tr(\mathbf{S}_{\text{noise}})} \tag{4.9}$$

where $tr(.)$ is the trace of the matrix, while

$$\mathbf{S}_x = \frac{1}{N}\sum_{\tau=1}^{N}\mathbf{x}_\tau(\mathbf{x}_\tau)^T, \tag{4.10}$$

corresponds to power in the information part $\mathbf{x}_\tau$, while

$$\mathbf{S}_{noise} = \frac{1}{N}\sum_{\tau=1}^{N}(y_\tau - \mathbf{x}_\tau)(y_\tau - \mathbf{x}_\tau)^T. \tag{4.11}$$

. where $y_\tau = \mathbf{U}_{h,r}\mathbf{U}_{h,r}^T\mathbf{x}_\tau^n$ corresponds to the projected sensor data.

Note that the SNR quantity given in (4.9) denotes the post-projection SNR denoted from now on as post-SNR. Depending on which method will be used to estimate $\mathbf{U}_{h,r}$ different post-SNR quantities will be obtained and extensive numerical tests in Chapt. 5 will advocate that our approach outperforms related alternatives. Setting $y_\tau = \mathbf{x}_\tau^n$ in (4.11) gives the pre-projection (before denoising) SNR which is denoted as pre-SNR. As a benchmark on how well the different distributed methods do the so called actual-SNR will be computed by using the true principal eigenspace $\mathbf{U}_{h,r}$ by applying singular value decomposition to the ensemble covariance matrix $\mathbf{\Sigma}_x$. Note that actual-SNR will give the highest possible post-SNR that can be achieved via projection denoising since the true principal eigenspace is used during data projection. Further, in Chapter 5 our D-PCA method will be compared with the distributed subspace estimation technique proposed in [38], and it will become apparent that our methodology achieved a better performance especially in noisy environments.

CHAPTER 5

NUMERICAL TESTS AND DISCUSSION

Here, we test the convergence properties of the two different distributed algorithms derived in this thesis; Distributed Principal Component Analysis (D-PCA) with a bridge sensor setting, and D-PCA framework without bridge sensors. We further test the robustness of the *bridgeless* network in the presence of noise across communication links. In each of these two settings, convergence is compared with the distributed in [38], abbreviated as S-PCA. Furthermore, we consider the *bridgeless* D-PCA is tested in a denoising application that involves both real and synthetic data. S-PCA is also applied in the denoising setting and compared with D-PCA. Then, conclusions are drawn about the advantages offered by the proposed algorithmic framework given here.

5.1 D-PCA in network with bridge sensors

Here we consider a sensor network with containing $p = 16$ sensors. Sensors are randomly placed in a unit square $[0, 1] \times [0, 1]$ with uniform distribution. The communication range of the network is set equal to 0.3, i.e., two sensors communicate as long as the distance is less than $d = 0.3$. Each sensor collects $t = 2000$ observations. Zero mean data is randomly generated for 2000 time instants for the 16 sensors and without observation noise. This was tested with $k = 5$ consensus (ADMM) iterations per time instant t. The parameter c in D-PCA is set $c = 4$ here, while the step-size in S-PCA was set to $\gamma = 10^{-3}$ The figure of merit used to compare different schemes will be the subspace projection error which is defined as $e(t) := \|\mathbf{C}^T(t)(\mathbf{C}(t)\mathbf{C}^T(t))^{-1}\mathbf{C}(t) - \mathbf{U}_{x,r}\mathbf{U}_{x,r}^T\|_F^2$, where $\|\cdot\|_F$ stands for the Frobenius norm operator and $\mathbf{C}(t)$ corresponds to the estimate of $\mathbf{U}_{x,r}$.

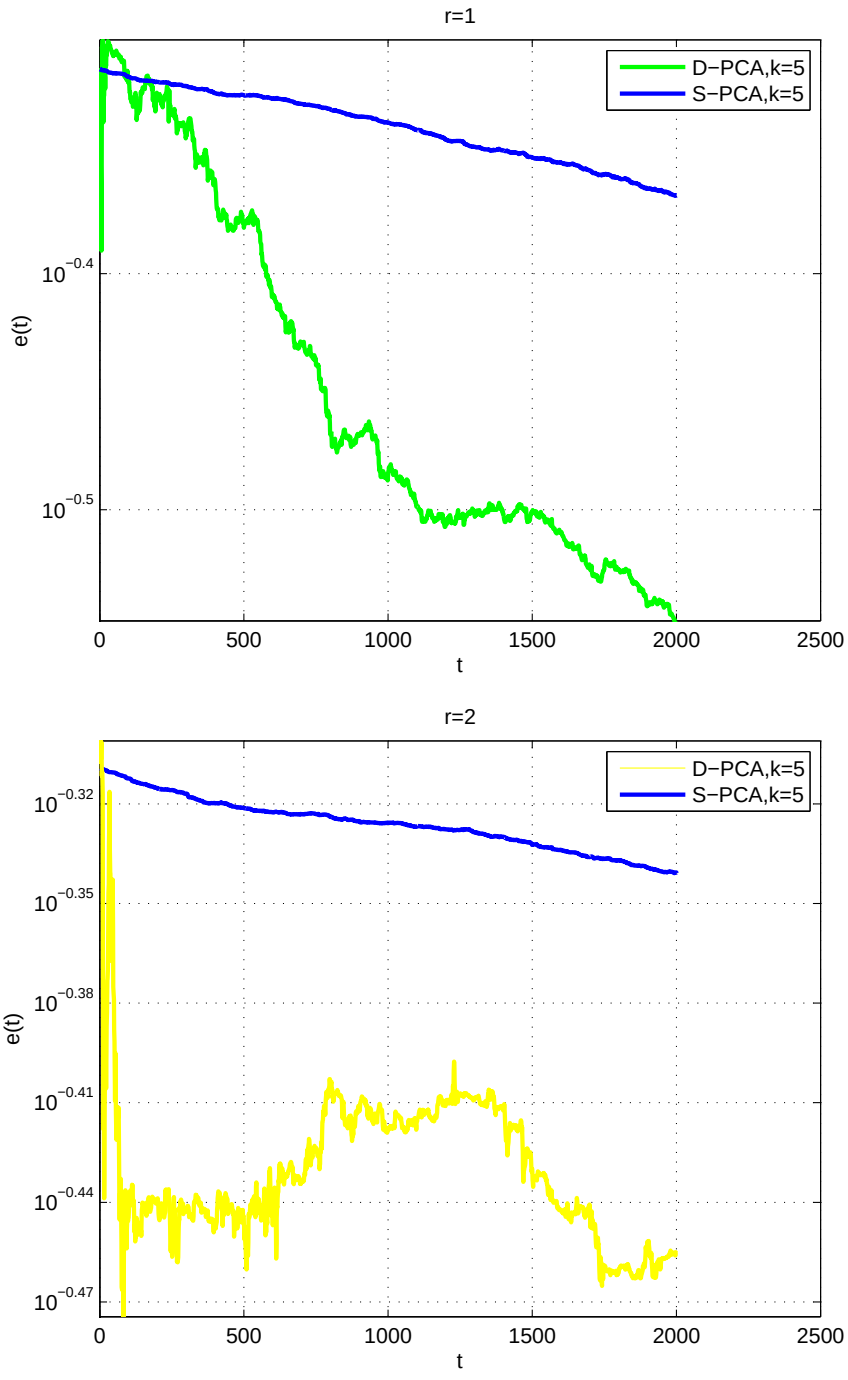

Figure 5.1. Subspace projection estimation error $e(t)$ vs. time index t for r=1 (top); and r=2 (bottom) in a setting with bridge sensors.

Figure 5.1 (top) shows $e(t)$ for the case where D-PCA and S-PCA are estimating $r = 1$ principal eigenvectors for both D-PCA and S-PCA. As it can be seen, D-PCA exhibits a much faster convergence rate than S-PCA, while it converges to a lower steady state error $e(t)$. Fig. 5.1 (bottom) depicts $e(t)$ for $r = 2$. In this case we test D-PCA and S-PCA for $K = 5$ consensus iterations. Again $e(t)$ in adaptive D-PCA decreases at a faster rate than in the case of S-PCA.

5.2 D-PCA in a network without bridge sensors

Here we compare D-PCA with S-PCA for a bridgeless sensor network setting while varying parameters such as the number of consensus iterations, number of principal components being estimated. We still consider a sensor network containing $p = 16$ sensors, and with sensors randomly placed in a unit square $[0, 1] \times [0, 1]$ with uniform distribution. The communication range of the network is $d = 0.3$. Each sensor collects $t = 7000$ observations. In the next simulations the subspace projection error $e(t) := \|\mathbf{C}^T(t)(\mathbf{C}(t)\mathbf{C}^T(t))^{-1}\mathbf{C}(t) - \mathbf{U}_{x,r}\mathbf{U}_{x,r}^T\|_F^2$ and the post-SNR (see Chapter 4) values are used as performance metrics to compared different principal eigenspace estimation methods.

5.2.1 Subspace Estimation error vs. iteration index

Here, zero mean data is randomly generated for 7000 time instants for the 16 sensors and without observation noise. These were tested with $K = 5$, or $K = 12$ consensus (ADMM) iterations per time instant t. The parameter c in D-PCA is set $c = 4$ here, while the step-size in S-PCA was set to $\gamma = 10^{-3}$. Figure 5.2 (top) shows $e(t)$ for the case where D-PCA and S-PCA are estimating $r = 1$ principal eigenvectors for both D-PCA and S-PCA. As it can be seen, D-PCA exhibits a much faster convergence rate than S-PCA, while it converges to a lower steady state error $e(t)$. Increasing the number of consensus iterations

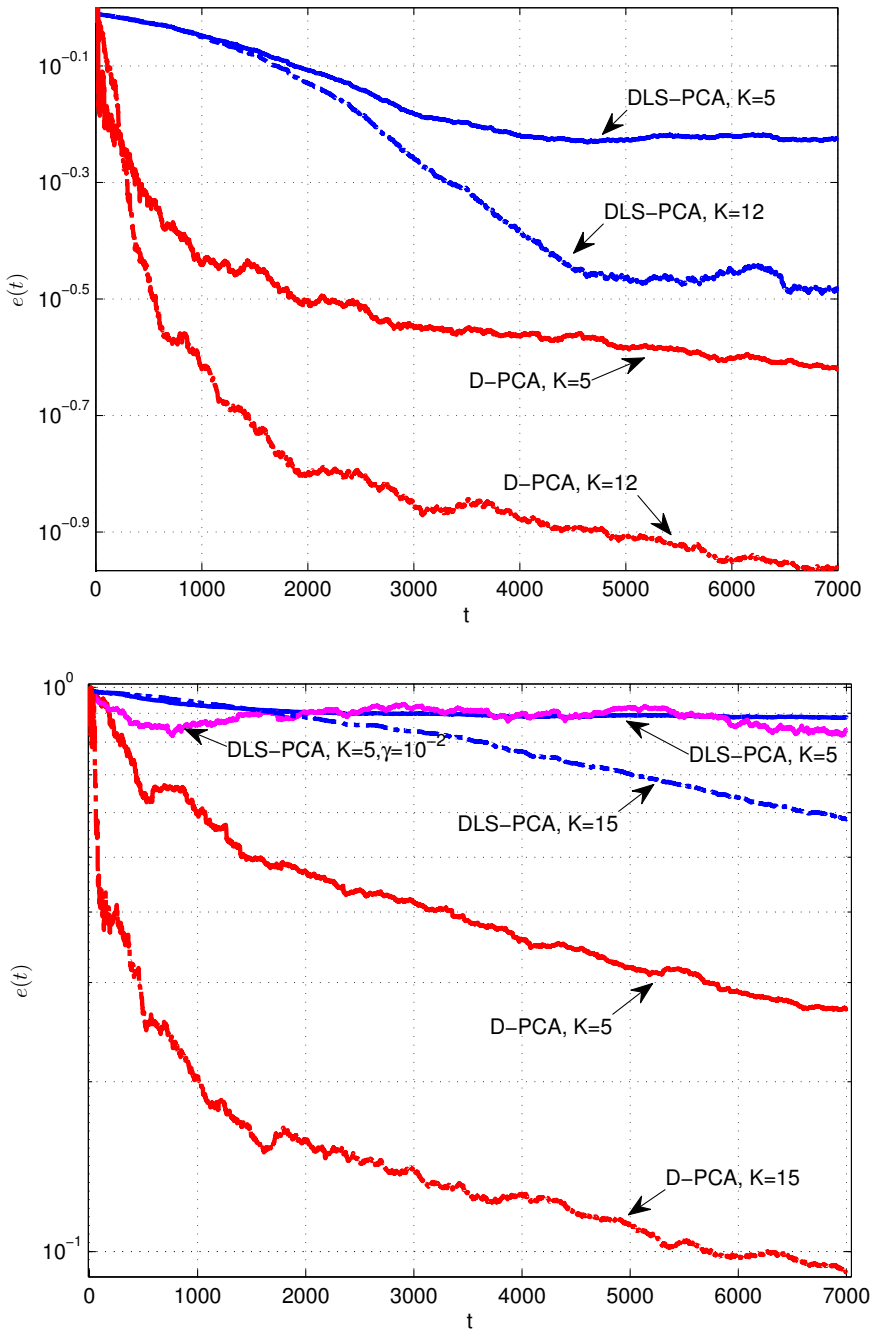

Figure 5.2. Subspace projection estimation error $e(t)$ vs. time index t for r=1 (top); and r=2 (bottom) in a network without bridge sensors.

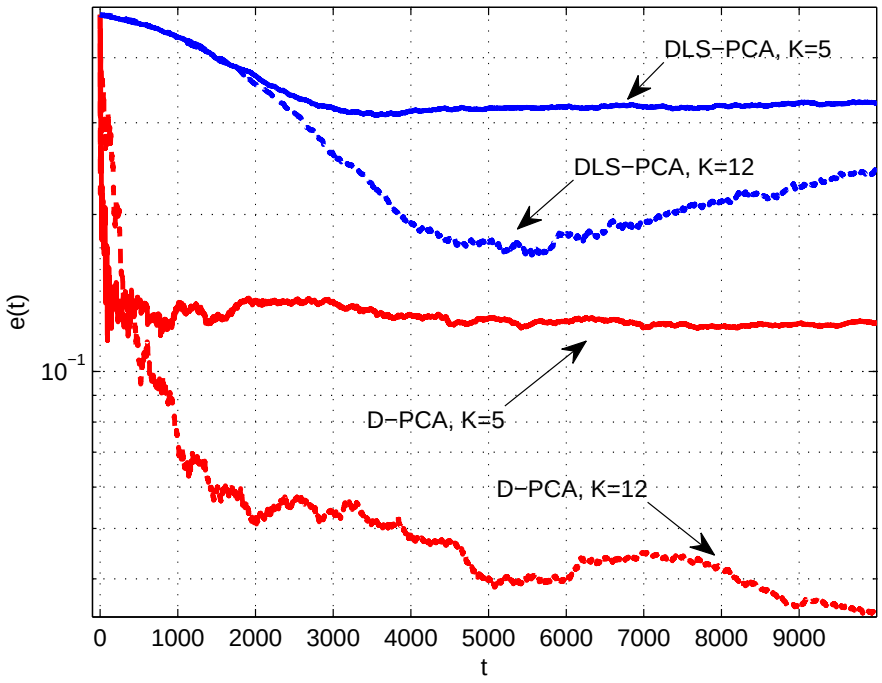

Figure 5.3. Subspace projection estimation error e(t) vs. time index t for r=1 (Steady state performance).

K leads to better steady-state estimation performance for both D-PCA and S-PCA. Fig. 5.2 (bottom) depicts $e(t)$ for $r = 2$. In this case we test D-PCA and S-PCA for either $K = 5$, or $K = 15$ consensus iterations. Again $e(t)$ in adaptive D-PCA decreases at a faster rate than in the case of S-PCA. The subspace estimation performance improves considerably as the number of consensus (ADMM) iterations K increases. The steady state performance for $r = 1$ as seen over 10000 iterations is shown in Figure 5.3.

5.2.2 Post-SNR vs. Number of principal eigenvectors

Here, zero mean data is randomly generated for 2000 time instants for the 16 sensors and without observation noise.Parameter $c = 1$ in D-PCA and $\gamma = 10^{-3}$ for S-PCA, where inter-sensor communication noise is set such that the communication SNR is equal

to $\text{SNR}_{\text{comm}} = 20$ dB. Table 5.1 shows the post-SNR for different number r of estimated principal eigenvectors. It can be seen that the post-SNR decreases with decreasing number of principal eigenvectors estimated. This is because the lower the number of principal eigenvectors estimated, the lower the information content in the reduced dimension data.

Table 5.1. post-SNR vs. number of estimated principal eigenvectors

Scheme	D-PCA		
Number of Principal Components(r)	1	2	3
post-SNR (dB)	55.7302	64.5508	74.7502
pre-SNR (dB)	47.1883	46.9943	47.0229

5.3 D-PCA with bridge sensors vs. D-PCA without bridge sensors

So far, we have considered D-PCA with and without bridge sensors separately. Here, we show which has a faster convergence for different parameters. Here we evaluate the subspace estimation error by generating zero mean data for 2000 iterations for 16 sensors for both $r = 1, 2$ eigenvectors in both cases without any communication noise. We set the parameter $c = 4$ in both different algorithmic implementations. From Figure 5.4 and Figure 5.5, without bridges, D-PCA performs better in terms of convergence. However, this is only suitable for cases where a few sensors have more computational capacity than others, but not suitable for situations where all sensors have the same capabilities.

5.4 Denoising of Synthetic Data

Again, here we consider a sensor network with $p = 16$ sensors which are randomly placed in a unit square $[0, 1] \times [0, 1]$ with uniform distribution. The communication range of the network $d = 0.3$. Each sensor collects $t = 2000$ observations. The subspace projection

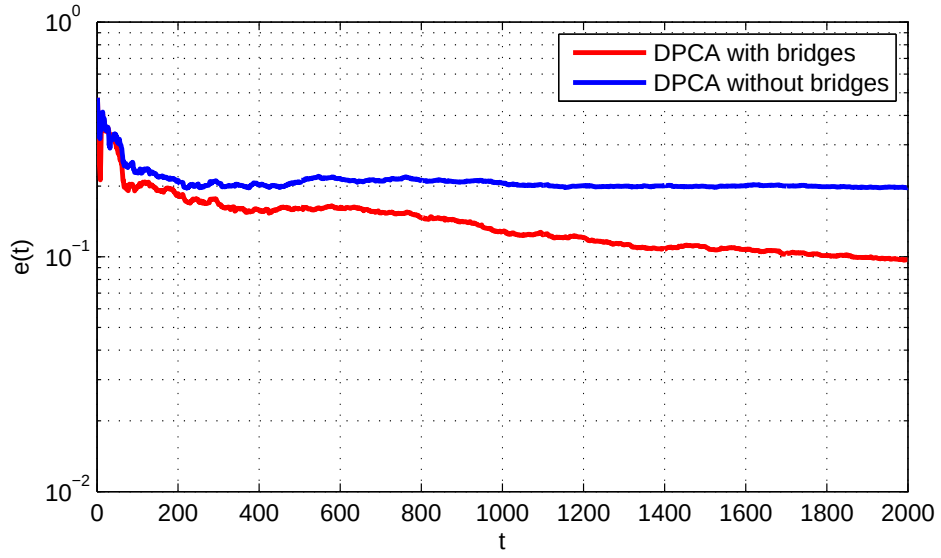

Figure 5.4. Subspace projection estimation error $e(t)$ vs. iteration index t for D-PCA with and without bridges for $r = 1$.

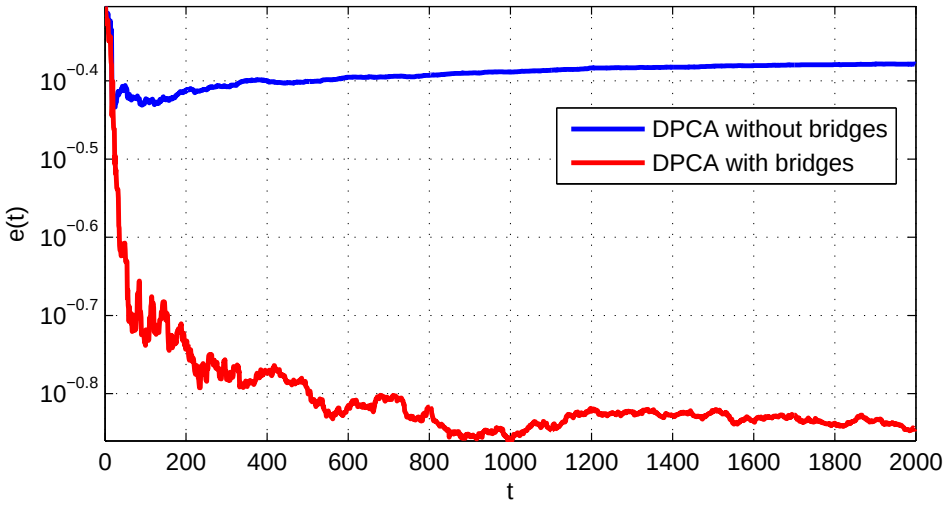

Figure 5.5. Subspace projection estimation error $e(t)$ vs. iteration index t for D-PCA with and without bridges for $r = 2$.

is $e(t) := \|\mathbf{C}^T(t)(\mathbf{C}(t)\mathbf{C}^T(t))^{-1}\mathbf{C}(t) - \mathbf{U}_{x,r}\mathbf{U}_{x,r}^T\|_F^2$. In this section, we examine the effects of the number of consensus iterations on principal subspace estimation in the presence of observation noise by estimating the principal subspace estimation error, evaluating the post-SNR and actual SNR (see Chapter 4 for definitions). Parameter $c = 1$ in D-PCA and

$\gamma = 10^{-3}$ for S-PCA. Zero mean data is randomly generated for 2000 time instants for the 16 sensors for each of the cases considered under this section.

5.4.0.1 post-SNR vs. number of consensus iterations

Here, the observation noise signal-to-noise ratio $SNR_{obs} = -3$dB and $SNR_{comm} = 25$dB while estimating $r = 3$ principal eigenvectors. Table 5.2 shows the comparison between post-SNR and the actual SNR (act-SNR) for different number of consensus iterations for both D-PCA and S-PCA. It is seen that in both cases, the post-SNR increases with the number of consensus iteration. This is because, with increasing number of ADMM iterations, sensors are able to communicate more with neighbors to reach consensus. Also, the SNR values are closer to the actual SNR for increasing number of consensus iterations. This is also true for D-PCA when compared with S-PCA. Further, D-PCA shows better performance on the average than S-PCA. We also calculate the difference between the act-SNR (which is the highest possible SNR that can be achieved) and the post-SNR, i.e., diff-SNR = act-SNR − post-SNR.

Table 5.2. post-SNR vs. Number of consensus iterations

Scheme	D-PCA			S-PCA		
Number of ADMM iterations (K)	5	10	15	5	10	15
post-SNR (dB)	50.0819	53.4347	54.5801	49.1154	50.7341	51.4089
act-SNR (dB)	58.2447	58.7176	58.4738	58.2447	58.7176	58.4738
diff-SNR (dB)	8.1627	5.2829	3.8938	9.1293	7.9835	7.0650

Figure 5.6 depicts the subspace estimation $e(t)$ error plotted against the iteration index t with the same parameters as above. This curves in this plot correspond to D-PCA for the different number of consensus iterations. As it can be observed, faster convergence is

observed for an increased number of consensus iterations K, and therefore a better steady state estimation performance is achieved. This testing is done for $K = 5, 15, 25$.

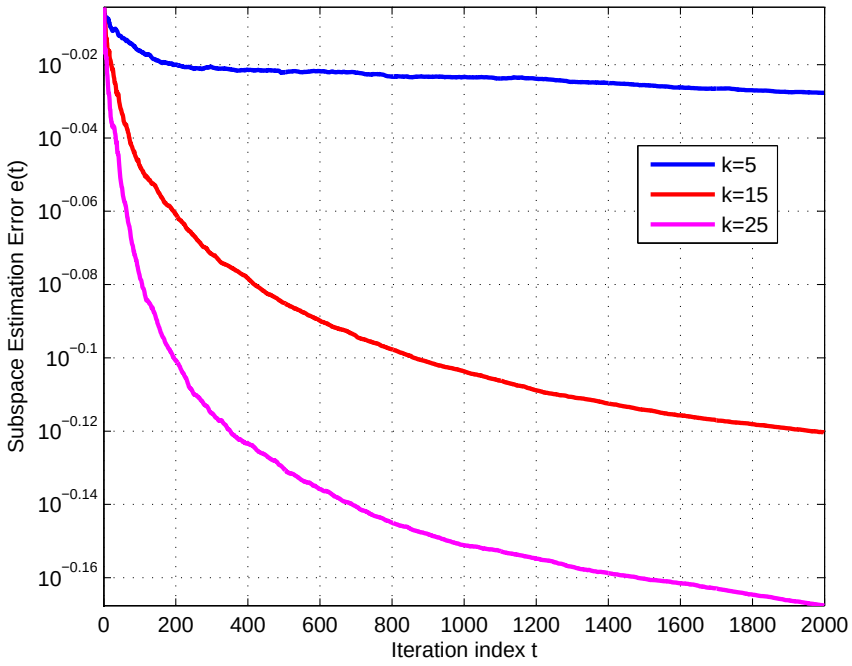

Figure 5.6. Subspace projection estimation error $e(t)$ vs. iteration index t for different number of consensus iterations for $\text{SNR}_{obs} = -3$dB and $\text{SNR}_{comm} = 25$dB.

5.4.0.2 post-SNR vs. number of estimated principal eigenvectors

Here, the post-SNR is estimated for different number of estimated principal eigenvectors r. The number of ADMM iterations is fixed to $K = 25$, while the observation SNR is $\text{SNR}_{obs} = 20$dB and $\text{SNR}_{comm} = 10$dB.

As seen in table 5.3, higher post-SNR values were obtained for an increasing number of principal eigenvectors being estimated, and overall, the SNR values for D-PCA are significantly larger that than the post-SNR values achieved by S-PCA in [38].

Table 5.3. post-SNR vs. Number of principal eigenvectors

Scheme	D-PCA			S-PCA		
Number of principal eigenvectors(r)	1	2	3	1	2	3
post-SNR (dB)	52.8956	60.3031	81.1464	52.3053	59.7009	74.8399
act-SNR (dB)	56.4966	68.4716	110.7478	56.4966	68.4716	110.7478

Table 5.4. post-SNR vs. communication noise SNR

Scheme	D-PCA	S-PCA	D-PCA	S-PCA
SNR(comm) (dB)	32db		Inf	
act-SNR (dB)	111.0		111.5	
post-SNR (dB)	81.7348	69.9509	85.4463	71.7930

5.4.0.3 post-SNR vs. communication noise SNR

The post-SNR is estimated for two scenaria where i) there is inter-sensor communication noise; and ii) the inter-sensors links are ideal and there is no noise. Again, we set $r = 3$ and $K = 25$ ADMM iterations, while the observation SNR is $\text{SNR}_{obs} = -3$dB. Figure 5.7 illustrates the fact that the convergence is slowed by the presence of communication noise, and D-PCA still outperforms S-PCA with or without observation noise. This is further substantiated by Table 5.4 where the Post SNRs without communication noise were larger than with it. A comparison was also made for different values of communication noise in the presence of observation noise for $r = 3$ and $K = 25$ consensus iterations and with $\text{SNR}_{obs} = 5$dB as shown in Table 5.8. D-PCA still outperforms the existing alternative for all three different values.

5.5 Denoising of Real Data

Here we test the denoising capability of D-PCA and compare it with existing alternatives on real ocean temperature data. The temperature data where obtained by the so called Argo

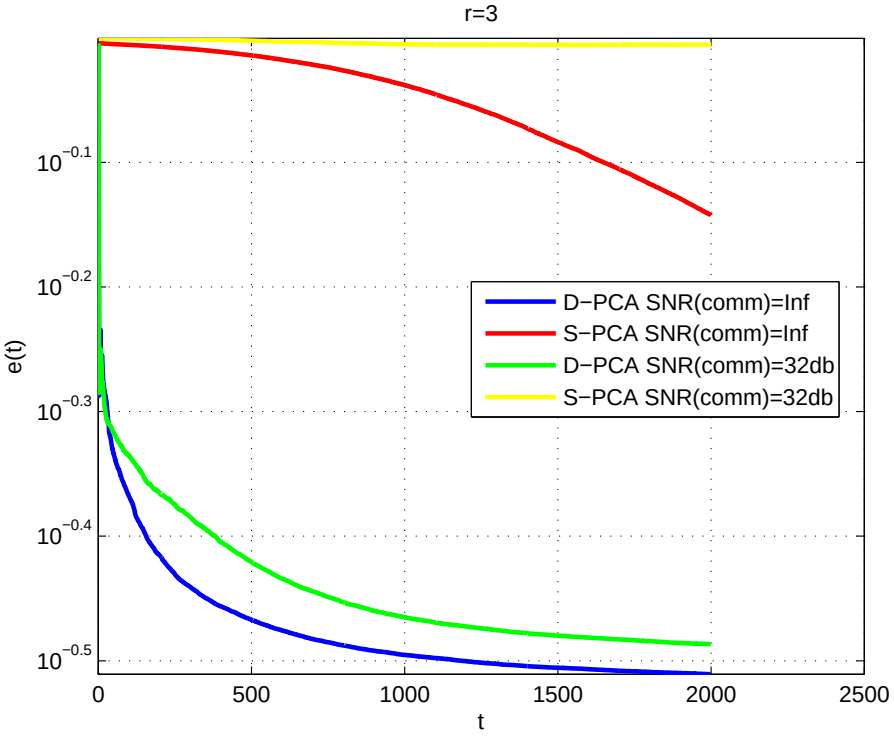

Figure 5.7. Subspace projection estimation error $e(t)$ vs. iteration index t with and without communication noise for $\text{SNR}_{obs} = -3db$.

Floats that have been deployed in all oceans. The data are obtained and handled by the National Oceanographic Data center (NODC), and we downloaded them from their online data repository [47]. These measurements are used to compare the performance of D-PCA and S-PCA, under different scenaria as detailed next.

Specifically, data measurements acquired across 20 sensors ($p = 20$) over $t = 117$ time stamps during the time period from $10/2/1998$ to $7/10/2002$. Even though the measurements are taken over an extensive time period, it has sufficient correlation to be employed in the denoising algorithm. The subspace projection $e(t) := \|\mathbf{C}^T(t)(\mathbf{C}(t)\mathbf{C}^T(t))^{-1}\mathbf{C}(t) - \mathbf{U}_{x,r}\mathbf{U}_{x,r}^T\|_F^2$ and post-SNR will be used as performance evaluation criteria. We still con-

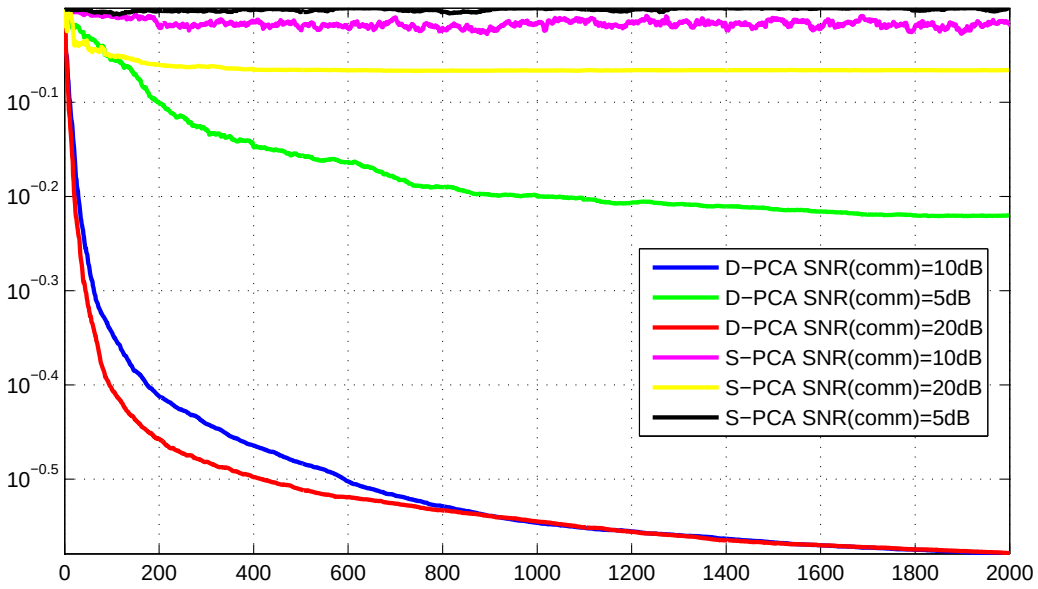

Figure 5.8. Subspace projection estimation error $e(t)$ vs. iteration index t for $\text{SNR}_{comm} = 5db, 10db, 20db$ with $\text{SNR}_{obs} = 5db$.

sider the presence of observation noise since this illustrates an actual sensor network setting. Parameter $c = 1$ in D-PCA and $\gamma = 10^{-4}$ for S-PCA.

5.5.1 pre-SNR vs. actual SNR vs. post-SNR

A comparison is made here between the pre-SNR, the actual SNR and the post-SNR for both D-PCA and S-PCA for $K = 25$ ADMM iterations while estimating $r = 3$ principal eigenvectors. Further, the observation SNR is set to $\text{SNR}_{obs} = 41$dB and the inter-sensor link variance is $\text{SNR}_{comm} = 10$ dB. As it can be seen in Table 5.5, for a fixed pre-SNR and act-SNR pair, the post-SNR for D-PCA is higher and closer to the actual SNR (act-SNR).

Table 5.5. pre-SNR, actual SNR (act-SNR) and post-SNR using real data

Scheme	D-PCA	S-PCA
pre-SNR (dB)	41.4080	
act-SNR (dB)	58.4738	
post-SNR (dB)	54.5801	51.4089

Table 5.6. post-SNR vs. number of estimated principal eigenvectors using real data

Scheme	D-PCA			S-PCA		
Number of principal eigenvectors(r)	1	2	3	1	2	3
post-SNR (dB)	18.6945	26.7986	28.5122	15.6597	17.1463	18.7691
act-SNR (dB)	21.0085	28.8255	37.4376	21.0085	28.8255	37.4376

5.5.2 post-SNR vs. number of estimated principal components

Here, the post-SNR is estimated for different number of estimated principal eigenvectors r after applying $K = 25$ ADMM iterations per time instant t where $\text{SNR}_{obs} = 44$dB and $\text{SNR}_{comm} = 10$ dB.

Similar to Table 5.1, Table 5.6 compares post-SNR for both D-PCA and S-PCA. Again, D-PCA shows a better performance in terms of SNR for the different number of principal eigenvector being estimated. A similar variation is observed as with the synthetic data, which further substantiates our claim of better performance of D-PCA as compared to S-PCA.

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

Two distributed dimensionality reduction techniques applicable to different settings were developed in this research work/thesis. The first involved sensor networks and is preferred for scenarios where a few sensors have a higher computational capacity than others, in which case those sensors (called bridge sensors) perform most of the computation and communication at the expense of the simple sensors. The second involved scenarios where all sensors have equal computational capabilities. The computational and communication costs were studied for both algorithms. In both cases, the approaches involved a formulation of the PCA cost as a separable constrained optimization problem and application of the coordinate descent technique combined with ADMM to approach the separable PCA formulation in a distributed manner. Only single-hop communications were required and no fusion center, as in traditional dimensionality reduction approaches is necessary. The involved computational and communication complexity is small and manageable due to the single-hop communications. Inter-sensor communication noise are also considered and extensive numerical testing corroborated the robustness of the proposed framework while existing alternatives fail to converge. Further, it is demonstrated that the convergence rate of the proposed distributed PCA framework is faster than related distributed approaches while having a comparable communication cost. The proposed algorithmic framework is then applied in a distributed denoising application, where the goal is to remove sensing noise from the sensor data and improve the signal-to-noise ratio in the processed (denoised) data. This is done by projecting the sensor measurement data onto an estimate of the data covariance principal eigenspace obtained via our distributed PCA algorithm. Different principal

eigenspace estimates obtained by alternative distributed methods are used as well, and it turns out that the distributed PCA framework put forth here leads to a substantial noise reduction in the data compared to existing alternatives. Both synthetic data and real data corresponding to ocean temperature measurements acquired from sensors are utilized to demonstrate the advantages of the novel algorithms existing techniques via extensive numerical tests.

This thesis work laid a foundation of a distributed PCA framework with applications in data dimensionality reduction and denoising. Future directions include:

- Convergence analysis of the proposed algorithmic framework.
- Extension of the algorithms for tackling nonlinear models in denoising and dimensionality reduction.
- Consideration of time-varying principal eigenspaces in nonstationary environments, e.g., mobile field sources.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] M. Ilyas, I. Mahgoub, and L. Kelly, "Handbook of sensor networks: compact wireless and wired sensing systems. 2005."

[3] Y.-A. Le Borgne, S. Raybaud, and G. Bontempi, "Distributed principal component analysis for wireless sensor networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008.

[4] T. Hastie, R. Tibshirani, and D. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Second Edition, Springer, 2009.

[5] D. R. Brillinger, *Time Series: Data Analysis and Theory*. Expanded Edition, Holden Day, 1981.

[6] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. on Sig. Processing*, vol. 43, no. 1, pp. 95–107, 1995.

[7] A. Bharathidasan and V. A. S. Ponduru, "Sensor networks: An overview."

[8] F. L. Lewis, "Wireless sensor networks."

[9] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 2033–2036.

[10] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. ACM, 1999, pp. 263–270.

[11] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Distributed detection and estimation in wireless sensor networks," *arXiv preprint arXiv:1307.1448*, 2013.

[12] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[13] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[14] S. V. Macua, P. Belanovic, and S. Zazo, "Consensus-based distributed principal component analysis in wireless sensor networks," in *Proc. of 11th IEEE Workshop on Sig. Proc. Advances in Wir. Com. (SPAWC)*, June 2010, pp. 1 –5.

[15] M. Sewell, "Principal component analysis."

[16] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[17] A. Gifi, *Nonlinear multivariate analysis*. Wiley Chichester, 1990.

[18] K. I. Diamantaras and S. Y. Kung, *Principal component neural networks*. Wiley New York, 1996.

[19] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Applicat.*, vol. 106, no. 1, pp. 69–84, 1985.

[20] Z. jian Bai, R. H. Chan, and F. T. Luk, "Principal component analysis for distributed data sets with updating," in *In Proceedings of International workshop on Advanced Parallel Processing Technologies (APPT*, 2005.

[21] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson, "Distributed clustering using collective principal component analysis," *Knowledge and Information Systems*, vol. 3, p. 2001, 1999.

[22] H. Qi, T. wei Wang, and J. D. Birdwell, "Global principal component analysis for dimensionality reduction in distributed data mining," 2004.

[23] Y. L. Borgne, S. Raybaud, and G. Bontempi, "Distributed principal component analysis for wireless sensor networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008.

[24] M. Gastpar, P. L. Dragotti, and M. Vetterli, "The distributed karhunen–loeve transform," *Information Theory, IEEE Transactions on*, vol. 52, no. 12, pp. 5177–5196, 2006.

[25] Z. Meng, A. Wiesel, and A. O. Hero, "Distributed principal component analysis on networks via directed graphical models," in *in Proc. of IEEE Intl. Conf. on Acoust., Speech and Sig. Proc*, March 2012, pp. 2877 –2880.

[26] L. Li, A. Scaglione, and J. H. Manton, "Distributed principal subspace estimation in wireless sensors networks," *IEEE Journal of Sel. Topics in Sig. Proc.*, vol. 5, no. 4, pp. 725–738, 2011.

[27] A. Bertrand and M. Moonen, "Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA," *Internal Report KU Leuven ESAT-SCD*, 2013.

[28] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.

[29] H. Bauer, *Probability Theory*, ser. De Gruyter studies in mathematics. Bod Third Party Titles, 1996.

[30] I. D. Schizas, G. B. Giannakis, and Z.-Q. Luo, "Optimal dimensionality reduction for multi-sensor fusion in the presence of fading and noise," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing*, Toulouse, France, May 2006, pp. 869–872.

[31] ——, "Distributed estimation using reduced dimensionality sensor observations," in *Proc. of 39th Asilomar Cof. On Signals, Systems and Computers*, Monterey, CA, Oct. 2005, pp. 1029–1033.

[32] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Dimensionality reduction, compression and quantization for distributed estimation with wireless sensor networks." pp. 259–296, Springer, New York, 2006: in Wireless Communications (P. Agrawal, D. M. Andrews, P. J. Fleming, G. Yin, and L. Zhang, eds.), vol. 143 of IMA Volumes in Mathematics and its Applications.

[33] ——, "Consensus in ad hoc wsns with noisy links- part i: Distributed estimation of deterministic signals," *IEEE Trans. on Sig. Processing*, vol. 56, no. 1, pp. 350–364, 2008.

[34] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. 2nd ed. Belmont, MA: Athena Scientic, 1999.

[35] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[36] D. P. Bertsekas, *Nonlinear Programming*. Second Edition, Athena Scientific, 2003.

[37] A. Aduroja, I. D. Schizas, and V. Maroulas, "Distributed principal components analysis in sensor networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5850–5854.

[38] "Distributed principal subspace estimation in wireless sensor networks."

[39] S. M. Kay, *Fundamental of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1993.

[40] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, Sept. 2004.

[41] S. J. Schiff, J. G. Milton, J. Heller, and S. L. Weinstein, "Wavelet transforms and surrogate data for electroencephalographic spike and seizure localization," *Optical Engineering*, vol. 33, no. 7, pp. 2162–2169, 1994.

[42] D. L. Donoho, "De-noising by soft-thresholding," *Information Theory, IEEE Transactions on*, vol. 41, no. 3, pp. 613–627, 1995.

[43] O. Bertrand, J. Bohorquez, and J. Pernier, "Time-frequency digital filtering based on an invertible wavelet transform: an application to evoked potentials," *Biomedical Engineering, IEEE Transactions on*, vol. 41, no. 1, pp. 77–88, 1994.

[44] A. Effern, K. Lehnertz, T. Schreiber, T. Grunwald, P. David, and C. Elger, "Nonlinear denoising of transient signals with application to event-related potentials," *Physica D: Nonlinear Phenomena*, vol. 140, no. 3, pp. 257–266, 2000.

[45] M. Ghil, M. Allen, M. Dettinger, K. Ide, D. Kondrashov, M. Mann, A. W. Robertson, A. Saunders, Y. Tian, F. Varadi, *et al.*, "Advanced spectral methods for climatic time series," *Reviews of Geophysics*, vol. 40, no. 1, p. 1003, 2002.

[46] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 530435, 14 pages, 2009. doi:10.1155/2009/530435.

[47] "Operational oceanography group:global argo data repository.april 2007," U.S. Department of Commerce, National Oceanic and Atmospheric Administration, National Oceanographic Data Center, Silver Spring, Maryland, 20910.October 12,2013. [Online]. Available: http://www.nodc.noaa.gov/argo

BIOGRAPHICAL STATEMENT

Abiodun T. Aduroja was born in Osun State, Nigeria in 1988. He received his B.S. degree in Electrical and Electronic Engineering from Ladoke Akintola University of Technology in 2009, his M.S. degree from The University of Texas at Arlington in 2013 in Electrical Engineering. From 2009 to 2010, he worked as an Electrical Engineer at Tralat Engineering and Energy Solutions. From August 2012 to May 2013, he worked as a Field Test Specialist at Blackberry as an intern. He is a member of the IEEE.