Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Dottorato di Ingegneria dell'Informazione
Indirizzo in Scienza e Tecnologia dell'Informazione
XXII Ciclo

Tesi di Dottorato / Ph.D. Thesis

Analysis and Development of Consensus–based Estimation Schemes

Ph.D. Advisor: Professor Sandro Zampieri

School Coordinator: Professor Matteo Bertocco

Ph.D. Candidate: Simone Del Favero

Padova, January, 2010

Analysis and Development of Consensus–based Estimation Schemes

Simone Del Favero

Advisor: Professor Sandro Zampieri

Abstract

In the last few decades we assisted to an extraordinary expansion of the Internet and of wireless technologies. These interconnection technologies allow a continuously increasing number of devices to exchange information. This fact, together with the parallel increase in the availability of inexpensive nodes carrying a wide range of sensing capabilities, attracts the interest in developing large-scale sensing platforms, which could be used to measure a variety of physical phenomena.

However, these huge networks of simple devices are subject to tight energy and bandwidth constraints, making efficient distributed estimation and data fusion algorithms a strong need, to avoid unmanageable computational and communicational burden on network bottleneck nodes.

In this thesis we address some issues in this field, presenting and analyzing distributed algorithms to solve specific distributed estimation problems and carrying out the analysis of some other recently-proposed algorithms.

To perform data fusion in a distributed fashion, we relay on consensus algorithms, namely algorithms that achieve agreement on a common value in the network. Using consensus as a basic brick to build estimation algorithms we can take advantage of the solid understanding on this problem that many recent contributions deepened and sharpened, and we can leverage for our analysis on powerful and effective tools.

In the thesis we propose a distributed algorithm for offset removal and an algorithm for least-square identification of the wireless-channel parameters, motivated by the application of localization and tracking of a moving object. We present moreover a novel linear algebra inequality, useful in the analysis of randomized algorithms. This result comes into play when we carry out an analysis of a recently-proposed distributed Kalman filtering algorithm. Finally, we look at the intriguing set up of a network cooperation to estimate different but correlated quantities, proposing and analyzing a distributed algorithm that performs inference over a simple Gauss-Markov random field.

Keywords: Distributed Estimation, Sensor Networks, Wireless Sensor Networks, Applications of Consensus, Distributed Kalman Filter, Distributed Kalman Smoothing, Localization and Tracking.

Analysis and Development of Consensus–based Estimation Schemes

Simone Del Favero

Supervisore: Professor Sandro Zampieri

Sommario

Gli ultimi decenni sono stati segnati dallo straordinario sviluppo di Internet e dalla pervasiva diffusione della tecnologia wireless, consentendo ad un numero sempre maggiore di dispositivi di scambiare tra loro informazioni. Questo fatto, assieme alla crescente disponibilità, a prezzi modici, di nodi equipaggiati con un'ampia varietà di dispositivi di misura, rende tecnologicamente concretizzabile l'idea di sviluppare grandi piattaforme di sensing, incaricate di monitorare qualsivoglia grandezza fisica.

Tuttavia, queste grandi reti di dispositivi estremamente semplici hanno stringenti vincoli sul consumo energetico e sulla banda di comunicazione, che rendono criticamente necessario lo sviluppo di tecniche efficienti per la stima e la data-fusion, così da evitare carichi computazionali e di comunicazione insostenibili ai colli di bottiglia della rete.

Questa tesi si propone di contribuire proprio in questo settore, presentando alcuni algoritmi per la soluzione distribuita di specifici problemi di stima ed analizzando le prestazioni di algoritmi recentemente proposti.

Strumento chiave nella decentralizzazione della stima è la teoria del consensus, che propone algoritmi in grado di portare l'intera rete a concordare su una specifica quantità. L'utilizzo di algoritmi di consensus come elemento base nella costruzione di algoritmi di stima ci consente di sfruttare la solida comprensione di questo problema, affinata dai molti risultati recentemente proposti in letteratura, e di sfruttare degli strumenti di analisi ben consolidati.

Nella tesi, motivati dal problema della localizzazione e del tracking di un oggetto, proponiamo un algoritmo per la compensazione degli offset ed un algoritmo per la stima ai minimi quadrati dei parametri caratterizzanti il canale wireless. Inoltre presentiamo un nuovo risultato di algebra lineare, utile nell'analisi di algoritmi randomizzati. Questo risultato giocherà un ruolo centrale nell'analisi qui proposta di un algoritmo distribuito per la stima alla Kalman. Infine, consideriamo l'interessante caso di una rete di sensori incaricata di stimare quantità diverse ma tra loro correlate e proponiamo un algoritmo per l'inferenza di un semplice campo di Gauss-Markov.

Parole chiave: Stima Distribuita, Reti di Sensori, Reti di Sensori Wireless, Applicazioni del Consensus, Filtro di Kalman Distribuito, Interpolatore di Kalman Distribuito, Localizzazione e Tracking

Analysis and Development of Consensus–based Estimation Schemes

Simone Del Favero

Contents

Introduction

In the last few decades we assisted to an extraordinary expansion of the Internet and of wireless technologies. These interconnection technologies allow a continuously increasing number of devices to exchange information. This fact, together with the parallel increase in the availability of inexpensive nodes carrying a wide range of sensing capabilities, enforces the interest in developing large-scale sensing platforms, which could be used to measure a variety of physical phenomena. However, these huge networks of simple devices have tight energy and bandwidth constraints, making efficient distributed estimation and data fusion algorithms a strong need to avoid unmanageable computational and communicational burden on network bottleneck nodes.

In particular, wireless sensor networks (WSNs), i.e. networks of smart devices that can sense, compute and exchange information with their neighbors, are becoming very popular because of their promise to revolutionize many engineering areas involving monitoring and control [1]. The strength of WSNs resides in their flexibility and scalability, since the same hardware and software can be rapidly reconfigured and adapted to manage rather different applications, from ambient monitoring to people tracking, from industrial control to energy management in buildings. However, many challenges ranging from hardware design, to real-time middleware prototyping, from data routing protocols to distributed signal processing still remain to be solved before WSNs can become really ubiquitous and successful.

In this thesis we address some problems in distributed estimation and carry out the analysis of some recently proposed algorithms. A detailed overview of this dissertation and a summary of its contribution can be found in the next section.

1 Overview of the dissertation

This thesis is organized as follows.

- **Chapter 1**. In this chapter we briefly review some material on *graph theory* and *consensus theory* that will be broadly used in this thesis. In particular, we summarize some convergence results both for the time-invariant and time-varying case, with special attention to the important class of randomized algorithms. Probabilistic convergence results are presented and two important examples, namely symmetric gossip and broadcast, are discussed and compared.

- **Chapter 2**. In this chapter we consider the problem of computing global quantities, i.e. quantities that are function of all the data collected in the network. We report a result that characterizes a class of global functions that can be computed by means of consensus algorithms and we show that many problems of interest fit this class. Consensus algorithms have therefore to be looked as one of the basic tools for distributed estimation. In particular, we show how to cast into a consensus problem the problem of computing the least square estimate of data collected by the nodes. We will apply this algorithm in Chapter 3 to estimate in a distributed manner the wireless channel parameters.

- **Chapter 3**. In this chapter we focus on one of the most important applications of wireless sensor networks: localization and tracking. In particular, we study two problems arising when localization is based on the strength of the radio signal received. Specifically, we first propose a distributed strategy to minimize the effects of unknown constant offsets in the reading of the Radio Strength Signal Indicator (RSSI), due to uncalibrated sensors. Then, we consider the problem of estimating the channel parameters for a generic wireless sensor network in a distributed manner. To do so, we formulate the estimation problem as global least-square problem, that we solve using the distributed algorithm presented in Chapter 2. The proposed algorithms do not require any knowledge on the global topology of the network nor on the total number of nodes. Finally, we apply these algorithms to experimental data collected from an indoor wireless sensor network.

  The material presented in this chapter is the result of the joint work with Saverio Bolognani, Damiano Varagnolo and Luca Schenato. These results are presented also in [2] and in [3].

- **Chapter 4**. In the analysis of a recently proposed distributed estimation algorithm based on the Kalman filtering and on gossip iterations, that will be object of Chapter 5, we developed a new inequality which is valid for i.i.d. matrix valued random processes. This inequality is a general linear algebra result that can be useful in the analysis of the convergence rate of general jump-Markov linear systems.

  In this chapter we present this inequality, based on the theory of majorization and on its use in the analysis of the singular values.

  The material presented in this chapter is the result of the joint work with Sandro Zampieri and they are present also in [4] and in [5].

- **Chapter 5**. In this chapter we consider the problem of estimating a random process from noisy measurements, collected by a sensor network. We analyze a distributed two–stage algorithm. The first stage is a Kalman–like estimate update, in which each agent makes use only of its own measurements. During the second phase agents communicate with their neighbors to improve their estimate. Estimates fusion is operated by running a consensus iteration. In literature it has been considered only the case of fixed communication strategies, i.e. described by a fixed constant consensus matrix. However, in many practical cases this is just a rough model of communication in a sensor network, that usually happens according to a randomized strategy. This strategy is more properly modeled by assuming that the consensus matrices are drawn, according to a selection probability, from an alphabet of matrices compatible with the communication graph, at each time instant. This chapter deals therefore with randomized communication strategies and in particular with the symmetric gossip. A worst–case and a mean–square performance analysis is carried out and an upper–bound for the estimation-error variance is derived, based on the result presented in Chapter 4. This upper–bound is a good performance assessment index and therefore it is assumed as a cost function to be minimized. Moreover we show that the problem of minimizing this cost function by choosing the Kalman gain and the selection probability is convex in each of the two variables separately although it is not jointly convex. Finally simulations are presented and the results discussed.

  The material presented in this chapter is the result of the joint work with Sandro Zampieri and they are present also in [4] and partially covered in [5].

- **Chapter 6**. In this chapter we consider the set up of a network of nodes cooperating to estimate different but correlated quantities and

we assume these quantities to be constant in time. We present a cooperative smoothing algorithm for Gauss-Markov linear models whose aim is to reconstruct the overall state sequence in a cooperative way, by taking advantage of all the data obtained by the network.

A convergence analysis is carried out, fully characterizing the gap between distributed and optimal estimate. This points out the importance, in the algorithm design, of finding the right trade-off between parallelism and rate of convergence toward the optimal estimate. In the simple yet significant case of a random walk, this issue is further investigated. The convergence rate has been studied as a function of J and we derived a simple approximation of it for large values of J, suggesting that the it increases exponentially with J.

The material presented in this chapter is the result of the joint work with Gianluigi Pillonetto and Bradley M. Bell and it represents an extension of [6]. We present here some recent results of our joint research on this topic, reported also in [7].

2 Notation

We will denote with $\mathbb{N}$ the set of the integer numbers with $\mathbb{R}$ the set of real numbers and with $\mathbb{C}$ the set of complex numbers.

Vectors are meant to be column vectors. We will denote with x^T the transposed of x and with x^* its conjugate transposed.

$\mathbb{1}_N$ represents the vector $[1, \dots, 1]^T \in \mathbb{R}^N$. If the dimension N is clear from the context, we will simply write $\mathbb{1}$. e_i denotes the vector of $\mathbb{R}^N$ having i-th entries equal to 1 and all the other entries equal to 0. Hence $e_1 = [1, 0, \dots, 0] \in \mathbb{R}^N$.

Given a square matrix $M \in \mathbb{R}^{N \times N}$, $\mathrm{tr}(M)$ will denote the trace of the matrix M and $\mathrm{vect}(M)$ it vectorization.

Given a matrix $M \in \mathbb{R}^{N \times N}$ we will denote with $\underline{\sigma}(M)$ the vector in $\mathbb{R}^N$ formed by the singular values of M decreasingly ordered and with $\underline{\lambda}(M)$ the vector in $\mathbb{C}^N$ formed by the eigenvalues of M ordered so that $|\underline{\lambda}_1(M)| \geq \cdots \geq |\underline{\lambda}_N(M)|$, where each eigenvalue appears as many times as its algebraic multiplicity. Recall moreover that for any normal matrix N, $NN^T = N^T N$, $\underline{\sigma}(N) \equiv |\underline{\lambda}(N)|$, where with $|\underline{\lambda}(N)|$ we mean the $\mathbb{R}^n$ vector having entries $|\lambda_i(N)|$.

Through the thesis, for ease of notation, we will write $\sigma_j^i(M)$ to denote $(\sigma_j(M))^i$. Note that, using the notation we just defined, $\lambda_j(M^i) = \lambda_j^i(M)$ and, therefore, for a normal matrix N, $\sigma_j(N^i) = \sigma_j^i(N)$.

Moreover we will denote with $\otimes$ the Kronecker product of two matrices while $\odot$ will represent the Hadamard product, i.e. the entry–wise product.

Recall that a stochastic matrix P is a matrix with non–negative entries such that $P\mathbb{1} = \mathbb{1}$. A matrix P is said to be doubly stochastic if both P and P^T are stochastic. Any symmetric stochastic matrix is therefore doubly stochastic.

We will denote with $\mathbb{S}$ the set of symmetric matrices.

We will denote with $\mathbb{P}$ a probability measure over a measurable space and with $\mathbb{E}$ the expected value of a random variable. Given a random vector x, taking values in $\mathbb{C}^N$, $\mathbb{V}(x)$ will represent its variance, $\mathbb{V}(x) = \mathbb{E}\left[(x - \mathbb{E}[x])(x - \mathbb{E}[x])^*\right]$ while $\mathbb{V}(xy)$ denotes the covariance between two random vectors $\mathbb{V}(xy) = \mathbb{E}\left[(y - \mathbb{E}[y])(x - \mathbb{E}[x])^*\right]$.

$x \sim \mathcal{N}(\mu, \Sigma)$ will denote a Gaussian random vector with mean μ and variance Σ.

Finally, we will denote a graph as $\mathcal{G}$, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the node set and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the edge set. For further notation related to graphs we refer to section 1.1.

$\mathbb{N}$	set of natural numbers
$\mathbb{R}$	set of real numbers
$\mathbb{C}$	set of complex numbers
x^T	transposed of x
x^*	conjugate transposed of x
x_i	i-th entry of the vector x
$\mathbb{1}_N, \mathbb{1}$	$\mathbb{R}^N$ vector having all entries equal to 1
e_i	$\mathbb{R}^N$ vector having the i-th entry 1 and the other entries 0
M_{ij}, $M_{i,j}$, $[M]_{ij}$	(i,j) entry of the matrix M
$\mathrm{tr}(M)$	trace of M
$\mathrm{vect}(M)$	vectorization of M
$\sigma(M)$, $\underline{\sigma}(M)$	$\mathbb{R}^N$ vector of the singular values of M, ordered decreasingly
$\lambda(M)$, $\underline{\lambda}(M)$	$\mathbb{C}^N$ vector of the eigenvalues of M, ordered by decreasing modulus
$\otimes$	Kronecker product
$\odot$	Hadamard product (entry-wise product)
$\mathbb{P}$	probability measure
$\mathbb{E}$	expected value
$\mathbb{V}(xy)$	covariance matrix of x, y
$\mathbb{V}(x)$	variance matrix x
$\mathcal{G}$	graph
$\mathcal{V}$	node set
$\mathcal{E}$	edge set
$\mathcal{V}(i)$	i's neighbor nodes set
$d(i)$	degree of node i

Table 1: List of Symbols

1

Networks Modeling and Consensus Theory: a brief review

In this chapter we briefly review some material on *graph theory* and *consensus theory* that will be broadly used in this thesis. In fact, the most natural way to model a network is by mean of a graph and the achievement of average consensus, namely to compute the arithmetic mean of local quantities, is one of the simplest distributed estimation problems. Indeed, consensus is much more than a data fusion toy problem: as we will see throughout this thesis and in particular in Chapter 2, consensus is the basic brick used to build much more complex distributed estimation algorithms.

Convergence results for consensus algorithms are summarized in this chapter, both for the time invariant and time-varying case. Moreover we will introduce an important class of time-varying algorithms: the randomized ones. For this class we present some probabilistic convergence results and two important examples, namely symmetric gossip and broadcast, are discussed and compared.

1.1 Networks Modeling

The most natural way to model a network is by means of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The agents of the network are modeled as nodes of the graph, namely as elements of the node set $\mathcal{V} = \{1 \dots N\}$, where N is the number of agents in the network. The edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ models the communication constraints of the network: if node i can send informations to node j than $(i, j) \in \mathcal{E}$. Through all this thesis we will assume moreover that all the self–loops (i, i) belong to the edge set, $(i, i) \in \mathcal{E}\ \forall i \in \mathcal{V}$, since in all the application of interest node i can access to the information that it stores.

If $(j,i) \in \mathcal{E}$, $i \neq j$ than j, is said to be *neighbor* of i and we denote with $\mathcal{V}(i)$ the set of neighbors of node i

$$\mathcal{V}(i) = \{j \,|\, (j,i) \in \mathcal{E}, i \neq j\}$$

and we call *in-degree*, or simply *degree*, of i

$$d(i) = |\mathcal{V}(i)|$$

the cardinality of this set, namely the amount of nodes that can send information from i.
Moreover, we call *out-degree* of i

$$d_{\text{out}}(i) = |\{j \,|\, (i,j) \in \mathcal{E}, i \neq j\}|,$$

that is the amount of nodes that can receive information from i.

Such a mathematical formulation is well suited to describe a broad variety of networks and to effectively model many different applications of interest. Nodes may represent sensors of a sensor network, computers in a Local Area Network or in the Internet. The network may consist of mobile robots that have to be coordinated or of computers running a peer to peer application. As well, communications that happen between agents may be very different: they might be based on a wired link between two nodes or rely on wireless transmissions. Links might be very reliable or be subject to interference and failure. Moreover, the fact that i can send informations to node j does not imply that also j can send informations to node i. If this happens the link is said to be bidirectional.

Furthermore the graph describing a network might vary in time, $\mathcal{G}(t)$. New communication links might be created or break down. Think for instance to a swarm of mobile robots that can communicate only when sufficiently close each other. As their relative positions change the communication graph changes consequently, leading to great challenges in their coordination. Link failure is an event that can not be considered rare in sensor networks. It might, for instance, happen if an obstacle interposes between two wireless nodes. Nodes themselves may fail and new nodes may join to the network, as for instance in a peer to peer network where nodes commonly join and leave the network or in a sensor network of battery powered sensors, where a node may permanently fail since it runs out of battery.

Even when nodes and links do not permanently fail, communications trough a link may occasionally fail. This fact might be not negligible for estimation and control purposes, especially when dealing with wireless transmissions that, due to frequent packet losses and interferences, are rather

unreliable. To take into account this uncertainty in the communication a typical approach is to associate to every link (i, j) the probability $c_{i,j}$ that a transmission is successfully carried out through that link. The probabilities c_{ij} are then collected in the *connectivity matrix* $C \in \mathbb{R}^{N \times N}$, where $[C]_{ij} = c_{ij}$. One can then define the *c-connectivity graph* $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ associated to the connectivity matrix C as the graph such that (i, j) belongs to the edge set $\mathcal{E}_c$ if and only if $c_{ij} \geq c$. Therefore, in the c-connectivity graph only the reliable links are kept. The matrix C can be experimentally estimated sending on each link M messages, recording the number m_{ij} of messages successfully received and setting $\hat{c}_{ij} = \frac{m_{ij}}{M}$.

Since this formalism is particularly useful in the case of wireless transmissions, it is worth to note that, in this case some peculiarity of the wireless channel can be exploited to improve the previously mentioned procedure. In fact, since the wireless channel is approximately symmetric we can assume that $C = C^T$. The matrix C can be experimentally estimated by letting each node broadcast M packets at random instants (with retransmission intervals sufficiently large in order to avoid collisions), making each node i record the number m_{ij} of messages received by each node j and setting $\hat{c}_{ij} = \frac{m_{ij}}{M}$. Subsequently, each node communicate its $\hat{c}_{ij}$ to its neighbors and sets $c_{ij} = \frac{\hat{c}_{ij} + \hat{c}_{ji}}{2}$ since $\hat{c}_{ij}$ and $\hat{c}_{ji}$ are different being empirical means.

It is worth, moreover, to remark that the communication graph $\mathcal{G}$ that we consider might be different from the graph of the communications feasible at the physical layer. In fact, even if two nodes i and j are not connected by means of a direct physical link, an appropriate routing protocol may allow information to flow between them. If the sequence of communications that allows information to flow between i and j is sufficiently fast with respect to the other operations sensors have to do, we might consider i connected with j and add the link (i, j) to the set $\mathcal{E}$.

Different nodes and links characteristics lead to different graph properties. For instance bidirectional communication links lead to undirected graph[1]. In the next paragraph we recall the definition of some further graph properties that will be of interest in the following.

Review of some graph properties

A graph is said to be a *tree rooted at node* $k \in \mathcal{V}$ if the is always a path connecting k to any other node j, $j \in \mathcal{V}$, $j \neq k$ and there are no cycles.

A graph is *rooted in* k if there exists a tree embedded in the graph and rooted at node $k \in \mathcal{V}$ which spans all nodes . It is said *strongly rooted in* k

[1] a graph is said to be *undirected* if $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$ and to be *directed* otherwise.

if node k is directly connected to all other nodes, i.e. $(j,k) \in \mathcal{E}, \forall j \in \mathcal{V}$.

A graph is *strongly connected* if there is a path from any node to any other node in the graph. Clearly, if a graph is strongly connected graph this implies the graph is also a rooted graph for any node. A graph is *complete* if $(i,j) \in \mathcal{E}, \forall i,j \in \mathcal{V}$.

The *concatenation* of two graphs $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}, \mathcal{E}_2)$ is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) = \mathcal{G}_2 \circ \mathcal{G}_1$ such that $(i,j) \in \mathcal{E}$ if there exists $k \in \mathcal{V}$ such that $(k,j) \in \mathcal{E}_1, (i,k) \in \mathcal{E}_2$. The concatenation $\mathcal{G} \circ \mathcal{G}$ describes the communication feasible in the network $\mathcal{G}$ in two hops.

Similarly, the *union* of two graphs is a graph $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ for which $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_1$. Clearly $\mathcal{G}_1 \cup \mathcal{G}_2 = \mathcal{G}_2 \cup \mathcal{G}_1$ while $\mathcal{G}_2 \circ \mathcal{G}_1 \neq \mathcal{G}_1 \circ \mathcal{G}_2$.

We recall than that, given a matrix $P \in \mathbb{R}^{N \times N}$, we define *graph associated to the matrix* P the graph $\mathcal{G}_P = (\mathcal{N}, \mathcal{E}_P)$, where the nodes $\mathcal{N} = \{1, 2, \ldots, N\}$ and $(i,j) \in \mathcal{E}_P$ if and only if $[P]_{i,j} \neq 0$.

A matrix P is said to be *adapted to, or compatible with, the graph* $\mathcal{G}$ if $[P]_{ij} \neq 0$ only if $(j,i) \in \mathcal{E}$. We write in this case $P \sim \mathcal{G}$. It is clear, then, that $P \sim \mathcal{G}$ if and only if $\mathcal{E}_P \subseteq \mathcal{E}$.

1.2 Review of Consensus Theory

In this section we briefly recall some well known results on consensus theory which we will use in the following chapters. Let us start by formalizing the concept of distributed algorithm implementable on a network described by the graph $\mathcal{G}$. Any recursive algorithm where the i-th node's update law depends only on the state of i and on the state of its neighbors $j \in \mathcal{V}(i)$

$$x_i(t+1) = f(x_i(t), x_{j_1}(t), \ldots x_{j_{d(i)}}(t), t) \qquad \text{with } j_1, \ldots, j_{d(i)} \in \mathcal{V}(i)$$

is said to be *adapted to the graph* $\mathcal{G}$ and to be performed needs only of information that each node can retrieve via communications permitted by the graph $\mathcal{G}$.

Let us introduce then the consensus problem: a network is said to reach consensus if all nodes agree on a common value, that is, if all the nodes reach asymptotically the same value. More precisely

Definition 1 (Algorithm achieving Consensus)**.**
A recursive distributed algorithm adapted to the graph $\mathcal{G}$ *is said to asymptotically* achieve consensus

1. *if there exist* T *such that* $x_i(T) = \alpha \ \forall \ i \in \mathcal{V}$ *then it implies*

$$x_i(t) = \alpha \qquad \forall t \geq T \text{ and } \forall i \in \mathcal{V} \tag{1.1}$$

2. if all the nodes reach asymptotically a common value

$$\lim_{t\to+\infty} x_i(t) = \alpha \qquad \forall i \in \mathcal{V} \tag{1.2}$$

If moreover this common value is the average of the nodes' initial conditions

$$\alpha = \frac{1}{N}\sum_{i\in\mathcal{V}} x_i(0),$$

then the algorithm is said to achieve average consensus.

If $x_i(t)$ is a random variable, than the limit in (1.2) has to be interpreted in a probabilistic sense. We will then speak about algorithms achieving consensus almost surely, in mean, in probability and so on.

The consensus problem has attracted great attention of researchers in the last few years and many algorithms have been proposed and analyzed in the literature, [8, 9, 10, 11, 12, 13, 14, 15, 16], to mention only few of them. Most of them are linear but also nonlinear algorithms have been proposed, for instance [17, 18], even achieving consensus in a finite time. Linear consensus algorithms maintain though a prominent role, since they are simpler to be analyzed. In particular, many results are available on their speed of convergence and it is possible to carry out a probabilistic analysis of their performances in presence of noise or quantization and to effectively take into account the effect of a time varying topologies, [19, 20, 21, 22, 23, 24, 25].

Let us consider, then, a linear recursive distributed algorithm, adapted to the graph $\mathcal{G}$. It can be written as

$$x_i(t+1) = p_{ii}(t)x_i(t) + \sum_{j\in\mathcal{V}(i)} p_{ij}(t)x_j(t) \tag{1.3}$$

Collecting all $x_j(t)$ in a vector $x(t) \in \mathbb{R}^N$ (1.3) can be rewritten as

$$x(t+1) = P(t)x(t) \tag{1.4}$$

where $P(t) \in \mathbb{R}^{N\times N}$, $P(t)$ compatible with $\mathcal{G}$.

Condition 1 of Definition 1 requires that[2], whenever $x(T) = \alpha\mathbb{1}$, then $x(t) = \alpha\mathbb{1}$ for all $t > T$, that is

$$P(t)\mathbb{1} = \mathbb{1} \qquad \forall t. \tag{1.5}$$

[2]Recall that we denote with $\mathbb{1}$ the vector of $\mathbb{R}^N$ having all entries equal to 1, see section 2 of the Introduction.

A matrix $P(t)$ having such a property is said to be *quasi-stochastic*.

If it holds moreover that $[P_{i,j}(t)] \geq 0 \ \forall i, j \in \mathcal{V}$, than the matrix is said to be *stochastic*. A stochastic matrix P is said *doubly stochastic* if also $\sum_{i=1}^{N} p_{ij} = 1$, i.e. each column sums to unity. Clearly, if a stochastic matrix is symmetric then it is also doubly stochastic: $\mathbb{1}^T P = (P^T \mathbb{1})^T = \mathbb{1}^T$.

Stochastic matrices have been widely studied in many fields and they are well understood. In particular, even if the assumption $[P_{i,j}(t)] \geq 0$ is not strictly necessary for consensus, convergence results for consensus algorithms are usually available only for stochastic matrices and there are not equivalent results of the same generality for quasi–stochastic matrices.

As a final comment, note that (1.5) reads that the sum of the entries of any row of the matrix is always 1, that is

$$p_{ii}(t) = 1 - \sum_{j \in \mathbb{N}(i)} p_{ij}(t).$$

Equation (1.3) can then be rewritten as

$$\begin{aligned} x_i(t+1) &= p_{ii}(t)x_i(t) + \sum_{j \in \mathbb{N}(i)} p_{ij}(t)x_j(t) \\ &= \left(1 - \sum_{j \in \mathbb{N}(i)} p_{ij(t)}\right) x_i(t) + \sum_{j \in \mathbb{N}(i)} p_{ij}(t)x_j(t) \\ &= x_i(t) + \sum_{j \in \mathbb{N}(i)} p_{ij}(t)(x_i(t) - x_j(t)) \end{aligned}$$

which has an appealing interpretation: the new state is the old state plus a control action that corrects the distance from consensus.

In the next section we summarize some sufficient conditions to guarantee that a linear algorithm reaches consensus.

1.3 Convergence and Design of Consensus Algorithms

To present convergence results we abandon the general approach we have taken so far and consider three different situations: constant consensus matrix $P(t) = P$, deterministic time-varying strategies $P(t)$ and randomized strategies where $P(t)$ is drawn from some distributions on a set of stochastic matrices $\mathcal{P}$.

1.3.1 Constant Consensus Matrix

Let us consider first the simple case a fixed communication scheme: we assume that all the information exchanges prescribed by the communication graph happen between two iterations of the algorithm. We assume moreover that communications can be considered reliable, for instance thanks to a communication protocol that allows to recover packet losses, say prescribing retransmission.

Clearly, this is just a rough model of communications in a network, valid if the time between two subsequent consensus iterations is sufficiently large. More interesting communication scenario will be analyzed in the following of this section. Nevertheless, this scenario allows to consider a class of particularly simple consensus algorithms, for which many interesting results can be stated. In fact, in such a scenario we can make use a consensus algorithm based on a sequence of constant matrices $P(t) = P$ and the consensus algorithm reduces to simple and well understood linear time-invariant dynamical system with transition matrix P.

Condition 1 of Definition 1 reads then that any point of the space span($\mathbb{1}$) has to be an equilibrium point and the space has to be attractive. From system theory we inherit the following proposition, [15]

Proposition 1. *A matrix P yields asymptotically to consensus if and only if*

1. *1 is an eigenvalue having $\mathbb{1}$ as an eigenvector.*
2. *1 is the only eigenvalue on the unit disk and its algebraic multiplicity is 1 (i.e. it is a simple root of the characteristic polynomial of P)*
3. *all the other eigenvalues have modulus strictly less than 1*

$$|\lambda_i(P) < 1| \qquad \forall i = 2, \dots, N$$

Note moreover that a matrix P satisfying the above three property achieves average consensus if and only if $\mathbb{1}^T P = \mathbb{1}$, in fact in this case

$$\sum x_i(t) = \mathbb{1}^T x(t) = \mathbb{1}^T P x(t-1) = \mathbb{1}^T x(t-1) = \sum x_i(t-1)$$

To the best of our knowledge, there is no other condition regarding quasi-stochastic matrices characterizing the convergence of the consensus algorithm. Something more can be said if we restrict to non–negative matrices, i.e., stochastic matrices. The following result is a straightforward consequence of standard results in on stochastic matrices

Theorem 2. *Given a stochastic matrix P, such that $\mathcal{G}_P$ is a directed graph containing all self–loops (i,i) and is $\mathcal{G}_P$ rooted, then P solves the consensus problem.*

If in addition P is doubly stochastic, then $\mathcal{G}_P$ is strongly connected and P solves the average consensus problem. Moreover, the convergence rate in both cases is exponential and it is given by second largest eigenvalue in absolute value of the matrix P.

Proof.
It is a straightforward consequence of standard results on stochastic matrices [26, pag. 88 and pag. 95] □

The above theorem shows how convergence conditions can be reframed as a graph problem which is easy to verify [27].

Algorithm Design

The question of how to assign the weights P_{ij} arises naturally at this point. In [23] it was presented a constructive algorithm that, given a symmetric graph, find a doubly stochastic matrix P compatible with the graph which maximize the rate of convergence of the consensus algorithm. Nevertheless, as it was shown in [28, 29], when consensus is not an objective per se, but rather it is used to solve an estimation or control problem, the convergence speed is not a meaningfully index for performance evaluation. In these cases it is important to consider other performance measures, more tightly related to the actual objective pursued. Different costs arise from different problems and this difference might be crucial: indeed, it can be shown by examples that considering a different performance measure can lead to preferring different graph topologies,[28, 29].

Moreover, it is common to assign consensus weights according to simpler roles, rather than carrying out complex optimization procedures, that might reveal extremely expensive on a computational point of view. We report in the following three of these methods commonly used in literature.

Nearest Neighbor role:

Every node weights uniformly its own state and the state of its neighbors. Therefore

$$P_{i,j} = \begin{cases} \frac{1}{d(i)+1} & j \in \mathcal{V}(i) \cup \{i\} \\ 0 & \text{otherwise} \end{cases}$$

It is clear that the matrix P obtained is stochastic and compatible with the graph, but in general not doubly stochastic. It can therefore be used to enforce consensus but not average consensus. Note moreover that the weights $P_{i,j}$ can be easily and distributedly computed by every node. In fact, the weight that node i has to apply to the states received by other nodes is determined by its degree, namely the number of messages it has received during two subsequent iterations of the consensus algorithm.

Maximum-degree weights:

Let us call d the maximum degree of the network

$$d = \max_{i \in \mathcal{V}} \; d(i) = \max_{i \in \mathcal{V}} |\mathcal{V}(i)|$$

Every node weights the state of its neighbor $\frac{1}{d+1}$ and give to its own state the remaining weight. Therefore

$$P_{i,j} = \begin{cases} \frac{1}{d+1} & j \in \mathcal{V}(i) \\ 1 - \frac{d(i)}{d+1} & i = j \\ 0 & \text{otherwise} \end{cases}$$

If the communication graph $\mathcal{G}$ is undirected, this method allows to have symmetric weights $P_{i,j} = P_{j,i}$. P is therefore symmetric and stochastic. It requires, although, to compute the maximum degree of the network. There are algorithms to compute the maximum over a network with local communications, but they might require a significant computation burden.

Metropolis weights:

Set

$$P_{i,j} = \begin{cases} \frac{1}{\max\{d(i),d(j)\}+1} & j \in \mathcal{V}(i) \\ 1 - \sum_{k \in \mathcal{V}(i)} P_{i,k} & i = j \\ 0 & \text{otherwise} \end{cases}$$

Again, the matrix P so obtained is stochastic and compatible with the graph $\mathcal{G}$ and, if the graph is undirected, this method leads to a P that is also symmetric and therefore doubly stochastic. In this case P can hence be used to enforce average consensus. It is worth noting that in the Metropolis weights case the node i can compute its weights P_{ij} $\forall j \in \mathcal{V}(i)$ exchanging $d(i)$ with its neighbors. Metropolis weights can therefore be easily computed in distributed manner and it is no need to evaluate the maximum degree of the network.

1.3.2 Deterministic Sequence of Stochastic Matrices

A more realistic communication model considers a time-varying communication graph $\mathcal{G}(t)$, rather than a constant one. Time dependence of the communication graph might be due to deterministic scheduling protocols, that manage nodes access to the medium to avoid collisions, or to the fact that we are considering mobile agents whose connectivity depends on their mutual position. $\{\mathcal{G}(t,\omega)\}_{t\in\mathbb{N}}$ might moreover represent the deterministic realization of the connectivity graph, when it occurs a particular sequence of random events ω, such as node failure and packet losses. The worst realization can be used to perform worst case analysis.

Let us consider, therefore, a deterministic sequence of stochastic matrices $\{P(t)\}_{t=0}^{+\infty}$ and the corresponding associated graphs $\mathcal{G}_P(t) = \mathcal{G}_{P(t)}$. Assume moreover that $\mathcal{G}_P(t)$ contains all the self–loops, for all time instants. Under this assumption it holds the following result

Theorem 3. *If there exists a finite positive integer number T such that the graphs*

$$\bar{\mathcal{G}}_T(\tau) = \mathcal{G}_P(\tau + T - 1) \circ \ldots \circ \mathcal{G}_P(\tau), \qquad \tau = 0, 1, \ldots$$

are rooted, then the sequence $\{P(t)\}$ solves the consensus problem.

If the matrices $P(t)$ are all doubly stochastic, then the sequence $\{P(t)\}$ solves the average consensus problem.

The above theorem shows that it is not necessary for the communication graph to be connected at any time instant but rather over a fixed time window [30, 24].

Finally we note that the strategies previously proposed to obtain assign consensus weights can be extended to the time varying case. In fact, these techniques are based on the degree of the nodes, that corresponds to the number of communication they received between two subsequent algorithm iterations. Hence we have

Nearest Neighbor role:

$$P_{i,j}(t) = \begin{cases} \frac{1}{d(i,t)+1} & j \in \mathcal{V}(i) \cup \{i\} \\ 0 & \text{otherwise} \end{cases}$$

Metropolis weights:

Set

$$P_{i,j}(t) = \begin{cases} \frac{1}{\max\{d(i,t),d(j,t)\}+1} & j \in \mathcal{V}(i,t) \\ 1 - \sum_{k\in\mathcal{V}(i)} P_{i,k}(t) & i = j \\ 0 & \text{otherwise} \end{cases}$$

Note moreover that this method yields to a symmetric matrix $P(t)$ if the graph $\mathcal{G}(t)$ is undirected.

More computationally demanding is the evaluation of the maximum degree weights since it prescribes to evaluate, at each time instant, $\max_{i\in\mathcal{V}} d(i,t)$, unless an a-priori bound on the degree is known: $\bar{d}(t) \leq d_{\max}$, $\forall t$. In this case set

$$P_{i,j}(t) = \begin{cases} \frac{1}{d_{\max}+1} & j \in \mathcal{V}(i,t) \\ 1 - \sum_{k\in\mathcal{V}(i)} P_{i,k}(t) & i = j \\ 0 & \text{otherwise} \end{cases}$$

It has to be noted that the convergence of the consensus algorithms obtained using these weight strategies can be guaranteed by ad hoc connectivity conditions, that are even weaker than the ones prescribed by Theorem 3, as shown in [31, 32].

1.3.3 Randomly Drawn Consensus Matrices

In many cases, especially in sensors networks, communication happens according to randomized communications protocols [33, 34], prescribing the random activation of certain links with designed probabilities. These strategies are used to handle the accesses to the communication channel, avoiding cumbersome communication scheduling and reducing the need of time-synchronization. An effective use of randomized protocols may moreover allow to reduce power consumption. A further cause of randomness in the communication is the potential unpredictably of the environment where these protocols are implemented: packet losses, collisions and node failure [35] are in fact rather common in a sensor network.

Therefore, it is rather natural to consider randomized consensus algorithms that exploit randomized communication protocols. Let us consider, then, a random i.i.d. sequence of stochastic matrices $\{P(t)\}_{t=0}^{+\infty}$ drawn according to some distribution from a set, $\mathcal{P} = \{P(\alpha) \quad \alpha \in \mathbb{A}\}$, of stochastic matrices compatible with the graph $\mathcal{G}$, $P(\alpha) \sim \mathcal{G} \ \ \forall P(\alpha) \in \mathcal{P}$, such that

$$[P(\alpha)]_{ii} > 0 \qquad \forall i \text{ and } \forall P(\alpha) \in \mathcal{P} \tag{1.6}$$

The following theorem shows a remarkable fact: the convergence of a randomized consensus algorithm is determined by its average consensus matrix $\overline{P} = \mathbb{E}[P(t)]$

Theorem 4. *Assume that (1.6) holds true and consider the stochastic matrix $\overline{P} = \mathbb{E}[P(t)]$. If $\mathcal{G}_{\overline{P}}$ is rooted, then the sequence $\{P(t)\}$ solves the consensus problem almost surely.*

If, in addition, $P(t)$ are all doubly stochastic, then they solve the probabilistic average consensus problem.

Hence, the convergence of a randomized algorithm reduces to the study of the convergence of the constant matrix algorithm.

Note, although, that to guarantee average consensus it is not sufficient that $\mathbb{E}[P(t)]$ is doubly stochastic but rather it is required that $P(t)$ is doubly stochastic at each time instant t.

We present now some randomized communication strategies and the associated randomizes consensus algorithms, commonly adopted in sensor networks: the *asymmetric gossip*, where a node i transmits a message to one of its neighbor nodes, the *symmetric gossip*, where a node i transmits a message one of its neighbor nodes and receives a reply message from it, and finally the *broadcast* communication, where a node transmits a message to all its neighbors.

Asymmetric Gossip:

A node, say i, randomly wakes up and randomly picks up one of its neighbor nodes, $j \in \mathcal{V}(i)$ to which it sends its state x_i, as depicted in figure 1.1. We call w_{ij} the probability that the link $(i,j) \in \mathcal{E}$ is selected. The consensus iteration associated to this communication is

$$\begin{aligned} x_i(t+1) &= x_i(t) \\ x_j(t+1) &= (1-p_{ji})x_j(t) + p_{ji}x_i(t) \\ x_k(t+1) &= x_k(t) \qquad\qquad \forall k \neq i,j \end{aligned}$$

Therefore, with probability $w_{i,j}$, it is extracted the matrix $P(t) = P^{\mathrm{AG}}_{(i,j)}$

$$P^{\mathrm{AG}}_{(i,j)} = I - p_{ji}e_j(e_i - e_j)^T.$$

Moreover, we set all the weights p_{ji} to a common value

$$p_{ji} = p, \qquad 0 < p < 1.$$

Example 1. In the case depicted in figure 1.1, setting $p = 1/3$,

$$P^{\mathrm{AG}}_{(4,1)} = \begin{bmatrix} 2/3 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

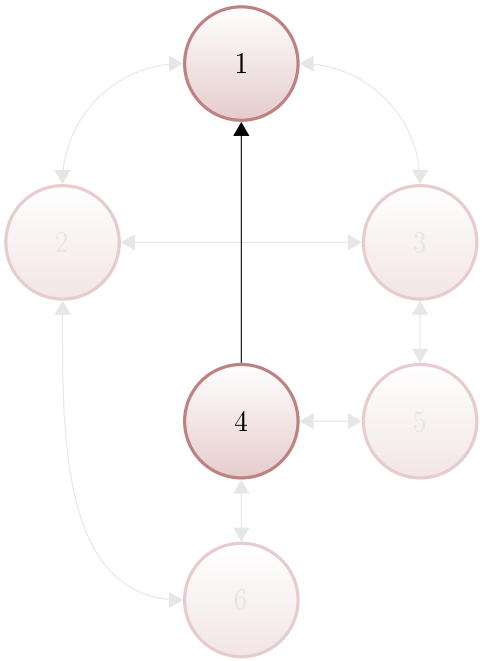


Figure 1.1: Asymmetric gossip. In this example node 4 wakes up and sends its estimate to node 1

Symmetric Gossip:

Consider an undirected graph $\mathcal{G}$. A node, say i, randomly wakes up and randomly picks up one of its neighbor nodes, $j \in \mathcal{V}(i)$. Nodes i and j exchange then their states. The communication is therefore bidirectional. A pictorial representation of this situation is reported in figure 1.2. Again, we call w_{ij} the probability that the link $(i,j) \in \mathcal{E}$ is selected. The consensus iteration associated to this communication is

$$\begin{aligned} x_i(t+1) &= (1-p_{ij})x_i(t) + p_{ij}x_j(t) \\ x_j(t+1) &= (1-p_{ji})x_j(t) + p_{ji}x_i(t) \\ x_k(t+1) &= x_k(t) \qquad\qquad \forall k \neq i,j \end{aligned}$$

where moreover, to enforce symmetry, we put $p_{ji} = p_{ij}$. Therefore, with probability $w_{i,j}$, it is extracted the matrix $P(t) = P^{\mathrm{SG}}_{(i,j)}$

$$P^{\mathrm{SG}}_{(i,j)} = I - p_{ij}(e_i - e_j)(e_i - e_j)^T.$$

Again, we set all the weights p_{ij} to a common value

$$p_{ij} = p, \qquad 0 < p < 1$$

and typically $p = 1/2$.

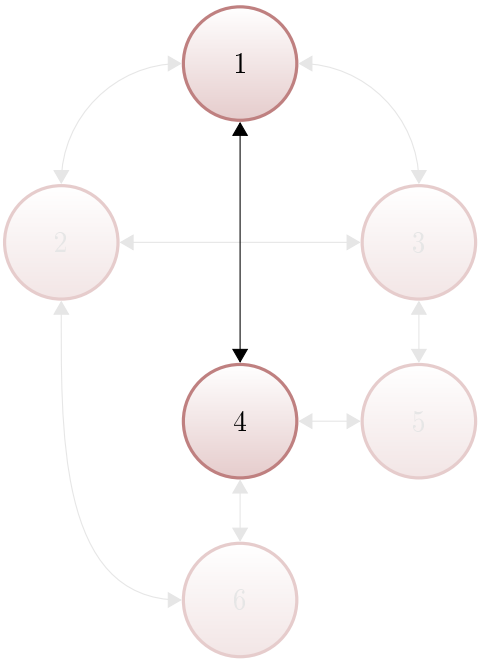


Figure 1.2: Symmetric gossip. In this example node 4 wakes up and sends its estimate to node 1. Node 1 responds sending its estimate to node 4. Clearly a bidirectional channel is need for this communication scheme.

Example 2. In the case depicted in figure 1.2

$$P^{\mathrm{SG}}_{(4,1)} = \begin{bmatrix} 1/2 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Broadcast:

A node, say i, randomly wakes up and broadcasts its state to anyone can hear its transmission. All its neighbor nodes receive therefore the state x_i. A pictorial representation of this situation is reported in figure 1.3. We call w_i the probability that node $i \in \mathcal{V}$ is selected. The consensus iteration associated to this communication is

$$\begin{aligned} x_i(t+1) &= x_i(t) \\ x_j(t+1) &= (1-p_{ji})x_j(t) + p_{ji}x_i(t) \qquad \forall j \in \mathcal{V}(i) \\ x_k(t+1) &= x_k(t) \qquad \forall k \notin \mathcal{V}(i) \end{aligned}$$

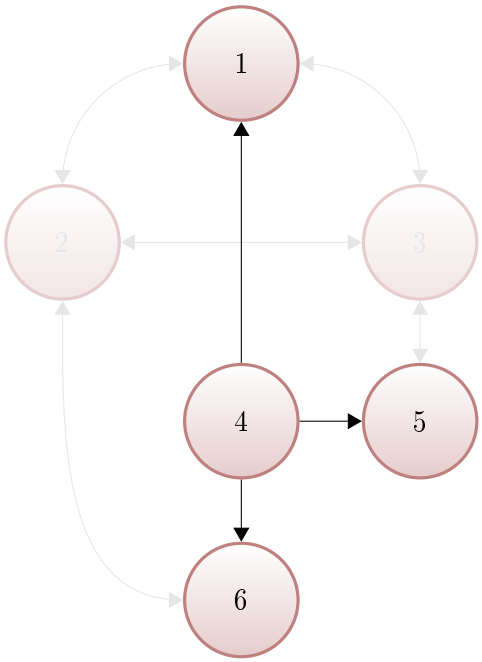


Figure 1.3: Broadcast. In this example node 4 wakes up and broadcasts its state to its neighbor nodes.

Therefore, with probability w_i, it is extracted the matrix $P(t) = P_i^{\mathrm{B}}$

$$P_i^{\mathrm{B}} = I - \sum_{j \in \mathcal{V}(i)} p_{ji} e_j (e_i - e_j)^T.$$

Moreover, we set all the weights p_{ij} to a common value

$$p_{ij} = p, \qquad 0 < p < 1.$$

The parameter p plays, this time, an important rule in tuning the algorithm. We will discuss its importance in the following. As a finally remark, note that a broadcast communication corresponds to $d(i)$ parallel asymmetric gossip iterations.

Example 3. In the case depicted in figure 1.3, setting $p = 1/4$, we get that

$$P_4^{\mathrm{B}} = \begin{bmatrix} 3/4 & 0 & 0 & 1/4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 3/4 & 0 \\ 0 & 0 & 0 & 1/4 & 0 & 3/4 \end{bmatrix}$$

Algorithm Design

The most relevant parameter to be designed in the case of randomized algorithms is the selection probability measure, namely the edge selection probabilities $\{w_{ij}\}_{(i,j)\in\mathcal{E}}$, in the gossip case, or node selection probabilities $\{w_i\}_{i\in\mathcal{V}}$, in the broadcast case.

An elegant and meaningful way to compute these selection probabilities is by solving an optimization problem, determining in this way the selection probabilities that minimize a suitable cost function, as for instance the speed of convergence of the consensus algorithm, [33]. As already mentioned, different performance assessment criteria might lead to very different cost functions, as pointed out in [28, 29], and therefore to very different selection probabilities. In chapter 5, for instance, we will make use of randomized consensus as a step of an estimation algorithm and we will compute the selection probabilities to minimize the steady state estimation error.

In other cases, one might prefer to assign simple and robust selection probabilities rather that solving a complex optimization problem. It has moreover to be noted that in certain cases, as in the one presented in chapter 5, the little improvement in the performance that can be achieved solving an optimization problem seems to be not worth the computation effort it takes. A simple choice is to assign uniform selection probability to every link or node:

$$w_{i,j} = \frac{1}{|\mathcal{E}|} \qquad\qquad w_i = \frac{1}{|\mathcal{V}|}.$$

Another very reasonable choice in the gossip case is to assume that every node wakes up with uniform probability $\frac{1}{|\mathcal{V}|}$ and choses then a node uniformly within its neighbors. Therefore, the probability that the link (i,j) is given by the probability i is selected:

$$w_{i,j} = \frac{1}{|\mathcal{V}|}\frac{1}{d(i)}.$$

The intuition suggests that this edge selection probability measure should lead to faster consensus algorithms that the uniform one on most of the graph. In fact, using the latter strategy every node has the same change to spread out its information while with the uniform probability nodes with few connections are penalized.

Another degree of freedom in the design of the algorithm is the consensus weight p. Roughly speaking, a small p means little confidence in the neighbor opinion and leads to slow convergence of consensus algorithms. On the contrary, p close to $1/2$ means high confidence in the neighbor opinion that produces fast information spread and hence fast consensus algorithms.

Nevertheless, the choice of p has another important effect in the broadcast case. In [34] it is shown that expected [3] steady state of the broadcast algorithms is average of the initial conditions and the steady state dispersion of a single realization depends on the weight p. The closest p is chose to zero, the smaller the dispersion.

In many cases, especially in the field of distributed estimation, to reach average consensus is a key feature and it is not sufficient to simply reach consensus. Therefore, in the broadcast case, the choice of the consensus weight p has to be a trade–off between steady state accuracy in reaching average consensus and speed of convergence of the algorithm. On the contrary, in the symmetric gossip algorithm, where it is guaranteed achievement of average consensus, it is usually chosen $p = 1/2$.

Analysis for Standard Selection Probabilities.

To better illustrate the issues presented so far, let us analyze the symmetric gossip and the broadcast with the commonly used selection probabilities. Given an undirected graph $\mathcal{G}$, let us consider the symmetric gossip with probability $w_{i,j} = \frac{1}{|\mathcal{E}|}\frac{1}{d(i)}$ that the link (i,j) is selected . Noting that $P^{\mathrm{SG}}_{(i,j)} = P^{\mathrm{SG}}_{(j,i)}$, it results that the expected consensus matrix $\overline{P}^{\mathrm{SG}} = \mathbb{E}[P^{\mathrm{SG}}]$ is

$$[\overline{P}^{\mathrm{SG}}]_{mn} = \begin{cases} 1 - \sum_{i \in \mathcal{V}(n)} \frac{p}{N}\left(\frac{1}{d(i)} + \frac{1}{d(j)}\right) & \text{if } m = n; \\ \frac{p}{N}\left(\frac{1}{d(i)} + \frac{1}{d(j)}\right) & \text{if } m \neq n \text{ and } (m,n) \in \mathcal{E}; \\ 0 & \text{otherwise.} \end{cases}$$

Obviously $\overline{P}^{\mathrm{SG}} = (\overline{P}^{\mathrm{SG}})^T$ since all the gossip matrices $P^{\mathrm{SG}}_{(i,j)}$ that can be drawn are symmetric by construction. Moreover, we have $\mathcal{G}_{\overline{P}^{\mathrm{SG}}} = \mathcal{G}$, i.e. the graph associated with the expected consensus matrix $\overline{P}^{B}$ coincides with the underlying communication graph $\mathcal{G}$. Furthermore, $P(t)_{ii} > 0 \ \forall t$. Therefore, using theorem 4, we have that if the graph $\mathcal{G}$ is strongly connected then the symmetric gossip achieves consensus w.p. 1. Since moreover the matrices $P(t)$ are all doubly stochastic then the symmetric gossip achieves average consensus w.p. 1.

Consider then the broadcast algorithm with uniform node selection probability $w_i = 1/|\mathcal{V}|$. It results that the expected consensus matrix $\overline{P}^{\mathrm{B}} = \mathbb{E}[P^{\mathrm{B}}]$

[3] where the expectation, here, is taken with respect to the randomness introduced by the random selection of the broadcasting node.

is given by:

$$[\overline{P}^{\mathrm{B}}]_{mn} = \begin{cases} 1 - \frac{p \cdot d(n)}{N} & \text{if } m = n; \\ \frac{p}{N} & \text{if } m \in \mathcal{V}(n); \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\mathcal{G}_{\overline{P}^B} = \mathcal{G}$. Again, if $\mathcal{G}$ is strongly connected, it implies that the randomized broadcast guarantees probabilistic consensus, although it does not guarantee average consensus for all possible realizations of $P^B(t)$, since the matrices $P(t)$ are not doubly stochastic. Even if the broadcast matrices are never symmetric, the *expected* consensus matrix $\overline{P}^B$ is symmetric and hence doubly stochastic. Therefore the expected steady state of the algorithm is the average of the initial conditions. As we said, in [34] was shown that the parameter p can be tuned to obtain a final consensus value closer to the average of the initial conditions, at the price of slower convergence rate. It was moreover proved, in the case of a complete graph, that the dispersion of the final consensus value from the average of the initial conditions decreases as the number of nodes increases.

One might also wonder if $\overline{P}^B$ provides some information about convergence rate for the randomized strategy. In [34] there is an extensive analysis of rates of convergence. The main message being that the second largest eigenvalue of $\overline{P}^B$ provides only an lower bound rate of convergence.

Packet Loss and Link Failure So far, we have not considered packet losses and link failure. These events can easily be taken into account in this set up. Assume, that the failure probability of a transmission from node i to a node $j \in \mathcal{V}(i)$ is equal to $1 - c_{i,j}$. For sake of simplicity assume that $c_{i,j} = c$ for all nodes.

When link failure happens in broadcast communication, the matrix P_i^B needs to be modified with $[P_i^B]_{jj} = 1, [P_i^B]_{ji} = 0$. Instead, when it happens in symmetric gossip, there is no communication at all, and then no update is performed, i.e. $P_{(i,j)}^{SG} = I$. Recomputed the expected consensus matrices taking packet loss probability into account gives the following results:

$$[\overline{P}^{\mathrm{SG}}]_{mn} = \begin{cases} 1 - \sum_{i \in \mathcal{V}(n)} \frac{cp}{N} \left(\frac{1}{d(i)} + \frac{1}{d(j)} \right) & \text{if } m = n; \\ \frac{cp}{N} \left(\frac{1}{d(i)} + \frac{1}{d(j)} \right) & \text{if } m \neq n \text{ and } (m,n) \in \mathcal{E}; \\ 0 & \text{otherwise.} \end{cases}$$

$$[\overline{P}^{\mathrm{B}}]_{mn} = \begin{cases} 1 - \frac{cp \cdot d(n)}{N} & \text{if } m = n; \\ \frac{cp}{N} & \text{if } m \in \mathcal{V}(n); \\ 0 & \text{otherwise.} \end{cases} .$$

Since the graph associated with the expected consensus matrix still coincides with the underlying communication graph $\mathcal{G}$ the above discussion is still valid.

Comparison between Gossip and Broadcast

Gossip communications naturally arise when the network we are dealing with relies on point to point communications at the physical layer, for instance it is based on wired links. On the contrary, broadcast communications naturally arise when considering wireless communication based on radio transmissions. Nevertheless, an appropriate communication protocol may be used to enforce symmetric gossip in a radio channel. The simplest version of such a protocol is the following:

1. A node randomly wakes up, say i, and broadcasts its state
2. Randomly, among all the nodes that received the transmission, one answers to the broadcaster node i, say node j.
3. Node i acknowledges node j that it received the replay.

Therefore, if working in a wireless environment, symmetric gossip is somehow inefficient on a communication point of view, requiring least two communications and an acknowledgment transmission to perform two state updates. On the contrary, broadcast, exploiting the potential of the radio channel, is very effective on a communication point of view : only one transmission is required to perform $d(i)$ state updates. Broadcast hence exhibits faster convergence than symmetric gossip. Unluckily broadcast algorithms can be used to enforce consensus but not average consensus, since P_i^B is not doubly stochastic, while symmetric gossip consensus algorithms are guaranteed to yield to average consensus if they reaches consensus. Figure 1.5 depicts exactly this two points, comparing a realization of the symmetric gossip consensus algorithm and of the broadcast consensus algorithm over the network depicted in figure 1.4 The fact that symmetric gossip leads to a converge is then much slower can also be guessed, at least in the case of standard selection probabilities, noting that the off-diagonal elements of the matrix $\overline{P}^{\mathrm{SG}}$ are smaller than their counterparts in $\overline{P}^{\mathrm{B}}$, i.e. there is slower information propagation. We will discuss these differences also in Chapter 2.

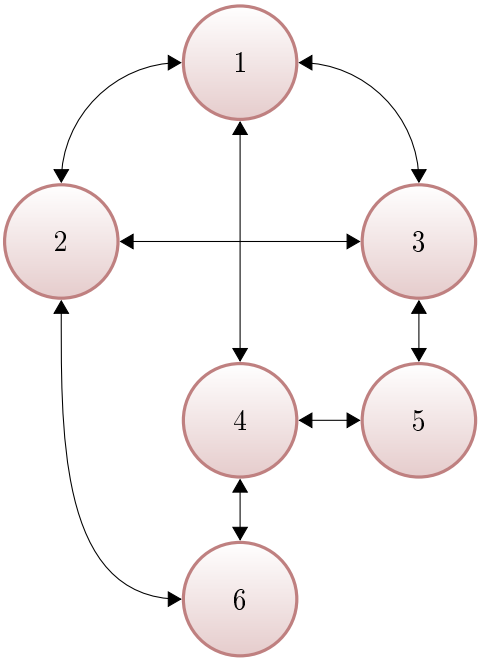
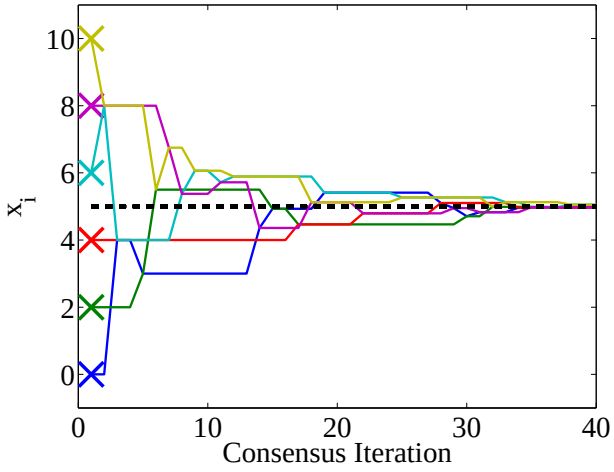
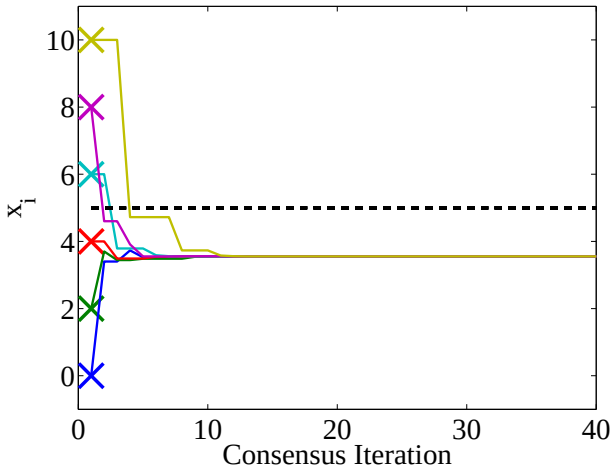


Figure 1.4: An example of network graph



(a) Symmetric Gossip



(b) Broadcast

Figure 1.5: Comparison between Symmetric Gossip and Broadcast: a typical realization is depicted. The dashed line represents the average of the initial conditions.

2

A class of Consensus Computable Functions

Consider the problem of evaluating a quantity that, to be computed, requires data from all nodes of a sensor network. Let us use the adjective **global** to describe such a quantity. One immediately realizes how this issue, the issue of fusing sensors data to compute global quantities, is crucial when dealing with estimation over sensor networks, as it is clear that such a problem is central in any multi-agents system and it has to be tackled when studying applications such as coordination of mobile robots, formation control, clock synchronization and so on.

We have seen in the previous chapter that one problem of this kind, the problem computing the average of the values stored in the nodes, has received great attention from the scientific community in recent years, leading to the development of an organic theory that goes under the name of consensus theory.

In this chapter we show how the problem of computing many other global quantities can be casted into an average consensus problem, making consensus algorithms one of the basic tools of distributed computing.

More precisely, in this chapter we characterize a class of global quantities computable through average consensus. An analogous result has been proposed in the context of continuous-time consensus problems by Bauso et al. [36] and later generalized by Cortes [37], leading to the so-called χ-consensus. Here we proposed a discrete time counterpart which has less strict hypothesis and a much simpler to proof.

Then, we consider in detail the problem of computing four important global quantities:

- Generalized means
- Distributed Kalman Filter
- Maximum Likelihood Parameter Estimation
- Least Square Parameter Estimation

and we show how to reduce this to an average consensus problem. This allows to perform these computations in a distributed manner by means of any standard average consensus algorithm.

The distributed algorithm for least square parameter estimation so obtained will be then used in Chapter 3 to estimate the wireless channel parameters.

Chapter organization

The chapter is organized as follows. In Section 2.1 we present a result that characterizes a class of global quantities whose computation can be casted into an average consensus problem. In each of the subsequent four sections we focus on the problem of computing one of the above global quantities: in Section 2.2 the generalized means, in Section 2.3 the Kalman filter estimate, in Section 2.4 the Maximum Likelihood estimate and finally in Section 2.5 the Least Square estimate. The chapter is concluded by Section 2.6 where we give some final comments and we discuss some of the issues still open.

2.1 A class Consensus-computable global quantities

The following proposition simply states that, if the variable we need to compute is a function of the mean of some transformation of the data, then it can be computed using average consensus algorithms. We will see in the following that many problems of interest fits in this category.

Proposition 5. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph associated to a sensor network with N nodes, i.e. $N = |\mathcal{V}|$. Let moreover $z_i \in \mathbb{R}^p, i = 1, \dots, N$ be the data available to i-th node. Consider the global quantity $\xi \in \mathbb{R}^r$. If such a quantity can be written as follows:*

$$\xi = f\left(\frac{1}{N}\sum_{i\in\mathcal{N}} g_i(z_i)\right) \tag{2.1}$$

*where $g_i : \mathbb{R}^p \to \mathbb{R}^q$ are generic functions, and $f : \mathbb{R}^q \to \mathbb{R}^r$ is continuous, than it is **consensus–computable**.*

To compute ξ using consensus, consider a consensus algorithm $\{P(t)\}$, namely a sequence of doubly stochastic matrices compatible with the graph $P(t) \sim \mathcal{G}$ $\forall t$ which solves the (probabilistic) average consensus problem. Set, then,

$$x_i(0) = g_i(z_i), \qquad \forall i \in \mathcal{V}. \tag{2.2}$$

Run then the consensus algorithm with such an initial condition:

$$x_i(t+1) = p_{ii}(t)x_i(t) + \sum_{j\in\mathcal{V}(i)} p_{ij}(t)x_j(t), \tag{2.3}$$

where $p_{ij}(t) = [P(t)]_{ij}$ and compute at each time instant

$$\eta_i(t) = f(x_i(t)). \tag{2.4}$$

One has that:

$$\lim_{t\to\infty} \eta_i(t) = \xi, \quad \forall i \in \mathcal{V}$$

Proof.
If $\{P(t)\}$ solves the average consensus problem, then

$$\lim_{t\to\infty} x_i(t) = x_{ave} = \frac{1}{N}\sum_{i=1}^{N} x_i(0) = \frac{1}{N}\sum_{i=1}^{N} g_i(z_i)$$

since f is continuous

$$\lim_{t\to\infty} \eta_i(t) = f\left(\lim_{t\to\infty} x_i(t)\right) = f\left(\frac{1}{N}\sum_{i=1}^{N} g_i(z_i)\right)$$

□

This result has been first proposed in the context of continuous-time consensus problems, where the proof is much more delicate, by Bauso et al. [36] and later generalized by Cortes [37], leading to the so called χ-consensus. Note that the discrete time counterpart here proposesd is less demanding in terms of conditions on the functions f and g.

In general, the key point is to find the appropriate functions g and f which solve the original problem, being the theorem a straightforward application of consensus theory. Many problems, as we will see, cannot be quite written as in Eqn. (2.1), but in the more general form

$$\xi = f\left(\sum_{i\in\mathcal{N}} g(z_i)\right).$$

Clearly, the previous result can still be used if N is known, with the simple change $\eta_i(t) = f(Nx_i(t))$ while the rest of the algorithm is left unchanged. One might therefore wonder if it is possible to compute N distributively, if its value is not known in advance. The following result provides a partial answer to this question.

Proposition 6. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ the connectivity graph associated to a sensor network with N nodes, i.e. $N = |\mathcal{V}|$, and assume that there is a special node $k \in \mathcal{V}$, referred as **graph leader**. Without loss of generality, we set $k = 1$.*

To compute N using consensus, consider a consensus algorithm $\{P(t)\}$, namely a sequence of stochastic matrices compatible with the graph $P(t) \sim \mathcal{G}$ $\forall t$ which solves the (probabilistic) average consensus problem and set:

$$\begin{aligned} x_1(0) &= 1, \\ x_i(0) &= 0, \qquad\qquad \forall i \in V, i \neq 1. \end{aligned}$$

Run then the consensus algorithm with such an initial condition:

$$x_i(t+1) = p_{ii}(t)x_i(t) + \sum_{j\in\mathcal{V}(i)} p_{ij}(t)x_j(t) \qquad \forall i \in \mathcal{V},$$

where $p_{ij}(t) = [P(t)]_{ij}$, and compute at each time instat

$$\eta_i(t) = \frac{1}{x_i(t)}$$

Then we have:

$$\lim_{t\to\infty} \eta_i(t) = N \qquad\qquad \forall i \in \mathcal{V}.$$

Proof.
Analogously to the previous theorem one gets that, since $\{P(t)\}$ solves the average consensus problem,

$$\lim_{t\to\infty} x_i = \frac{1}{N}\sum_{i=1}^{N} x_i(0) = \frac{1}{N} \qquad \forall i \in \mathcal{V}$$

and therefore

$$\lim_{t\to\infty} \eta_i = \lim_{t\to\infty} \frac{1}{x_i} = N$$

□

This proposition shows that it is possible to compute the number of nodes in a graph using an average consensus algorithm as long as a leader is elected. As a consequence, the algorithm is not truly distributed, in the sense that not all the nodes perform the same actions, being the initialization different. Nonetheless, the problem of leader election is a very well studied problem and efficient algorithms exist [38], therefore a fully distributed algorithm would consist in two stages: a leader election stage and an average consensus stage.

In the remaining part of the chapter we show how many important data aggregation problems can be reframed as in Proposition 5.

2.2 Generalized means

As shown in [36], it possible to use Proposition 5 to compute different means besides the standard arithmetic mean. For example the *geometric mean* $\xi = \sqrt[N]{\prod_{i=1}^{N} z_i}$ can be computed as

$$\xi = \sqrt[N]{\prod_{i=1}^{N} z_i} = e^{log\left(\sqrt[N]{\prod_{i=1}^{N} z_i}\right)} = e^{\frac{1}{N}\sum_{i=1}^{N}\log(z_i)},$$

therefore in this case $g_i(\cdot) = \log(\cdot)$ and $f(\cdot) = \exp(\cdot)$.

Similarly, the *harmonic mean* defined as $\xi = \left(\frac{1}{N}\sum_{i=1}^{N}\frac{1}{z_i}\right)^{-1}$ can be computed distributively by setting $g_i(\cdot) = (\cdot)^{-1}$ and $f(\cdot) = (N\cdot)^{-1}$.

As a final example of generalized mean, consider the *mean of order p*, defined as $\xi = \sqrt[p]{\frac{1}{N}\sum_{i=1}^{N} z_i^p}$, where $p = 1, 2, \ldots$, can be obtained by setting $g_i(\cdot) = (\cdot)^p$ and $f(\cdot) = \sqrt[p]{\cdot}$.

2.3 Kalman Filter

Spanos et al. [39] proposed the use of consensus algorithms to implement distributed Kalman filters in sensor networks, achieving the same performance of the centralized Kalman filter.

Let us consider a linear discrete time stochastic system observed by N distinct sensors, modeled as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + w(k) \\ y_i(k) &= C_i x(k) + v_i(k) \end{aligned}$$

where w and v_i are white Gaussian noises with covariance Q and $R_i > 0$, respectively, and $y_i(k)$ is the observation of the i-th node in the network. The centralized Kalman estimate is defined as

$$\hat{x}_c(k|\tau) = \mathbb{E}[x(k) \,|\, y(\tau), y(\tau-1), \ldots, y(0)],$$

where $y(\tau) = [y_1^T(\tau)\, y_2^T(\tau) \,\ldots\, y_N^T(\tau)]^T$ is the complete sensors measurements vector. It is well known that the optimal filter estimate $\hat{x}_c(k|k)$ can be computed using the inverse covariance as:

$$\begin{aligned} S(k|k-1) &= AS(k-1|k-1)A^T + Q \\ \hat{x}_c(k|k-1) &= A\hat{x}(k-1|k-1) \\ S(k|k) &= \left(S^{-1}(k|k-1) + \sum_{i=1}^{N} C_i^T R_i^{-1} C_i \right)^{-1} \\ \hat{x}_c(k|k) &= S(k|k) \left(S^{-1}(k|k-1)\hat{x}(k|k-1) + \sum_{i=1}^{N} C_i^T R_i^{-1} y_i(k) \right). \end{aligned}$$

If each node knows all the model parameters $(A, C_1, \ldots, C_N, Q, R_1, \ldots, R_N)$, then it could in principle compute off-line the matrices $S(t|t-1)$ and $S(t|t)$. Therefore, the only quantity necessary to obtain the centralized Kalman estimate is the sufficient statistic $\xi(k) = \sum_{i=1}^{N} C_i^T R_i^{-1} y_i(k)$. This quantity can be computed as in Proposition 5, by setting

$$z_i = y_i(t), \quad x_i(0) = g_i(z_i) = C_i^T R_i^{-1} z_i \quad \text{and} \quad \eta_i(t) = N x_i(t).$$

It is important to remark that each node needs to know the total number of nodes N in the network. Also note that $\xi(k) = \lim_{t\to 0} \eta_i(t)$, i.e. between two measurements instants k and $k+1$, it is necessary to iterate the consensus algorithm till convergence has been reached. If the number of consensus

iterations t between two consecutive measurements is small, then $\eta_i(t)$ might not coincide with the exact sufficient statistics, therefore this strategy is not guaranteed to achieve the centralized Kalman filter estimate.

Many other strategies to implement distributed Kalman filter have been proposed in literature. Chapter 5 is devoted to the analysis of one of them, optimized for the case in which there is not enough time to achieve consensus between two subsequent measurements. For a detailed discussion on distributed Kalman filters, we refer the reader to Chapter 5 or to the papers [40, 4].

The distributed Kalman filter can be generalized even further to time-varying system parameters and to scenarios where each node knows only the system dynamics and its own parameters, i.e.

$$A(k), Q(k), C_i(k), R_i(k),$$

and the total number of nodes N in the network. In fact in this case, besides the sufficient statistics vector

$$\xi(k) = \sum_{i=1}^{N} C_i^T(k) R_i^{-1}(k) y_i(k),$$

each node needs to compute the matrix

$$\Xi(k) = \sum_{i=1}^{N} C_i^T(k) R_i^{-1}(k) C_i(k)$$

necessary to recover the matrix $S(k|k)$. It is not difficult to see that by setting

$$x_i(0) = C_i^T(k) R_i^{-1}(k) C_i(k)$$

and

$$\eta_i(t) = N x_i(t),$$

where with a little abuse of notation x_i has to be intended as the vectorization of the matrix, we have

$$\lim_{t \to 0} \eta_i(t) = \Xi(k).$$

2.4 Maximum Likelihood Parameter Estimation

Another interesting estimation problem that can be formulated as an average consensus problem, at least approximately, is the maximum likelihood parameter estimation (MLPE). In many applications where a probabilistic model is available, it is desired to compute the parameters that maximize the likelihood of given a set of observations. Let $z_i, i = 1, \ldots, N$ be the observation available to node i of a network. We assume that these data are sampled from independent probability density $h_i(\cdot|\theta)$, parameterized by the same unknown parameter vector θ. The likelihood is defined as $\Gamma(\theta) = h(z_1, \ldots, z_N|\theta) = \prod_{i=1}^{N} h_i(z_i|\theta)$ and the MLPE is defined as

$$\hat{\theta}_{ML} = \arg\max_{\theta} \Gamma(\theta) = \arg\max_{\theta} \log \Gamma(\theta) = \arg\max_{\theta} \sum_{i=1}^{N} \log h_i(z_i|\theta).$$

A common practice to find at least a local maximum of the previous function is to use the centralized gradient descent:

$$\theta_c(k+1) = \theta_c(k) + \epsilon \xi_c(k), \quad \xi_c(k) = \left.\frac{\partial \log \Gamma}{\partial \theta}\right|_{\theta=\theta(k)} = \sum_{i=1}^{N} \left.\frac{\partial \log h_i(z_i|\theta)}{\partial \theta}\right|_{\theta=\theta(k)}$$

where $\epsilon > 0$ is a small positive number. The gradient can be computed in a distributed fashion using Proposition 5, as follows:

$$\begin{aligned} \theta_i(k+1) &= \theta_i(k) + \epsilon \xi_i(k), \quad i = 1, \ldots, N \\ \xi_i(k) &= \lim_{t \to \infty} \eta_i(t) \end{aligned}$$

where

$$x_i(0) = \left.\frac{\partial \log h_i(z_i|\theta)}{\partial \theta}\right|_{\theta=\theta_i(k)} \qquad \text{and} \qquad \eta_i(t) = N x_i(t).$$

Note that knowing N is not so relevant in this context, since what it is important is to compute the direction of the gradient. It is sufficient to have a rough idea of the value of N to adjusted the step length by properly tuning the parameter ϵ, therefore we could simply use

$$\eta_i(t) = x_i(t).$$

As for the distributed Kalman filter, this is a two-stage strategy. In the first stage a local update of the parameter $\theta_i(k)$ is performed at time k in each node, and in the second stage, before the next update at time $k+1$, the

gradient of the centralized log-likelihood $\log \Gamma(\theta)$ is computed by running t iterations of the consensus algorithm. Also in this case the equivalence between the centralized algorithm and the decentralized consensus-based algorithm is based on an asymptotic result, therefore it is not clear what is the behavior of the algorithm for finite number of iterations t. We are not aware of any work in this direction.

2.5 Least Square Parameter Estimation

We now introduce the problem of Least Square Parameter Estimation (LSPE). This problem arises when we have a data set $\mathcal{D} = \{(a_m, b_m), m = 1, \dots, M\}$ where $a_m \in \mathbb{R}^\ell$ and $b_m \in \mathbb{R}$. The data set is generated according to the model $a_m^T \theta = b_m + v_m$, where $\theta \in \mathbb{R}^\ell$ is a parameter vector to be estimated and $v_m \in \mathbb{R}$ is an unknown error. Let us define the matrix $A \in \mathbb{R}^{M \times \ell}, A = [a_1 \dots a_M]^T$ and the vectors $b, v \in \mathbb{R}^M, b = [b_1 \dots b_M]^T, v = [v_1 \dots v_M]^T$. The least square estimation of the parameter θ is defined as follows:

$$\hat{\theta}_{LS} = \arg\min_\theta \|v\| = \arg\min_\theta \|A\theta - b\| = A^\dagger b$$

where the symbol $\dagger$ represents the pseudo-inverse of a matrix, namely, if AA^T is invertible, $A^\dagger = (A^T A)^{-1} A^T$. We now present a proposition showing how the centralized least square parameter estimation can be performed over a sensor network.

Proposition 7. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the connectivity graph associated to a sensor network with N nodes, i.e. $N = |\mathcal{V}|$. Let us denote with $\mathcal{D}(i) = \{(a_m, b_m)\}$ the partition of the whole data set $\mathcal{D}$ available to i-th node. We suppose that*

$$\mathcal{D}(i) \cap \mathcal{D}(j) = \emptyset, \quad i \neq j \qquad , \qquad \cup_{i \in \mathcal{V}} \mathcal{D}(i) = \mathcal{D}.$$

Let us moreover define $|\mathcal{D}(i)| = M_i$ and $|\mathcal{D}| = M = \sum_{i \in \mathcal{V}} M_i$.

Consider a consensus algorithm $\{P(t)\}$, namely a sequence of stochastic matrices compatible with the graph $P(t) \sim \mathcal{G}$ $\forall t$ which solves the (probabilistic) average consensus problem and set:

$$\begin{aligned} x_i^A(0) &= \sum_{m \in \mathcal{D}(i)} a_m a_m^T, \\ x_i^b(0) &= \sum_{m \in \mathcal{D}(i)} a_m b_m \qquad\qquad \forall i \in \mathcal{V} \end{aligned}$$

Run then two consensus algorithms, one having as in initial condition $x_i^A(0)$ and the other having as initial condition $x_i^b(0)$:

$$x_i^k(t+1) = p_{ii}(t)x_i^k(t) + \sum_{j\in\mathcal{V}(i)} p_{ij}(t)x_j^k(t), \qquad k = A, b$$

where $p_{ij}(t) = [P(t)]_{ij}$ and compute at each time instant

$$\eta_i(t) = \left(x_i^A(t)\right)^\dagger x_i^b(t)$$

One has that:

$$\lim_{t\to\infty} \eta_i(t) = \hat{\theta}_{LS}, \qquad \forall i \in \mathcal{V}$$

Proof.
The proof is very similar in to the one of Proposition 5. The key point is to note thau the matrix

$$S = A^T A = \sum_{i=1}^{M} a_i a_i^T$$

and the vector

$$q = A^T b = \sum_{i=1}^{M} a_i b_i,$$

therefore $\hat{\theta}_{LS} = S^\dagger q$. By construction we have that

$$\lim_{t\to\infty} x_i^A(t) = \frac{1}{N}\sum_{i=1}^{N} x_i^A(0) = \frac{1}{N}\sum_{i=1}^{M} a_i a_i^T = \frac{1}{N}S$$

and similarly

$$\lim_{t\to\infty} x_i^b(t) = \frac{1}{N}\sum_{i=1}^{N} x_i^b(0) = \frac{1}{N}\sum_{i=1}^{M} a_i b_i = \frac{1}{N}q.$$

By continuity

$$\lim_{t\to\infty} \eta_i(t) = (\frac{1}{N}S)^\dagger \frac{1}{N}q = S^\dagger q = \hat{\theta}_{LS}.$$

□

This proposition shows that LSPE can be computed as the solution of a distributed algorithm which does not require the knowledge of the total number of nodes N or the total number of data M available. Moreover, the data can be arbitrarily partitioned among the nodes. Since the matrix $S = A^T A$

is symmetric, it is not necessary to compute all its ℓ^2 entries, therefore the x_i^A can be reduced to a vector of size $(\ell^2+\ell)/2$. Nonetheless the complexity in terms of communication, i.e. the dimension of the vector of parameters to be averaged, is $O(\ell^2)$ which can be impractical if the dimension ℓ of the unknown parameter θ is large.

We will show in the next section that there are strategies that trade-off accuracy in the estimation of $\hat{\theta}_{LS}$ for a decrease in communication complexity to $O(\ell)$.

2.6 Final Comments and Open Issues

In this chapter we considered one of the key problems in distributed estimation and more generally of multi–agent systems: the problem of fusing sensor data to compute global quantities.

A special class of global quantities has been introduced: those quantities can be computed in a distributed manner using average consensus algorithms.

We have then shown that many relevant problems, such as Kalman filtering, maximum likelihood and least squares parameters estimation, fit in this class. This fact makes the consensus algorithms one of the basic tools for distributed data fusion.

The equivalence between the centralized algorithm and the decentralized consensus-based algorithm is based on an asymptotic result. It is not clear what is the behavior of these algorithms for finite number of iterations. This certainly relevant issue has not yet been addressed, at the best of our knowledges.

The fact that consensus has to be reached for the algorithm to work, has its most relevant impact in the Kalman filer case. Indeed, in this example the algorithm prescribes to perform a whole consensus algorithm run during two subsequent measurements. Therefore, the time available to the consensus algorithm depends on the application and if it is not sufficient to reach consensus, then the strategy is not guaranteed to achieve the centralized Kalman filter estimate.

For this reason, when the time between two subsequent measurements is small, other strategies might be preferable. One of the possible alternative strategies is presented and analyzed in Chapter 5: it minimizes the steady state estimation variance assuming that only a given amount m of consensus iterations is exploitable.

3

Distributed Sensor Calibration for Localization and Tracking

In this chapter we focus on wireless sensor networks, WSNs from now on, and we address some of the modeling and algorithmic aspects of one their popular application, namely localization and target tracking, which has been widely studied in the last few years.

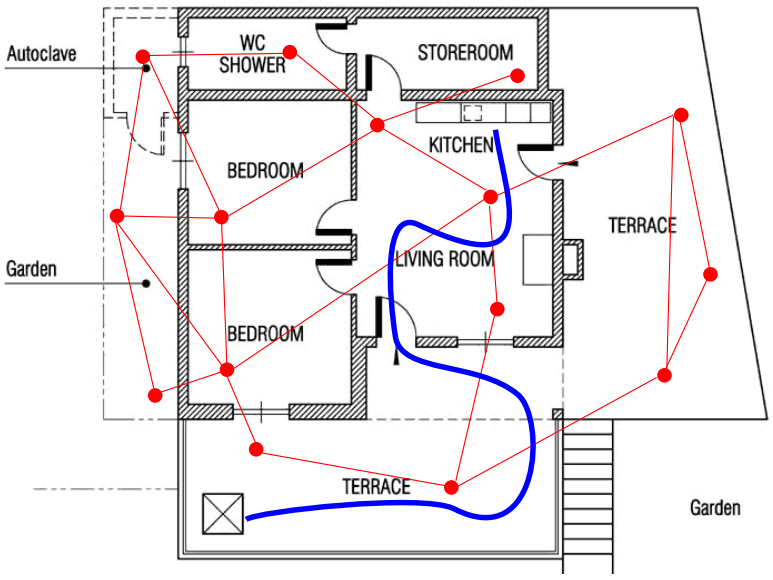
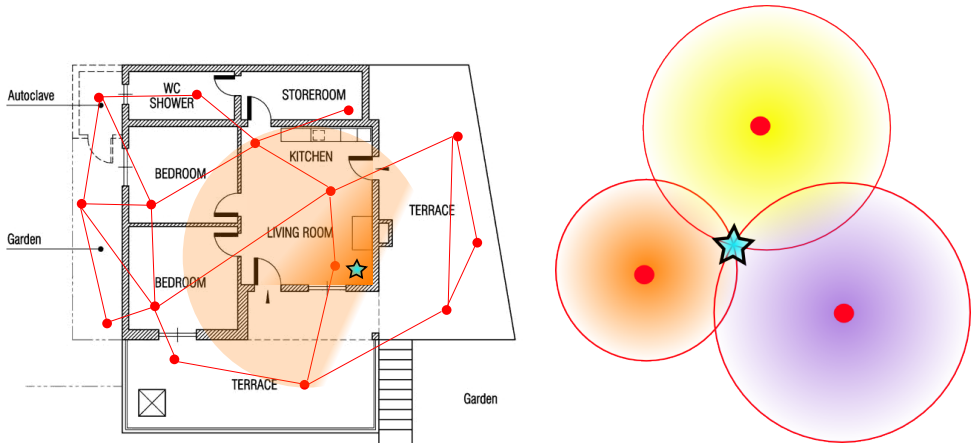


Figure 3.1: Pictorial representation of one popular application for WSNs: localization and target tracking of a moving object.



(a) Map based approach (b) Range based approach

Figure 3.2: The two most common techniques used for localization and tracking: map based and range based approach.

A commonly used method to perform the localization and tracking task is to equip network nodes with a sensor that measures the time-of-flight differences between an acoustic wave and an electromagnetic wave emitted by the moving object. From this quantity a very accurate estimate of the distance between the moving object and the sensor can be obtained. Unfortunately, those sensors increase significantly the overall network cost. It is currently under study an alternative to such a solution. The idea is to use the radio signal strength, that, being a function of the locations of transmitter and receiver, can be used to estimate their relative positions. In fact, the wireless radio in each node of the WSN can be used not only to communicate but also to measure the radio signal strength associated with the received packet, making available a zero-cost sensor that can be used for localization purposes.

There are two main approaches to target tracking: map-based and range-based. In the map-based approach the position of the moving target is obtained by finding the most likely location which matches the recorded signal strengths, based on previously learned maps [41, 42]. This strategy can be a good solution but it requires an extensive work to learn the maps. Differently, the range-based algorithms first try to estimate relative distances based on a simple model of the wireless channel and then they estimate the positions by triangulation, similarly to the GPS system where the satellites correspond to the static nodes of the WSN [43]. This approach requires a higher nodes density than the map-based one, but it does not require extensive learning phase. We focus on this last approach.

Unfortunately, the radio signal strength is a rather poor indicator of the distance, being very noisy and influenced by many other effects, such as presence of moving obstacles, reflections, and so on. Moreover, the radio signal strength indicator provided by the radio chips of the sensor nodes is affected by uncalibrated offsets in the receiving nodes. As a consequence, the estimated distance can be constantly biased in some nodes, thus degrading tracking performance. Therefore, it is necessary to devise some strategies to compensate these offsets [44].

Furthermore most of the previous work on range-based tracking proposed in the literature focuses on triangulation algorithms where the wireless channel model parameters are assumed to be known or are identified off-line by collecting all data in some centralized location [45]. Unfortunately, these parameters are strongly dependent on the environment [46], in particular indoor, therefore it is desirable to identify them in-situ, possibly using distributed algorithms suitable for the WSN node computational resources. Moreover, the same environment can present a hourly or daily variation of these parameters due to the periodic presence of people populating the indoor spaces.

This facts make localization based on radio signal strength an issue that is still open and that calls for a way to cope with the above reported effects.

The main contribution presented in this chapter goes in this direction. We propose in fact the use of consensus algorithms for automatically calibrating the sensors, without the use of a reference node, and for least–square–estimating the optimal channel parameters with a distributed algorithm.

Although we present these algorithms applied to localization and tracking for WSNs, they are very general since they can be applied in any context where there is a need to calibrate sensors and to solve a global least square identification problem.

Another contribution reported in this chapter is to mathematically model the wireless channel and the communication protocols of typical WSN based on experimental data, which is an aspect that it is often overlooked, leading to models which are unrealistic. For example, due to packet loss or time synchronization, it is rather problematic in WSNs to enforce convergence to the mean of initial condition, i.e. to enforce average consensus. Therefore, in our work we posed particular care in exploring the trade offs between perfect average consensus and use of simpler algorithms.

Chapter organization

The chapter is organized as follows. In Section 3.1 we provide a mathematical model for the wireless channel in WSNs. In Section 3.2 we describe the

experimental testbed used to collect data and, based on these data, we find the numerical values for the model parameters given in the previous Section. This allows us to show that various terms in the previously presented model are negligible and therefore to propose an experimentally–validated simplified model. In Section 3.3 we propose a consensus-based strategy for calibrating sensors with unknown measurement offset readings and we apply it to experimental data. In Section 2.5 we use the previously introduced consensus-based least square algorithm for identifying the wireless channel parameters under different communication strategies and we highlight trade-offs between performance, speed of convergence and computation complexity, based on experimental data. Finally, in Section 3.5 we summarize the results presented in this chapter and discuss some open questions.

3.1 Wireless Channel Model

Here we model the behavior of the wireless channel between two nodes in terms of received power P_{rx} (in dBm). We start by considering the most general model and then we highlight which are the most relevant elements based on experimental data. The Radio Signal Strength Indicator (RSSI) measured by a generic node i after having successfully received a packet, sent by the generic node j, can be modeled in the most general form as:

$$P_{rx}^{ij} = f\left(P_{tx}^{j}, \underline{x}_i, \underline{x}_j, i, j, t\right)$$

where P_{tx}^{j} (in dBm) is the nominal transmitted power, i and j are the IDs representing the receiver and the transmitter node, respectively, $\underline{x}_i, \underline{x}_j \in \mathbb{R}^3$ are their spatial positions and t is the time when the communication occurs. The previous equation can be decomposed into simpler elements which takes into accounts different effects. Combining the models of each element, described in [47] and [46], and adding parts due to offsets in the RSSI measurements and in the power transmission, we obtain the following model:

$$P_{rx}^{ij} = P_{tx}^{j} + r_j + f_{pl}(\|\underline{x}_i - \underline{x}_j\|) + f_{sf}(\underline{x}_i, \underline{x}_j) + f_a(\underline{x}_i, \underline{x}_j) + v_{ff}(t) + o_i \quad (3.1)$$

where:

- P_{tx}^{j} is the nominal transmitted power and r_j is the *transmission offset* between the nominal and the effectively transmitted power. This factor is due to fabrication mismatches and it is assumed to be constant in time;

- $f_{pl}(\cdot)$ represents the *Path Loss* effect, modeled as (see [46]):

$$f_{pl}(d) = \beta - 10\gamma \log_{10}(d) \tag{3.2}$$

  where $d = \|\underline{x}_i - \underline{x}_j\|$ is the euclidean distance between the nodes i and j, β represents the radio receiver gain at a nominal distance of $d = 1m$, and γ is the loss factor (in an ideal outdoor setting $\gamma \approx 2$). The parameters β and γ are in general unknown since they depend on the specific environment where the WSN is placed;

- $f_{sf}(\cdot)$ takes into account the *Shadow Fading* and other slow fading components. It is assumed (see [47]) to be symmetric (i.e. $f_{sf}(\underline{x}_i, \underline{x}_j) = f_{sf}(\underline{x}_j, \underline{x}_i)$) and Gaussian, with a spatial correlation dependent on the difference between the distances of the various points. More precisely, $\mathbb{E}_{\underline{x}}[f_{sf}(\underline{x}_i, \underline{x}_j)] = 0$ and $\mathbb{E}_{\underline{x}}[(f_{sf}(\underline{x}_i, \underline{x}_j))^2] = \sigma_{sf}^2$ are the spatial mean and variance where the expectation is performed w.r.t. to the random node positions. Moreover, let $\underline{x}_i, \underline{x}_j^a$ and $\underline{x}_i, \underline{x}_j^b$ two different configurations s.t. $\delta = \|\underline{x}_j^a - \underline{x}_j^b\|$, then the spatial correlation is:

$$\mathbb{E}_{\underline{x}}\left[\left(f_{sf}(\underline{x}_i, \underline{x}_j^a)\right)\left(f_{sf}(\underline{x}_i, \underline{x}_j^b)\right)\right] = \sigma_{sf}^2 \rho_D^{\delta/D}$$

  where ρ_D is a parameter and D is the typical correlation distance. Note that the expected value of f_{sf} is assumed to be zero;

- $f_a(\cdot)$ represents the *channel asymmetry* factor. It is due to non symmetric reflections and we model it as a Gaussian r.v. with zero-mean and covariance $\mathbb{E}_{\underline{x}}[f_a^2(\underline{x}_i, \underline{x}_j)] = \sigma_a^2$;

- $v_{ff}(\cdot)$ represents the *fast fading* component that can be modeled (see [46]) as a white temporal noise with covariance $\mathbb{E}_t[v_{ff}^2(t)] = \sigma_{ff}^2$;

- $o_i(\cdot)$ represents the *offset that affects the measured received strength* of the receiving node due to fabrication mismatches in the radio chip. For example, in the case of the nodes used in our experimental testbed the RSSI sensor has a tolerance of ± 6 dB (see [48]).

Eqn. (3.1) is a general model for the wireless channel, in which parameters depend on the physical environment where the WSN is placed and on the sensors under consideration. It is important to remark that these parameters are not known in advance but they need to be estimated on-site. This is the objective of the rest of the paper. In the next section we describe the experimental testbed used to collect experimental data, we determine some of the numerical values for the terms in Eqn. (3.1) and we show that some of them are negligible. Then, in Section 3.3 we estimate the offsets o_i, and in Section 3.4 we identify β and γ.

3.2 Experimental Testbed and Model Validation

The experimental data used in the simulations consist in a series of measurements relative to packet transmissions and receptions performed by a net of 25 Tmote-Sky nodes [49], equipped with the Chipcon CC2420 RF Transceiver [48] and powered by alkaline batteries. These nodes were randomly placed inside a single conference room of $15m \times 10m$, depicted in Figure 3.3, at about $50cm$ from the ground. The relative position of the nodes is shown in Figure 3.4.

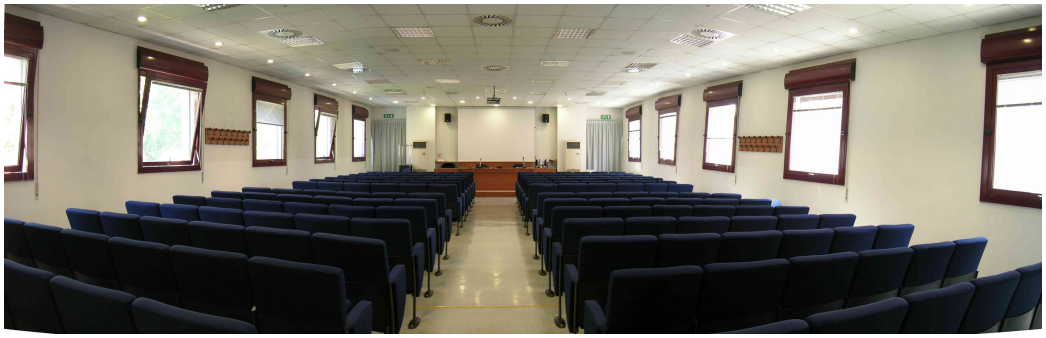


Figure 3.3: Picture of the experimental testbed room: Aula Magna "A. Lepschy", Dept. of Information Engineering, University of Padova.

Each node implemented the randomized broadcast communication using the same transmission power P_{tx} and intercommunication interval $\tau = 15s$ so that the event of a packet collision was negligible. Each node sent a fixed number of packets, $M = 500$, each one including the sender node ID, and also stored a table with the total number of messages received from their neighbors and the corresponding RSSI measures P_{rx}^{ij}. [1]

These tables were then collected for off-line data processing. In particular, from these data we constructed the connectivity graph. Given the short distance among nodes, each node received at least one packet from any other node, however the empirical packet reception probability was different. In fact, the c-connectivity graph $\mathcal{G}_c$ obtained for $c = 0.75$ (i.e. removing poor links with showed an empirical packet loss probability greater than 25%) is not the complete graph, even if it is still connected, as shown in Figure 3.4.

In the following we explain how it is possible to estimate the various parameters of the wireless channel model (3.1) using the various $P_{rx}^{ij}(t)$ collected

[1] We would like to thank to G. Zanca and F. Zorzi, that collected all the measurement used in this chapter. See [50].

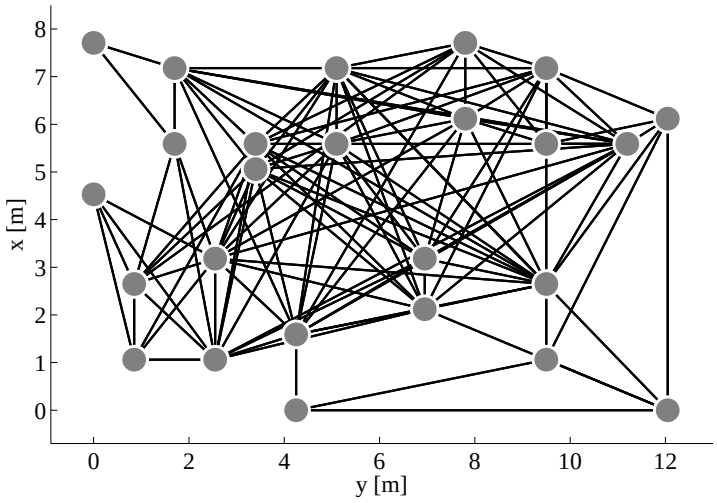


Figure 3.4: Network topology and node displacement of experimental testbed. Only edges with empirical packet loss smaller than 25% are displayed.

from the nodes.

The transmission power offsets r_j of Eqn. (3.1) can be directly measured substituting the antenna of the nodes with a probe connected to a power measurer. Measurements made on the set of the nodes used for the experimental data showed that these offsets are negligible (see [50]), so in the following we will ignore them, i.e. we set $r_i = 0$, $\forall i$.

Then, for every link $(i, j) \in \mathcal{E}$ in the connectivity graph, we compute the empirical mean of the received power

$$\bar{P}_{rx}^{ij} = \frac{1}{M_{ij}} \sum_t P_{rx}^{ij}(t)$$

and the empirical variance

$$(\hat{\sigma}_{ff}^{ij})^2 = \frac{1}{M_{ij} - 1} \sum_t (P_{rx}^{ij}(t) - \bar{P}_{rx}^{ij})^2,$$

where M_{ij} is the total number of messages received. The empirical variance around each link is due to fast fading, therefore, a good estimate for the fast fading variance is:

$$\sigma_{ff}^2 = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} (\sigma_{ff}^{ij})^2.$$

The measurements $\bar{P}_{rx}^{ij}$ include the effects of path loss, shadow-fading, channel asymmetry and reception offsets. We can try to isolate first the contribution of the channel asymmetry and reception offset by noting that the path loss and the shadow fading are symmetric, therefore the quantity

$$\Delta\bar{P}_{rx}^{ij} = \bar{P}_{rx}^{ij} - \bar{P}_{rx}^{ji} = f_a^{ij} - f_a^{ji} + o_i - o_j,$$

where for ease of notation $f_a^{ij} = f_a(\underline{x}_i, \underline{x}_j)$, depends only on the asymmetry component and on the offsets. We can also remove the effects of the offsets considering a closed path over three nodes i, j, k and noting that

$$\Delta\bar{P}_{rx}^{ijk} = \Delta\bar{P}_{rx}^{ij} + \Delta\bar{P}_{rx}^{jk} + \Delta\bar{P}_{rx}^{ki} = f_a^{ij} - f_a^{ji} + f_a^{jk} - f_a^{kj} + f_a^{ki} - f_a^{ik}.$$

We experimentally observed that $\Delta\bar{P}_{rx}^{ijk}$ has approximately zero-mean over the set of all the independent feasible cycles (i, j, k), set that we denote with $\mathcal{C}$. Since the nodes are sufficiently far from each other and we have experimentally observed that the shadow fading correlation distance $D \approx 10cm$, all f_a^{ij} can be considered uncorrelated, therefore we can compute the covariance of the channel asymmetry as:

$$\sigma_a^2 = \frac{1}{6\,|\mathcal{C}|} \sum_{(i,j,k)\in\mathcal{C}} \left(\Delta\bar{P}_{rx}^{ijk}\right)^2.$$

If we also assume independence between channel asymmetry components f_a^{ij} and the offsets o_i, we can estimate the offset variance σ_o^2 from the following formula:

$$2\sigma_o^2 + 2\sigma_a^2 = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \left(\Delta\bar{P}_{rx}^{ij}\right)^2.$$

Finally, we can estimate the parameters $\theta = [\beta \;\; \gamma]^T$ of the path loss channel. As it will be shown in the next section, it is possible to calibrate sensors by adding a compensating offset $\hat{o}_i$ such that $o_i + \hat{o}_i = \alpha$ for all nodes. Averaging all sensor readings received from the same node removes the effect of fast-fading, therefore the calibrated average received power $\hat{P}_{rx}^{ij} = \bar{P}_{rx}^{ij} + \hat{o}_i$ is given by:

$$\hat{P}_{rx}^{ij} = P_{tx} + \beta - 10\gamma \log\left(d_{ij}\right) + f_{sf}^{ij} + f_a^{ij} + \alpha$$

where $f_{sf}^{ij} = f_{sf}(\underline{x}_i, \underline{x}_j)$. Since β needs to be estimated and α is constant, we can assume w.l.o.g. that $\alpha = 0$, since its contribution will be included in the estimated β. Shadow fading f_{sf}^{ij} and channel asymmetry f_a^{ij} are unknown but they can be assumed to be independent zero-mean disturbances, therefore it is possible to find the best mean square estimate of the unknown parameters

as $\hat{\theta}_{LS} = (A^T A)^{-1} A^T b$, where $A = [a_1 \ \dots \ a_M]^T$, $b = [b_1 \ \dots \ b_M]$, and $M = |\mathcal{E}|$. The generic elements of matrix A and vector b are $a_m = [1 \quad -10\log(d_{ij})]^T$ and $b_m = (\hat{P}_{rx}^{ij} - P_{tx})$, where $d_{ij} = \|\underline{x}_i - \underline{x}_j\|$ and $\hat{P}_{rx}^{ij}$ are known. Figure 3.5 shows the identified path-loss model and all collected pairs $(\hat{P}_{rx}^{ij}, d_{ij})$. The residues obtained from the path-loss model correspond to the variance due to the shadow fading and channel asymmetry, i.e.:

$$\sigma_a^2 + \sigma_{sf}^2 = \frac{1}{|\mathcal{E}|} \left\| A\hat{\theta}_{LS} - b \right\|^2 .$$

Table 3.1 summarizes the estimated parameters of the model (3.1) based on the experimental data collected. Note that the terms due to the asymmetry in the channel, f_a^{ij}, can be safely neglected when compared to the slow fading terms, f_{sf}^{ij}, i.e. the wireless channel is indeed symmetric.

β $[dBm]$	γ $[dBm]$	σ_{sf} $[dBm]$	σ_a $[dBm]$	σ_{ff} $[dBm]$	σ_o $[dBm]$	r_i $[dBm]$
-45.7	1.76	3.78	0.16	1.31	1.01	≈ 0

Table 3.1: Results of the estimation of the channels parameters of the model (3.1) via the centralized estimation strategies presented in Section 3.2.

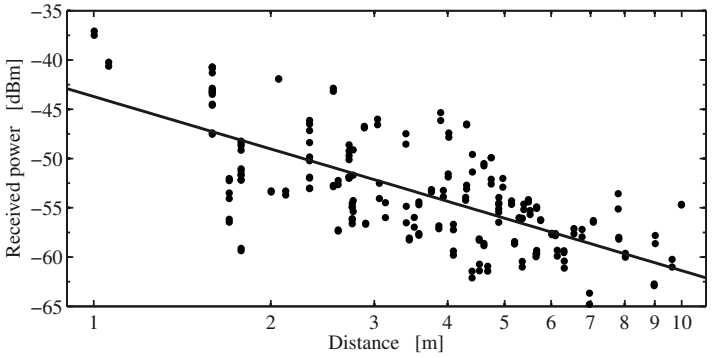


Figure 3.5: Estimated path-loss model for the wireless channel of the experimental environment using the standard centralized mean square estimate. The continuous line represents the path-loss function, while the dots are the measures experimentally collected.

3.3 Consensus-Based Sensors Calibration

Experimental evidence indicates that sensor offsets o_i in the nodes are not negligible and can be substantially large for some nodes (up to $6dB$). The

effect of this offset is to bias the estimate of the distance between two nodes, which is particularly harmful in tracking application. In fact, if one node has a high offset o_i, then its estimated distance from a moving node is smaller than the true distance. Since unknown location of a moving target is obtained, similarly to the GPS, by triangulating its position from three or more static nodes whose position is known, the estimated position will be closer to the node with high offset o_i than it should be. This is particularly clear in Figure 3.6, which reports a tracking experiment where the moving node to be tracked is following a straight line (the basketball court centerline) between two rows of nodes of a WSN. However, its estimated trajectory is not straight but it is bent to the left (left panel). When the two central nodes on one side are swapped with the other side, the estimated trajectory is now bent to the right, thus clearly showing a problem due to uncalibrated offsets. Here we present a fully distributed and simple strategy which aims at estimating and removing the offsets o_i from each node, and we show its benefits on experimental data.

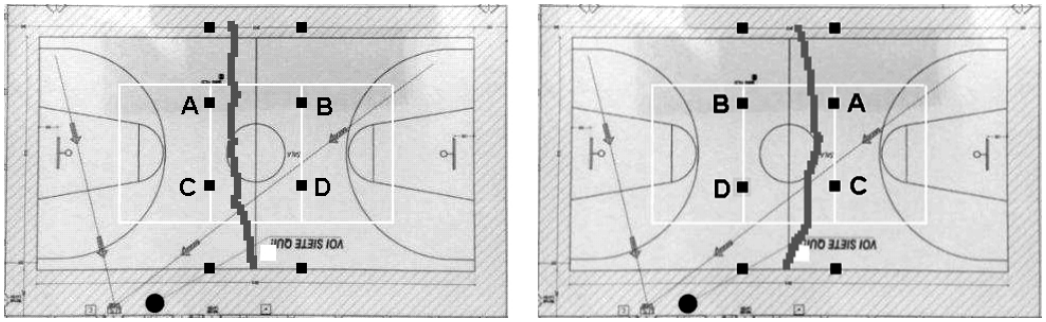


Figure 3.6: Experiment inside a basketball court showing the effects of reception offsets in WSN tracking when nodes are swapped. True trajectory in both panels is the court cenerline. Courtesy of ST Microelectronics [51].

3.3.1 Offset calibration algorithm

Ideally, we would like to add to the reading of received power a compensation offset $\hat{o}_i$ such that $o_i - \hat{o}_i = 0$, and then use the compensated received power $\widehat{P}_{rx}^{ij} = P_{rx}^{ij} - \hat{o}_i$ to estimate the relative distance. However, we do not have the possibility to directly measure o_i of each node, nonetheless we would like to, at least partially, compensate it. More precisely, we would like to have, for all nodes,

$$o_i - \hat{o}_i = \alpha \quad \forall i \in \mathcal{V}, \qquad \alpha \approx 0.$$

If $\alpha \neq 0$ such strategy does not compensate the offset, but at least all nodes either underestimate or overestimate the relative distance similarly, therefore after a GPS-like triangulation stage these errors should partially cancel out. We now show how this strategy can be casted as a consensus problem. Let us consider a static WSN where the nodes are at fixed positions and transmit at the same power P_{tx}. Let us introduce the variable y_{ij}:

$$y_{ij} = f_{ij} + o_i.$$

Note that y_{ij} is closely approximated by the average received signal strength by node i from node j:

$$\begin{aligned} \bar{P}_{rx}^{ij} &= \frac{1}{T}\sum_{t=1}^{T} P_{rx}^{ij}\left(P_{tx}, \underline{x}_i, \underline{x}_j, i, j, t\right) = \\ &= f_{ij} + o_i + f_{ij}^a + r_j + \frac{1}{T}\sum_{t=1}^{T} v_{ff}(t) \approx f_{ij} + o_i = y_{ij} \end{aligned} \tag{3.3}$$

where P_{rx}^{ij} is modeled as in Eqn. (3.1), $f_{ij} = P_{tx} + f_{pl}(\|\underline{x}_j - \underline{x}_i\|) + f_{sf}(\underline{x}_j, \underline{x}_i)$, and $f_{ij}^a = f_a(\underline{x}_i, \underline{x}_j)$. The approximation is based on parameters in Table 3.1 which imply that

$$|f_{ij}^a + r_j + \frac{1}{T}\sum_{t=1}^{T} v_{ff}(t)| \ll |o_i|$$

for T sufficiently large, $v_{ff}(t)$ being white noise. Note that f_{ij} is symmetric, i.e. $f_{ij} = f_{ji}$. The next theorem shows how the problem of compensating the offset o_i can be casted as a consensus problem:

Theorem 8. *Let us consider the connectivity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of a WSN, and let $\{Q(t)\} \sim \mathcal{G}$ be a sequence of stochastic matrices that solves the (probabilistic) consensus problem. Assume that $y_{ij} = f_{ij} + o_i$ where $f_{ij} = f_{ji}$. Consider the following algorithm:*

$$\begin{aligned} \hat{o}_i(0) &= 0, \qquad \forall i \in \mathcal{V} \\ \hat{o}_i(t+1) &= \hat{o}_i(t) + \sum_{j \in \mathcal{V}(i)} q_{ij}(t)\big(y_{ij} - y_{ji} + \hat{o}_j(t) - \hat{o}_i(t)\big) \end{aligned} \tag{3.4}$$

where $q_{ij}(t) = [Q(t)]_{ij}$. Then

$$\lim_{t\to\infty} o_i - \hat{o}_i(t) = \alpha \qquad \forall i \in \mathcal{V}$$

where $\alpha \in [\min_i(o_i), \max_i(o_i)]$.
If in addition $Q(t)$ *is doubly stochastic* $\forall t$, *then*

$$\alpha = \frac{1}{N} \sum_{i \in \mathcal{V}} o_i.$$

Proof.
Let us define the new variables $x_i(t) = \hat{o}_i(t) - o_i$. From this definition it follows that $x_i(0) = \hat{o}_i(0) - o_i = -o_i$. Moreover, Eqn. (3.4) can be rewritten as follows:

$$\begin{aligned}
\hat{o}_i(t+1) - o_i &= \hat{o}_i(t) - o_i + \sum_{j \in \mathcal{V}(i)} q_{ij}(t)\big(f_{ij} + o_i - f_{ji} - o_j + \hat{o}_j(t) - \hat{o}_i(t)\big) \\
x_i(t+1) &= x_i(t) + \sum_{j \in \mathcal{V}(i)} q_{ij}(t)\big(x_j(t) - x_i(t)\big) \\
&= \Big(1 - \sum_{j \in \mathcal{V}(i)} q_{ij}(t)\Big) x_i(t) + \sum_{j \in \mathcal{V}(i)} q_{ij}(t) x_j(t) \\
&= q_{ii}(t) x_i(t) + \sum_{j \in \mathcal{V}(i)} q_{ij}(t) x_j(t)
\end{aligned}$$

The last equation can be written in compact form as $x(t+1) = Q(t)x(t)$. Since $Q(t)$ solves the (probabilistic) consensus problem, then

$$\lim_{t \to \infty} x_i(t) = \alpha \qquad \forall i \in \mathcal{V}.$$

The claim that $\alpha \in [\min_i(o_i), \max_i(o_i)]$ follows from the property that if Q is a stochastic matrix, then $\max(Qx) \leq \max_i(x)$ and $\min(Qx) \geq \min(x)$ [52]. □

The previous theorem indicates how we can compensate the offsets o_i without knowing their values. Also, it is not necessary to know the exact value of f_{ij} since, being symmetric, it cancels out.

In practice the assumption y_{ij} is not available but, since $y_{ij} \approx \bar{P}_{rx}^{ij}$, we can use $\bar{P}_{rx}^{ij}$ in place of y_{ij} . As we will show shortly, the fact that $\bar{P}_{rx}^{ij} = f_{ij} + o_i$ is not exact leads to an oscillating steady state behavior in the consensus algorithm.

We might wonder whether there is an appropriate choice of $Q(t)$ to have $\alpha \approx 0$, which is the ideal solution. We can argue that the offsets o_i of the radio chips are on average 0, have some dispersion due to imperfect fabrication and are independent, i.e.

$$\mathbb{E}[o_i] = \mu_o = 0, \qquad \mathbb{E}[o_i^2] = \sigma_o^2, \qquad \text{and} \qquad \mathbb{E}[o_i o_j] = \mathbb{E}[o_i]\mathbb{E}[o_j] = 0.$$

It is well known that the best estimate of the mean μ_o given a set of offsets is

$$\mathbb{E}[\mu_o \,|\, o_1, \ldots, o_N] = \frac{1}{N} \sum_{i \in \mathcal{V}} o_i = \alpha^*$$

which has the property that

$$\mathbb{E}[\alpha^*] = \mu_o = 0$$

and

$$\mathbb{E}[(\alpha^*)^2] = \frac{\sigma_o^2}{N},$$

i.e. the average consensus is the strategy for which α is closer to zero in mean square sense and its error becomes smaller and smaller as the number of nodes N increases.

Although the best choice for the compensation of offsets o_i is to choose doubly stochastic matrices $Q(t)$, this can be difficult to be enforced in a WSN, since it requires synchronization among the nodes and compensation for packet loss. However, it is not necessary to enforce the average consensus since the nonzero offset α reached after the calibration phase is completely absorbed during the identification of the path loss model parameter β.

3.3.2 Simulations based on experimental data

The proposed algorithm for distributed offset calibration has been tested off–line on the same set of data collected during the experimental setup described in Section 3.2. Here we considered the c-connectivity graph $\mathcal{G}_c$ with $c = 0.1$, i.e. we considered all links which received at least 10% of the packets. Differently from the graph with $c = 0.75$ shown in Figure 3.4, the resulting graph with $c = 0.1$ reported in Figure 3.7 is complete, i.e. all edges exist. The set of all the edges has been divided into two subsets: the first subset of edges (60% of the total edges, in black in Figure 3.7) has been used for the estimation of the node offsets. Therefore the proposed distributed sensor calibration algorithm has been executed on the data collected on these edges. In particular, the calibration algorithm was set with $y_{ij} = \bar{P}_{rx}^{ij}$ corresponding to these edges. The second subset (40% of the total edges, in grey in Figure 3.7) has been used in a second stage for validation purposes: we evaluated the asymmetric difference $(\bar{P}^{ij} - \hat{o}_i) - (\bar{P}^{ji} - \hat{o}_j)$ on the data collected on this subset. This approach allows us to both evaluate the effect of the offset removal in a rigorous way, and to validate at the same time the model we proposed.

We simulated the randomized broadcast consensus described in Section 1.2 on the graph $\mathcal{G}_c$ using the experimental data and including i.i.d. packet loss

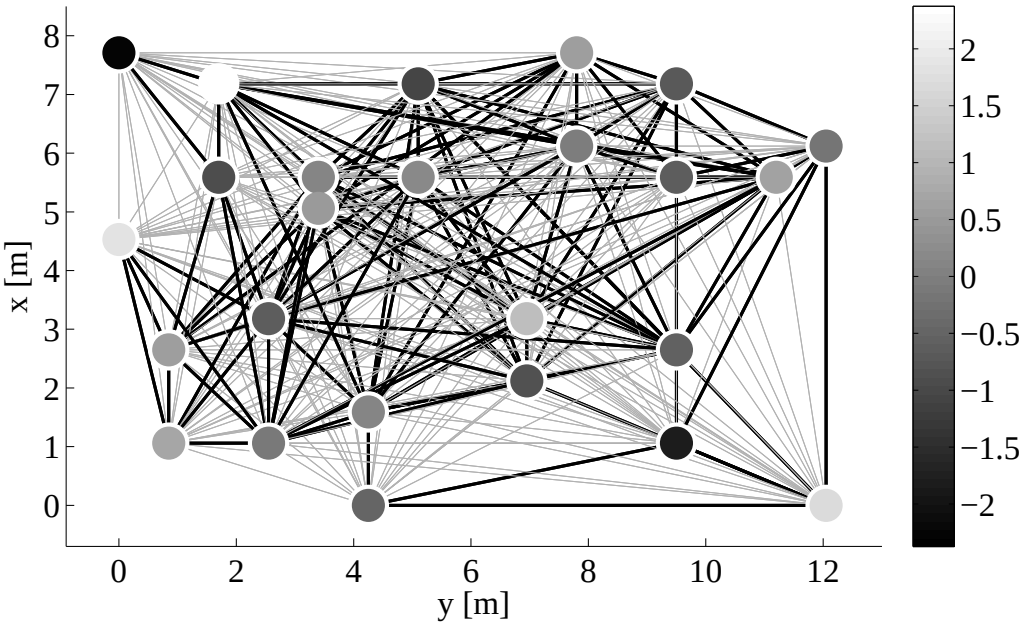


Figure 3.7: Network topology and node displacement for c-connectivity graph for $c = 0.1$. Nodes' grey intensity represents the estimated offset $\hat{o}_i$ after calibration. Black and grey edges represent the edges used for training and validation data sets, respectively.

failure set by the connectivity matrix C. Figure 3.8 shows the behavior of the consensus algorithm for a specific realization with two different values of the weight parameter in matrices $Q(t)$. The steady state compensation offsets $\hat{o}_i(\infty)$ are displayed in Figure 3.7 where the gray intensity of the nodes is proportional to $\hat{o}_i(\infty)$. Since the true node offsets o_i are unknown it is not possible to plot the behavior of $x_i(t) = \hat{o}_i(t) - o_i$ which are the variables that should converge to a common value, however the fact that all $\hat{o}_i$ converge to a steady state is an indication of correct functioning. It is also interesting to note the effect of unmodeled measurement noise arising from having neglected channel asymmetry and fast fading. In fact for larger w, i.e. for larger weight on the off-diagonal terms in the consensus matrix, the oscillation at steady state is not negligible, i.e. a large w tends to amplify noise. On the other hand, a small w leads to slower rate of convergence, thus indicating a tradeoff between convergence rate and noise sensitivity. Note also that the magnitude of steady state values of $\hat{o}_i$ is consistent with the a-priori dispersion indicated by the standard deviation σ_o reported in Table 3.1.

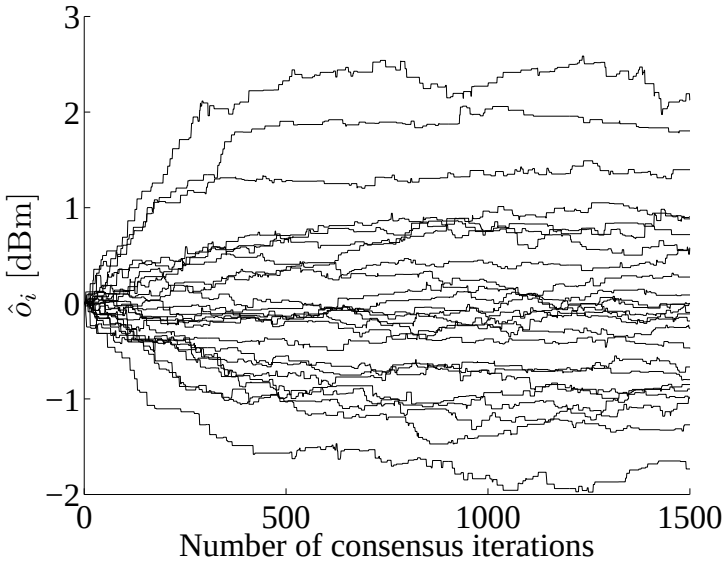
In order to evaluate the effectiveness of the offset calibration, we tested the channel asymmetry after calibration on a validation set different from the

set used for computing the offsets $\hat{o}_i$. The results of this second stage has been plotted in Figure 3.9. The white bars represent the distribution of $|\bar{P}^{ij} - \bar{P}^{ji}|$ on the validation edges, before the distributed sensor calibration algorithm is executed. The black bars, instead, show the distribution of $|(\bar{P}^{ij} - \hat{o}_i) - (\bar{P}^{ji} - \hat{o}_j)|$ after the algorithm has run. The offset reduction clearly appears. After the calibration, 56% of the validation edges have an asymmetric difference smaller than $0.5dBm$ (it was 24% before calibration), while 88% of them have an absolute error smaller than $1dBm$ (it was 50% before calibration). After the offset removal algorithm, almost all the measurements (99.4% of them) are affected by an asymmetric error smaller than $2dBm$.

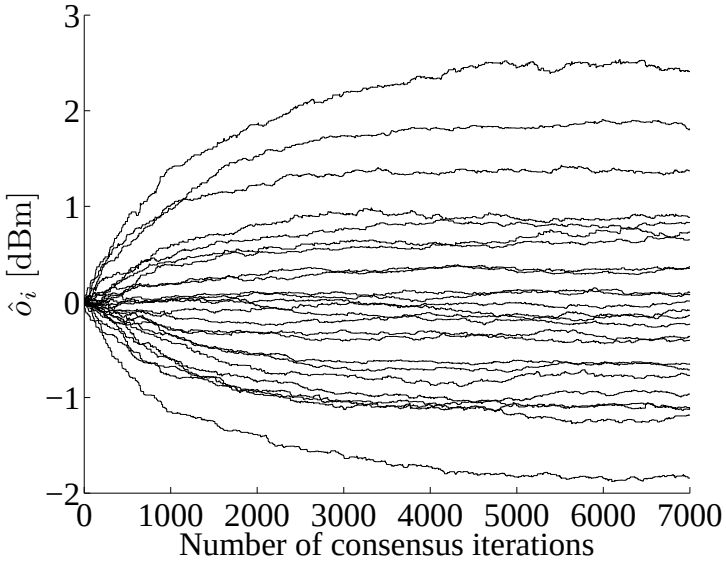
The importance of offset removal in the received power measurements is evident when these measurements are used for wireless–based localization. In fact, relative distance is estimated by inverting the path-loss function based on the calibrated measured power $\hat{P}_{rx}^{ij}$, i.e. $\hat{d}_{ij} = 10^{\frac{P_{tx}^{j} - \hat{P}_{rx}^{ij} + \beta}{10\gamma}} = 10^{\frac{P_{tx}^{j} - P_{rx}^{ij} - \hat{o}_i + \beta}{10\gamma}}$. If the calibration offset $\hat{o}_i$ is not included in the previous formula, there can be measurements errors up to $6dBm$ due to uncalibrated offsets, as Figure 3.9 suggests. In fact, a systematic calibration error of $6dBm$ corresponds to an uncertainty range from $0.9m$ to $4.4m$ when estimating the relative position of a node at $2m$, and to a practically useless estimation when the node is farther. An error of $1dBm$, on the other hand, corresponds to a error in the distance of only $28cm$ for a $2m$ long link, and to a $1.4m$ error when the node is at $10m$ distance.

3.4 Consensus–Based Least-Squares Wireless Channel Parameter Estimation

In this section we apply the algorithm presented in Section 2.5 to estimate in a distributed manner the unknown path-loss channel parameters (β, γ) given in equation (3.2) using different communication strategies. As mentioned above these two parameters are used in localization and target tracking algorithms in order to estimate relative distances between the moving node and the nodes of the static WSN. Therefore, it is critical to be able to identify the path-loss parameters in a distributed way, in a manner that is robust to node failure, with minimal exchange of data and low computational power, and without a central unit. It has to be noted that an accurate *a-priori* model for power loss in different indoor environments is almost unavailable (for example γ can vary from 1 to 6 according to the room sizes, the amount of furniture and people and the number of walls that the signal has to cross in average). Furthermore, the same environment can present a hourly or daily variation



(a) $w = 0.05$



(b) $w = 0.01$

Figure 3.8: Offset estimation $\hat{o}_i$ for each node of the considered WSN using randomized broadcast consensus for different values of the consensus weight w.

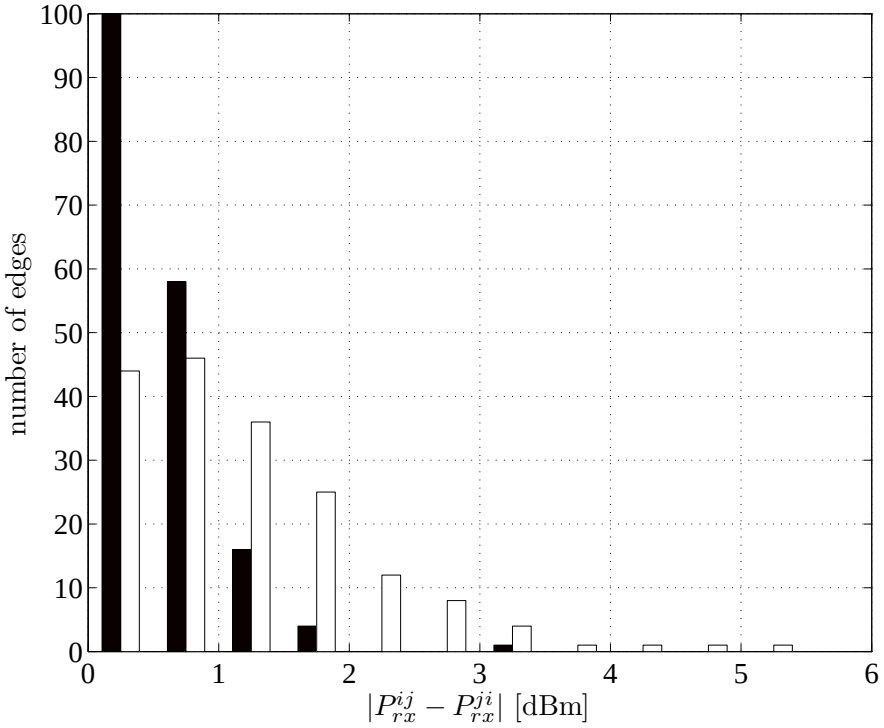


Figure 3.9: Asymmetric error distribution before ($|\bar{P}_{rx}^{ij} - \bar{P}_{rx}^{ji}|$, white) and after ($|\bar{P}_{rx}^{ij} + \hat{o}_i - \bar{P}_{rx}^{ji} - \hat{o}_j|$, black) distributed sensor calibration.

of these parameters due to the periodic presence of people populating the indoor spaces [53]. Fortunately, distributed algorithms with these features can be used to periodically or adaptively identify the channel parameters in a changing environment.

Based on these considerations, the focus of this section is to compare the performances of three different communication strategies which have different characteristics in terms of rate of convergence, communication complexity and parameter identification accuracy. The first and the second strategies are based on the implementation of the distributed least square identification described in Proposition 7 using the randomized broadcast and the randomized symmetric gossip, respectively. The third strategy performs the randomized symmetric gossip consensus on local estimates $\hat{\theta}_i$ of the channel parameters vector θ, rather than on the local least-square sufficient statistics $\left(x^A, x^b\right)$ relative to $\left(A^T A,\ A^T b\right)$ of Proposition 7. Each strategy has its own advantages. In fact, the randomized symmetric gossip guarantees average consensus, therefore it is guaranteed to provide the best identifica-

tion accuracy since it satisfies the hypotheses of Proposition 7. Randomized broadcast does not guarantee average consensus, and consequently the best performance, however it is very easy to implement since it needs no coordination between nodes. Moreover it is faster than the symmetric gossip since, on average, there are $d(i)$ updates per iteration compared with only 2 updates for the symmetric gossip. Finally, the strategy based on the average consensus of the local least-square estimates does not guarantee optimal performance nor best speed of convergence, however the number of parameters to be exchanged among nodes is equal to the size ℓ of the parameter vector θ, while for the first two strategies it is proportional to ℓ^2.

3.4.1 Simulations based on experimental data

We now describe in detail how the simulations are obtained. We considered the c-connectivity graph $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ for $c = 0.75$ shown in Figure 3.4. The data set $\mathcal{D}(i)$ available to each node i is given by $\mathcal{D}(i) = \{(\bar{P}_{rx}^{ij}, d_{ij}) \,|\, j \in \mathcal{V}(i)\}$, i.e. all the averaged received power measurements from each neighbor coupled with the corresponding relative distance (note that the distances are assumed to be known by the nodes). The data set of all measurements is indicated with $\mathcal{D} = \cup_{i \in \mathcal{V}} \mathcal{D}(i)$. We also assume that the offset calibration procedure of Section 3.3 has been performed in order to obtain the compensating offsets $\hat{o}_i$, and that the effect of fast-fading can be neglected since the measurements have been averaged over a large number of packets. Therefore, as shown at the end of Section 3.2, the channel parameters $\theta = [\beta \; \gamma]^T$ can be identified using a least square minimization by setting $a_m = [1 \; -10\log(d_{ij})]$, $b_m = \bar{P}_{rx}^{ij} - \hat{o}_i - P_{tx}$, where $m = 1, \ldots, M$ indicates a generic data element, and $M = |\mathcal{D}| = |\mathcal{E}_c|$. Using the same terminology of Proposition 7 we indicate with $\hat{\theta}_{LS}$ the centralized least-square estimate using the complete data set $\mathcal{D}$. We also indicate with $\hat{\theta}_{LS}^i$ the least-square estimate performed by the i-th node using only its data set $\mathcal{D}(i)$, which is the best estimate a node can have without communicating with the others. The performance (in terms of identification accuracy) is based on the residues of the estimate $\hat{\theta}$ given by:

$$J(\hat{\theta}) = \left\| A\hat{\theta} - b \right\|.$$

Note that A and b are constructed using the whole data set, and therefore $J(\hat{\theta})$ represents the global residual. Since $\hat{\theta}_{LS} = \arg\min_\theta J(\hat{\theta})$, it is obvious that $J(\hat{\theta}_{LS}) \leq J(\hat{\theta}_{LS}^i), \forall i$ from which it follows $J(\hat{\theta}_{LS}) \leq \frac{1}{N} \sum_{i \in \mathcal{V}} J(\hat{\theta}_{LS}^i)$. Being $\eta_i(0) = \hat{\theta}_{LS}^i$, if all $Q(t)$'s are doubly stochastic then from Proposition 7 it follows that

$$\lim_{t \to \infty} J(\eta_i(t)) = J(\hat{\theta}_{LS}), \forall i,$$

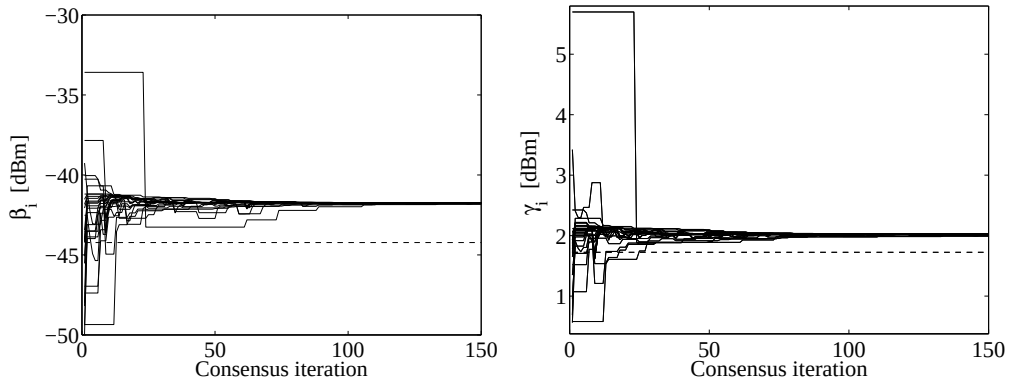


Figure 3.10: Convergence of parameter estimates β_i, γ_i using randomized broadcast least-square consensus and consensus weight $w = 0.5$. The dashed lines are the centralized least squares estimates $\hat{\beta}_{LS}, \hat{\gamma}_{LS}$.

and so

$$\lim_{t\to\infty} \frac{1}{N} \sum_{i\in\mathcal{V}} J(\eta_i(t)) = J(\hat{\theta}_{LS}).$$

In the first simulation, we tested the randomized broadcast least-square strategy using the connectivity matrix C defined in Section 1.1 for the link failure probabilities. Figure 3.10 shows the identified channel parameters of all nodes $\eta_i(t) = [\hat{\beta}_i(t)\ \hat{\gamma}_i(t)]^T$ as a function of the number of iterations for a typical realization of the system (thought as the stochastic process of information exchange). It can be seen that the identified parameters of all nodes converge to a common value, however, since broadcast does not guarantee average consensus, identified parameters do not necessarily coincide with the optimal estimate $\hat{\theta}_{LS}$. It is also interesting to note that most of the nodes have already good estimates of the parameters without communicating with the others, since most of them have lots of links and there are only two parameters to estimate. However, there are some nodes which have poor initial estimates, especially the ones on the perimeter of the graph which have few links. Nonetheless, thanks to the consensus algorithm, they rapidly converge to a good value.

In the second set of simulations, shown in Figure 3.11, we compared the rate of convergence and the steady state identification error for the three different strategies described above. More precisely, we compared the average estimation residual $\bar{J}(k) = \frac{1}{N}\sum_{i\in\mathcal{V}} J(\hat{\theta}_i(k))$ of all nodes as a function of iteration error. To reduce the randomness due to the choice of a particular realization of $\{P(t)\}_{t\in\mathbb{N}}$ we actually depicted $\mathbb{E}[\bar{J}(k)]$, approximately computed as the average of 50 independent extractions of the sequence $\{P(t)\}_{t\in\mathbb{N}}$. In

Consensus Algorithm	$\mathbb{E}[\bar{J}(\infty)]$	$\max \bar{J}(\infty)$	$\min \bar{J}(\infty)$
Broadcast $w = 0.5$	3.9816	4.1477	3.9320
Broadcast $w = 0.25$	3.9615	4.0919	3.9318
Symmetric Gossip	3.9307	3.9307	3.9307
Ave. of local estim.	3.9635	3.9635	3.9635

	$J_{Cent.L.S.}$
Centralized LS	3.9307

Table 3.2: Comparison of the mean estimation residual.

Table 3.2 it is reported also the steady state dispersion of $\bar{J}(k)$ around its mean value, obtained by recording the maximum and the minimum value of $\bar{J}(k)$ over the 50 extractions. In the bottom line the residual of the centralized optimal estimate is also reported for comparison.

Initially we tested the randomized broadcast least square algorithm for two different weights w. As already mentioned, larger w leads to faster convergence rates, however it also leads, in mean, to a larger steady state identification error (see [34, 54]). We also have that the steady state value is strongly realization dependent, as it can be noticed from to the large dispersion interval. This is due to the fact the first communications tend to bias the final value toward the initial condition of that node. Differently, if w is reduced, then this bias is smoothed out and $\mathbb{E}[\bar{J}(k)]$ end up closer to exact average consensus. Also the dispersion of the single realization with respect to $\mathbb{E}[\bar{J}(k)]$ reduces. Moreover it has been proved in [34] that the distance of $\mathbb{E}[\bar{J}(k)]$ form the average consensus decreases by increasing the number of nodes in the network, thus suggesting fast convergence rate with negligible performance degradation as compared, for example, to random symmetric gossip.

The same Figure 3.11 also shows the performance of the randomized symmetric gossip least square algorithm. As expected, the rate of convergence is slower, but the final value converges to the minimum identification error given by the centralized least-square estimate $J(\hat{\theta}_{LS})$. We remark that all the single realizations tend to the exact optimal value, as shown by the fact that there is no dispersion around the mean value (Table 3.2), not only that $\mathbb{E}[\bar{J}(k)]$ tends to optimal value.

Finally, we tested also a randomized gossip algorithm that directly averages the local least-square estimates. As shown in Figure 3.11, this strategy has the same rate of convergence of the randomized symmetric gossip (which computes the exact centralized least-square solution), but a slightly worse

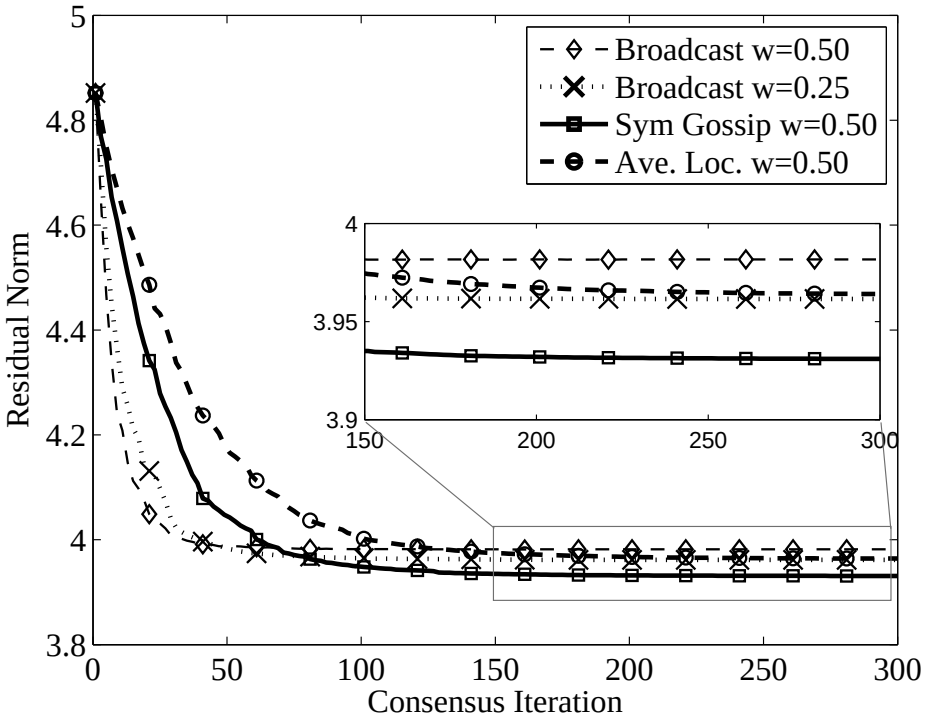


Figure 3.11: Comparison of the mean estimation residual $\mathbb{E}[\bar{J}(k)]$ for different randomized consensus algorithms.$\mathbb{E}[\bar{J}(k)]$ is approximately computed as the average of 50 independent extractions of the sequence $\{P(t)\}_{t\in\mathbb{N}}$.

performance. However, in terms of communication complexity this algorithm only requires the exchange of 2 parameters while the exact distributed least square one requires in this example the exchange of 4 parameters. It has to be noticed, though, that if the initial estimates were less reliable (for instance because the graph topology were much less connected) then the distributed least square would behave far better that the simple solution of an average of the local least squares estimations.

3.5 Final Comments and Open Issues

In this chapter we proposed consensus-based algorithms for wireless sensor networks and we successfully applied them to experimental data collected from a real WSN. In particular, we applied these algorithms to remove unknown offsets from the sensor measurements and to identify the parameters of the wireless channel for localization and tracking purposes.

However, these algorithms are rather general and can be applied in other fields and research areas. Indeed, we showed how it is possible to cast a wide class of problems into the consensus framework, such as problems in which the agents have to actually agree on a common estimate of few parameters (like in the least-square fitting), and problems in which every agent has to estimate its own parameter (like in the offset-removal algorithm). This duality deserves further investigation, possibly leading to a tighter relation between consensus algorithms [14] and asynchronous parallel iterative methods for the solution of system of linear equations [55].

The field of application of the proposed solution is definitely wider than the few applications presented in this work. For example, the offset removal algorithm could also be used to detect malfunctioning sensors by observing the magnitude of the compensation offset $\hat{o}_i$, while the least square square parameter identification algorithm can be used to identify any model parameter which is linear in the data.

Many issues remain to be explored, in particular in terms of correctly modeling real WSNs. For example we showed that, although the optimal solution to some problems depends on the average of the initial conditions, there are algorithms which do not guarantee convergence to the average, nonetheless providing good performances. Therefore, there is a definite need to better understand the trade-offs between performance, rate of convergence, communication complexity and noise sensitivity for different consensus strategies on real WSNs. Another important research avenue is the formulation of possibly nonlinear or non-standard problems into standard consensus problems, which is by no mean trivial.

4

A majorization Inequality result

In this chapter we will present a novel linear algebra result that will play a key role in the analysis carried out in Chapter 5.

Indeed, the proposed results can be more generally applied to the analysis of a jump Markov linear system, namely a linear time–varying systems for which it is available a statistical characterization of the transition matrix $Q(t)$, [27]. In a jump Markov linear system, in fact, the matrix $Q(t)$ is selected as a function of a Markov random process $\mathbf{r}(t)$, $Q(\mathbf{r}(t))$, becoming therefore itself a matrix-valued Markov random process. Jump Markov linear systems arise naturally in many applications and in particular in the sensor networks area, where, as we discussed in Section 1.2, very often the system under study is well modeled assuming that $Q(t)$ is drown at each time instant form an alphabet of matrices, according to a known probability law.

More precisely, assume we have a Markov random process $\mathbf{Q}(t)$, $t = 0, 1, 2, \ldots$, taking values in the set $\mathbb{R}^{N\times N}$ of $N \times N$ matrices and a jump Markov linear system described by equation

$$\mathbf{x}(t+1) = \mathbf{Q}(t)\mathbf{x}(t) \tag{4.1}$$

where $\mathbf{x}(t)$ is the state random vector.

It is natural to try to evaluate how "big" is the positive semidefinite matrix

$$P(t) := \mathbb{E}[\mathbf{Q}(t-1)\ldots\mathbf{Q}(1)\mathbf{Q}(0)\mathbf{Q}^T(0)\mathbf{Q}^T(1)\ldots\mathbf{Q}^T(t-1)] \tag{4.2}$$

Indeed, it is easy to see that

$$\mathbb{E}[||\mathbf{x}(t)||^2] = \mathbf{x}(0)^T P(t)\mathbf{x}(0)$$

where $||\cdot||$ means the Euclidean norm. Therefore, the evolution of the state $\mathbf{x}(t)$ is well described by the evolution of the positive semidefinite matrix $P(t)$.

One way to evaluate how big is $P(t)$ is through its 2-norm, which coincides with the largest eigenvalue of $P(t)$. An alternative way is to consider the trace of $P(t)$, which coincides with the sum of all the eigenvalues of $P(t)$. In this chapter we propose an upper bound on $P(t)$ in terms of the matrix $[P(t)]^t$ for the particular case in which $\mathbf{Q}(t)$ are independent.

We will see an application of the proposed result to distributed estimation in Chapter 5

Chapter Organization

The chapter is structured as follows. In section 4.1 we recall some basic concepts and some known results in majorizarion theory. In section 4.2 the main result is stated, we comment on the possibility to extend the obtained result and we formulate a conjecture. In section 4.3 the proof of the previously presented result is presented and commented. Finally, in section 4.4 we recapitulate the proposed results.

A Notation Note

In this chapter we will use lower case letters, $a, b, c, \ldots$, to denote scalars or vectors, capital letters, $A, B, C, \ldots$, to denote matrices and bold letters, $\mathbf{a}, \mathbf{A}, \mathbf{b}, \mathbf{B}$,$\mathbf{c}, \mathbf{C}, \ldots$, to denote random variables. We hope, in this way, to point out more clearly when we are dealing with a random variable $\mathbf{x}$ and when with its realization x.

Moreover, it might be worth to recall briefly to the reader some notations defined Section 2 and in particular that, given a matrix $M \in \mathbb{R}^{N\times N}$, we will denote with $\underline{\sigma}(M)$ the vector in $\mathbb{R}^N$ formed by the singular values of M, decreasingly ordered, and with $\underline{\lambda}(M)$ the vector in $\mathbb{C}^N$ formed by the eigenvalues of M, ordered so that $|\underline{\lambda}_0(M)| \geq \cdots \geq |\underline{\lambda}_{N-1}(M)|$, where each eigenvalue appears as many times as its algebraic multiplicity. Moreover, to improve readability, we will write $\sigma_j^i(M)$ to denote $(\sigma_j(M))^i$.

Recall finally that we say that the matrix N is normal if $NN^T = N^TN$.

4.1 Some Preliminary Results in Majorization Theory

Let us begin by reviewing some basic concepts about majorization.

Definition 2. *Given two vector $x, y \in \mathbb{R}^N$ whose components are ordered decreasingly, i.e. $x_1 \geq x_2 \geq \cdots \geq x_N$, we say that x submajorize y and we write $y \prec_w x$ if*

$$\sum_{i=1}^{k} y_i \leq \sum_{i=1}^{k} x_i \qquad \forall k = 1, \ldots, N$$

If moreover the inequality for $k = N$ holds as an equality, then we say that x majorize y and we write $y \prec x$

A broad literature on majorization theory and its applications is available. An introduction to the topic can be found, for instance, in [56, 57]. Majorization theory is extensively treated in [58] and in [59], a book fully devoted to this topic.

In particular, recall that the following lemma holds

Lemma 9.
Let x, y, and z be real, non–negative decreasingly ordered vectors. Then

$$y \prec_w x \quad \Rightarrow \quad y \odot z \prec_w x \odot z.$$

Proof.
See [59, page 92, H.2.c]. An alternative proof can be found in the appendix of this chapter, 4.A. □

Recall moreover an important result on the singular values of the product of two matrices

Lemma 10. *Given any two matrices M_1 and M_2*

$$\underline{\sigma}(M_1 M_2) \prec_w \underline{\sigma}(M_1) \odot \underline{\sigma}(M_2). \tag{4.3}$$

Proof.
See [57, 58]. □

Furthermore there is an important result specifying the relation between eigenvalues and singular values of a matrix

Lemma 11. *Given any matrix M*

$$|\underline{\lambda}(M)| \prec_w \underline{\sigma}(M). \tag{4.4}$$

Proof.
See [57, 58]. □

Combining the previous two results, we get that, given any two normal matrices N_1 and N_2,

$$|\underline{\lambda}(N_1 N_2)| \prec_w \underline{\sigma}(N_1) \odot \underline{\sigma}(N_2) = |\underline{\lambda}(N_1)| \odot |\underline{\lambda}(N_2)|. \tag{4.5}$$

We will use this inequality in the following, together with the fact that for any matrix M

$$\lambda_j(M^i) = \lambda_j^i(M)$$

and, therefore, for any normal matrix N, $NN^T = N^T N$

$$\sigma_j(N^i) = \sigma_j^i(N).$$

4.2 A Majorization Inequality

Let us introduce a finite indexes set $\mathbb{A}$ and consider a finite alphabet of normal matrices

$$\{Q_\alpha \ \alpha \in \mathbb{A}\}, \qquad Q_\alpha Q_\alpha^T = Q_\alpha^T Q_\alpha \quad \forall \alpha \in \mathbb{A}$$

from which we assume to randomly draw a matrix. Denote with $\mathbb{P}$ the probability measure on the alphabet. We will refer to $\mathbb{P}$ as selection probability and we will denote p_α the probability that Q_α is drawn. Consider then the random process $\{\mathbf{Q}(t)\}_{t\in\mathbb{N}}$ of independent and identically distributed random variables $\mathbb{P}[\mathbf{Q}(t) = Q_\alpha] = p_\alpha \ \forall t$ that describes independent extractions form the alphabet $\{Q_\alpha \ \alpha \in \mathbb{A}\}$.

We want to study the matrix

$$\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\right].$$

We present now a result that relates the spectrum of this positive semidefinite matrix to the spectrum of $\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^i$. The proof is reported in Section 4.3.

Theorem 12. *Given any finite normal matrix alphabet* $\{Q_\alpha \ \alpha \in \mathbb{A}\}$, $\forall i \in \mathbb{N}$ *we have that*

$$\underline{\sigma}\Big(\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\right]\Big) \prec_w \underline{\sigma}\left(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^i\right). \tag{4.6}$$

that is

$$\sum_{j=1}^{k}\sigma_j\Big(\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\right]\Big)\leq\sum_{j=1}^{k}\sigma_j\left(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^i\right) \tag{4.7}$$

$\forall k=1,\dots,N$ *and* $\forall i\in\mathbb{N}$.

The above result, for $k=n$, gives the following trace inequality

Corollary 13. *Given any finite normal matrix alphabet* $\{Q_\alpha\ \alpha\in\mathbb{A}\}$, $\forall i\in\mathbb{N}$ *we have that*

$$\mathrm{tr}\Big(\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\right]\Big)\leq\mathrm{tr}\left(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^i\right).$$

Moreover, again as a trivial corollary of the above result, we get an inequality on the largest eigenvalue $\lambda_{\max}$, already proved in [20]

Corollary 14. *Given any finite normal matrix alphabet* $\{Q_\alpha\ \alpha\in\mathbb{A}\}$,

$$\lambda_{\max}\Big(\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\right]\Big)\leq\lambda^i_{\max}\left(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]\right).$$

Remark 1. Since any symmetric matrix is also normal, the proposed theorem covers the special case of symmetric matrix alphabet.

4.2.1 Comments on the result

One might wonder if it is possible to relax the hypothesis of normality of the matrices Q_α of the alphabet. The answer is no, as the following counterexample shows.

Counterexample.

Let $\mathbb{A}=\{1,2\}$, let $p_{\mathbb{A}}=\{p_1=\frac{1}{3},\ p_2=\frac{2}{3}\}$ and let

$$\{Q_\alpha\ \alpha\in\mathbb{A}\}=\left\{Q_1=\begin{bmatrix}0&1\\0&0\end{bmatrix},\qquad Q_2=\begin{bmatrix}0&0\\0&1\end{bmatrix}\right\}.$$

Note that Q_1 is not normal since

$$Q_1Q_1^T=\begin{bmatrix}1&0\\0&0\end{bmatrix}\neq\begin{bmatrix}0&0\\0&1\end{bmatrix}=Q_1^TQ_1$$

We get that

$$\mathbb{E}\left[\mathbf{Q}(1)\mathbf{Q}(0)\mathbf{Q}(0)^T\mathbf{Q}(1)^T\right]=\frac{1}{9}\begin{bmatrix}2&0\\0&4\end{bmatrix}\qquad\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^2=\frac{1}{9}\begin{bmatrix}1&0\\0&4\end{bmatrix}$$

Therefore it does not hold that $\underline{\sigma}(\mathbb{E}\left[\mathbf{Q}(1)\mathbf{Q}(0)\mathbf{Q}(0)^T\mathbf{Q}(1)^T\right]) \prec_w \underline{\sigma}(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^2)$ since

$$\sigma_0(\mathbb{E}\left[\mathbf{Q}(1)\mathbf{Q}(0)\mathbf{Q}(0)^T\mathbf{Q}(1)^T\right]) + \sigma_1(\mathbb{E}\left[\mathbf{Q}(1)\mathbf{Q}(0)\mathbf{Q}(0)^T\mathbf{Q}(1)^T\right]) = \frac{6}{9} > \\ > \frac{5}{9} = \sigma_0(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^2) + \sigma_1(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^2)$$

□

Many numerical experiments instead supported the conjecture that the statement of Theorem 12 is still holds true in the case of stochastic, possibly non-normal, matrices.

Conjecture. *Given any finite stochastic, possibly non–normal, matrix alphabet, it holds the following*

$$\underline{\sigma}\Big(\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\right]\Big) \prec_w \underline{\sigma}\left(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^i\right) \ \forall i \in \mathbb{N}.$$

The conjecture has been verified for $i \leq 10$, for matrices of size $N < 10$ and for alphabet's cardinality $|\mathbb{A}| < 20$. In total, we randomly generate more than $60 \cdot 10^6$ stochastic matrix alphabets and probability measures on them. In all these cases we verified that the conjecture holds.

The proposed conjecture have relevant applications in distributed estimation over wireless sensors network, since in that case the matrices of the alphabet are known to be always stochastic.

4.3 Proof of Theorem 12

The proof is based on the following lemma

Lemma 15. *Given any normal matrix alphabet $\{Q_\alpha\ \alpha \in \mathbb{A}\}$, and any symmetric positive semi-definite matrix $P \geq 0$, we have that*

$$\underline{\sigma}\Big(\mathbb{E}\left[\mathbf{Q}(0)P\mathbf{Q}(0)^T\right]\Big) \prec_w \underline{\sigma}\left(P\right) \odot \underline{\sigma}\left(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]\right). \tag{4.8}$$

Proof.
Note that the thesis (4.8) is equivalent to prove that, $\forall k = 1, \dots, N$

$$\sum_{j=1}^{k} \sigma_j\left(\sum_{\alpha\in\mathbb{A}} p_\alpha Q_\alpha P Q_\alpha^T\right) \leq \sum_{j=1}^{k} \sigma_j\left(P\right) \sigma_j\left(\sum_{\alpha\in\mathbb{A}} p_\alpha Q_\alpha Q_\alpha^T\right). \tag{4.9}$$

To this aim recall ([57, 58]) that for any matrix M

$$\sum_{j=1}^{k} \sigma_j (M) = \max_{U^T U = I_k} \operatorname{tr} \left(U^T M U \right), \tag{4.10}$$

therefore

$$\sum_{j=1}^{k} \sigma_j \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha P Q_\alpha^T \right) = \max_{U^T U = I_k} \operatorname{tr} \left(U^T \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha P Q_\alpha^T \right) U \right).$$

Note, moreover, that

$$\begin{aligned} P &= V^T \mathrm{diag}\{[\sigma_1(P), \ldots, \sigma_{k-1}(P), \sigma_k(P), \ldots, \sigma_N(P)]\} V \\ &\leq V^T \mathrm{diag}\{[\sigma_1(P) - \sigma_k(P), \ldots, \sigma_{k-1}(P) - \sigma_k(P), 0, \ldots, 0]\} V + \sigma_k(P) I \\ &= \bar{P} + \sigma_k(P) I, \end{aligned}$$

where

$$\bar{P} = V^T \mathrm{diag}\{[\sigma_1(P) - \sigma_k(P), \ldots, \sigma_{k-1}(P) - \sigma_k(P), 0, \ldots, 0]\} V.$$

Therefore

$$\begin{aligned} \max_{U^T U = I_k} \mathrm{tr} U^T \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha P Q_\alpha^T \right) U \leq \max_{U^T U = I_k} \mathrm{tr} U^T \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha \bar{P} Q_\alpha^T \right) U + \\ + \sigma_k(P) \mathrm{tr} U^T \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha Q_\alpha^T \right) U. \end{aligned} \tag{4.11}$$

One has, again for (4.10),

$$\sigma_k(P) \mathrm{tr} \left(U^T \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha Q_\alpha^T \right) U \right) \leq \sigma_k(P) \sum_{j=1}^{k} \sigma_j \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha Q_\alpha^T \right). \tag{4.12}$$

The other term of the sum in (4.11) can be upper–bounded by noting that

$$\begin{aligned} \operatorname{tr} \left(U^T \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha \bar{P} Q_\alpha^T \right) U \right) &= \sum_{\alpha \in \mathbb{A}} p_\alpha \operatorname{tr} \left(U^T Q_\alpha \bar{P}^{\frac{1}{2}} \bar{P}^{\frac{1}{2}} Q_\alpha^T U \right) \\ &= \sum_{\alpha \in \mathbb{A}} p_\alpha \operatorname{tr} \left(\bar{P}^{\frac{1}{2}} Q_\alpha^T U U^T Q_\alpha \bar{P}^{\frac{1}{2}} \right). \end{aligned} \tag{4.13}$$

Noting that

$$UU^T \leq I$$

in the sense of the positive semidefinite matrices and recalling that, given two positive semidefinite matrices A and B, such that $A \leq B$, it holds that

$$\mathrm{tr}X^T AX \leq \mathrm{tr}X^T BX,$$

form (4.13), one gets

$$\begin{aligned}\sum_{\alpha\in\mathbb{A}} p_\alpha \mathrm{tr}\left(\bar{P}^{\frac{1}{2}} Q_\alpha^T U U^T Q_\alpha \bar{P}^{\frac{1}{2}}\right) &\leq \sum_{\alpha\in\mathbb{A}} p_\alpha \mathrm{tr}\left(\bar{P}^{\frac{1}{2}} Q_\alpha^T Q_\alpha \bar{P}^{\frac{1}{2}}\right) \\ &= \mathrm{tr}\left(\bar{P}^{\frac{1}{2}}\left(\sum_{\alpha\in\mathbb{A}} p_\alpha Q_\alpha^T Q_\alpha\right)\bar{P}^{\frac{1}{2}}\right) \\ &= \sum_{j=1}^{N} \lambda_j\left(\bar{P}^{\frac{1}{2}}\left(\sum_{\alpha\in\mathbb{A}} p_\alpha Q_\alpha^T Q_\alpha\right)\bar{P}^{\frac{1}{2}}\right).\end{aligned}$$

Using (4.5) and recalling that, for any semi–positive definite matrix $A \geq 0$, $\underline{\lambda}(A) \in \mathbb{R}^N$ and $\underline{\lambda}(A) \geq 0$ one gets

$$\sum_{j=1}^{N} \lambda_j\left(\bar{P}^{\frac{1}{2}}\left(\sum_{\alpha\in\mathbb{A}} p_\alpha Q_\alpha^T Q_\alpha\right)\bar{P}^{\frac{1}{2}}\right) \leq \sum_{j=1}^{N} \lambda_j\left(\bar{P}^{\frac{1}{2}}\right)\lambda_j\left(\sum_{\alpha\in\mathbb{A}} p_\alpha Q_\alpha^T Q_\alpha\right)\lambda_j\left(\bar{P}^{\frac{1}{2}}\right).$$

Therefore

$$\begin{aligned}\sum_{\alpha\in\mathbb{A}} p_\alpha \mathrm{tr}\left(\bar{P}^{\frac{1}{2}} Q_\alpha^T U U^T Q_\alpha \bar{P}^{\frac{1}{2}}\right) &\leq \sum_{j=1}^{N} \lambda_j\left(\bar{P}\right)\lambda_j\left(\sum_{\alpha\in\mathbb{A}} p_\alpha Q_\alpha^T Q_\alpha\right) \\ &= \sum_{j=1}^{k-1}(\sigma_j(P) - \sigma_k(P))\sigma_j\left(\sum_{\alpha\in\mathbb{A}} p_\alpha Q_\alpha^T Q_\alpha\right).\end{aligned} \tag{4.14}$$

Recall that, since we are considering a normal matrix alphabet,

$$Q_\alpha^T Q_\alpha = Q_\alpha Q_\alpha^T,$$

combining (4.11), (4.12) and (4.14) one gets

$$\begin{aligned}
\sum_{j=1}^{k}\sigma_j\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha P Q_\alpha^T\right) &= \max_{U^TU=I_k}\operatorname{tr}\left(U^T\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha P Q_\alpha^T\right)U\right)\\
&\leq \sum_{j=1}^{k-1}(\sigma_j(P)-\sigma_k(P))\sigma_j\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha Q_\alpha^T\right)+\sigma_k(P)\sum_{j=1}^{k}\sigma_j\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha Q_\alpha^T\right)\\
&= \sum_{j=1}^{k-1}\sigma_j(P)\sigma_j\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha Q_\alpha^T\right)-\sum_{j=1}^{k-1}\sigma_k(P)\sigma_j\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha Q_\alpha^T\right)+\\
&\qquad +\sigma_k(P)\sum_{j=1}^{k}\sigma_j\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha Q_\alpha^T\right)\\
&= \sum_{j=1}^{k-1}\sigma_j(P)\sigma_j\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha Q_\alpha^T\right)+\sigma_k(P)\sigma_k\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha Q_\alpha^T\right)\\
&= \sum_{j=1}^{k}\sigma_j(P)\sigma_j\left(\sum_{\alpha\in\mathbb{A}}p_\alpha Q_\alpha Q_\alpha^T\right).
\end{aligned}$$

which concludes the proof of the lemma. □

Proof of the theorem.
We will prove the theorem by induction.
It is trivially true for $i=1$.
Suppose that the thesis (4.6) holds true for i and let us prove that this implies it holds true for $i+1$. Let us define

$$P(i)=\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\right]$$

Note that $P(i)$ is symmetric and positive semidefinite $P(i)\geq 0\ \forall i$.
Recalling that $\mathbf{Q}(0)\dots\mathbf{Q}(i)$ are independent and identically distributed, one gets

$$\begin{aligned}
P(i+1) &= \mathbb{E}\left[\mathbf{Q}(i)\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\mathbf{Q}(i)^T\right]\\
&= \mathbb{E}\left[\mathbb{E}\left[\mathbf{Q}(i)\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\mathbf{Q}(i)^T|\mathbf{Q}(i)\right]\right]\\
&= \mathbb{E}\left[\mathbf{Q}(i)\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T|\mathbf{Q}(i)\right]\mathbf{Q}(i)^T\right]\\
&= \mathbb{E}\left[\mathbf{Q}(i)\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\right]\mathbf{Q}(i)^T\right]\\
&= \mathbb{E}\left[\mathbf{Q}(i)P(i)\mathbf{Q}(i)^T\right]=\mathbb{E}\left[\mathbf{Q}(0)P(i)\mathbf{Q}(0)^T\right]
\end{aligned}$$

We want therefore to prove that

$$\sum_{j=1}^{k}\sigma_j\left(\mathbb{E}\left[\mathbf{Q}(0)P(i)\mathbf{Q}(0)^T\right]\right)\leq\sum_{j=1}^{k}\sigma_j^{i+1}\left(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]\right)\ \forall k=1,\dots,N$$

that is, $\forall k = 1, \ldots, N$

$$\sum_{j=1}^{k} \sigma_j \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha P(i) Q_\alpha^T \right) \leq \sum_{j=1}^{k} \sigma_j^{i+1} \left(\sum_{\alpha \in \mathbb{A}} p_\alpha Q_\alpha Q_\alpha^T \right).$$

From lemma 15 we know that

$$\sum_{j=1}^{k} \sigma_j \Big(\mathbb{E}\left[\mathbf{Q}(0) P(i) \mathbf{Q}(0)^T \right] \Big) \leq \sum_{j=1}^{k} \sigma_j \left(P(i) \right) \sigma_j \left(\mathbb{E}\left[\mathbf{Q}(0) \mathbf{Q}(0)^T \right] \right).$$

Moreover, by inductive hypothesis, we have that

$$\sum_{j=1}^{k} \sigma_j \left(P(i) \right) \leq \sum_{j=1}^{k} \sigma_j^i \left(\mathbb{E}\left[\mathbf{Q}(0) \mathbf{Q}(0)^T \right] \right).$$

Therefore, using lemma 9, we get

$$\begin{aligned} \sum_{j=1}^{k} \sigma_j \left(\mathbb{E}\left[\mathbf{Q}(0) P(i) \mathbf{Q}(0)^T \right] \right) &\leq \sum_{j=1}^{k} \sigma_j^i \left(\mathbb{E}\left[\mathbf{Q}(0) \mathbf{Q}(0)^T \right] \right) \sigma_j \left(\mathbb{E}\left[\mathbf{Q}(0) \mathbf{Q}(0)^T \right] \right) = \\ &= \sum_{j=1}^{k} \sigma_j^{i+1} \left(\mathbb{E}\left[\mathbf{Q}(0) \mathbf{Q}(0)^T \right] \right), \end{aligned}$$

that completes the proof. □

Remark 2. The proposed theorem holds true also in the case of infinite alphabet sets, both countable and uncountable. In fact, the assumption of finiteness of the alphabet does not play a central role in the proof and can be dropped, taking care to guarantee the boundedness of the expectations that appear in the proof. To this aim it is sufficient to add the further assumption of boundedness of $\mathbb{E}\left[\mathbf{Q}(0) \mathbf{Q}(0)^T \right]$.

To state precisely and clearly the theorem in the case of non-finite alphabet some care has to be used in the definition of the random variable $\mathbf{Q}$. Consider then a measurable space $(\mathbb{A} \subseteq \mathbb{R}, \mathcal{A})$ endowed with a probability measure $\mathbb{P}$. Consider then a random variable

$$\begin{aligned} \mathbf{Q} : A \subseteq \mathbb{R} &\longrightarrow \{ N \in \mathbb{R}^{N \times N} : \ N N^T = N^T N \} \\ \alpha &\longmapsto \mathbf{Q}(\alpha). \end{aligned}$$

We have then that

$$\mathbb{E}[\mathbf{Q}] = \int \mathbf{Q}(\alpha)\mathrm{d}\mathbb{P}(\alpha) = \int_{\mathbb{A}} \begin{bmatrix} q_{11}(\alpha) & \dots & q_{1N}(\alpha) \\ \vdots & & \vdots \\ q_{N1}(\alpha) & \dots & q_{NN}(\alpha) \end{bmatrix} \mathrm{d}\mathbb{P}(\alpha) =$$
$$= \begin{bmatrix} \int_{\mathbb{A}} q_{11}(\alpha)\mathrm{d}\mathbb{P}(\alpha) & \dots & \int_{\mathbb{A}} q_{12}(\alpha)\mathrm{d}\mathbb{P}(\alpha) \\ \vdots & & \vdots \\ \int_{\mathbb{A}} q_{21}(\alpha)\mathrm{d}\mathbb{P}(\alpha) & \dots & \int_{\mathbb{A}} q_{22}(\alpha)\mathrm{d}\mathbb{P}(\alpha) \end{bmatrix}.$$

In particular, if the measure $\mathbb{P}$ is purely discrete

$$\mathbb{E}[\mathbf{Q}] = \sum_{\alpha\in\mathbb{A}} Q(\alpha)\mathbb{P}[\{\alpha\}],$$

while, if the measure $\mathbb{P}$ is absolutely continuous with respect to the Lebesgue measure, $\mathrm{d}\mathbb{P} = p(\alpha)\mathrm{d}\alpha$

$$\mathbb{E}[\mathbf{Q}] = \begin{bmatrix} \int_{\mathbb{A}} q_{11}(\alpha)p(\alpha)\mathrm{d}\alpha & \dots & \int_{\mathbb{A}} q_{12}(\alpha)p(\alpha)\mathrm{d}\alpha \\ \vdots & & \vdots \\ \int_{\mathbb{A}} q_{21}(\alpha)p(\alpha)\mathrm{d}\alpha & \dots & \int_{\mathbb{A}} q_{22}(\alpha)p(\alpha)\mathrm{d}\alpha \end{bmatrix}.$$

When using a non-finite alphabet $\mathbb{A}$, the existence and the boundedness of the expectation is an issue. Nevertheless, to guarantee finiteness of the expectations that appear in the proof of Theorem 12 it is sufficient to assume that $\mathbb{E}[\mathbf{Q}(0)^T\mathbf{Q}(0)]$ exists finite. In fact, using the normality assumption of Theorem 12, we have that

$$\begin{aligned} \mathrm{tr}\mathbb{E}[\mathbf{Q}(0)^T M\mathbf{Q}(0)] &= \mathrm{tr}M\mathbb{E}[\mathbf{Q}(0)\mathbf{Q}(0)^T] \\ &= \mathrm{tr}M\mathbb{E}[\mathbf{Q}(0)^T\mathbf{Q}(0)] \\ &\leq \mathrm{tr}M\mathrm{tr}\mathbb{E}[\mathbf{Q}(0)^T\mathbf{Q}(0)]. \end{aligned}$$

Therefore, if $\mathbb{E}[\mathbf{Q}(0)^T\mathbf{Q}(0)]$ exists finite then $\mathbb{E}[\mathbf{Q}(0)^T M\mathbf{Q}(0)]$ exists finite for any matrix M and hence there always exists finite $\mathbb{E}[\mathbf{Q}(i)\dots\mathbf{Q}(0)^T\mathbf{Q}(0)\dots\mathbf{Q}(i)]$.

Once the existence issues has been addressed, (4.6) still holds also for infinite alphabets of normal matrices as it can be seen easily going through the proof and noting that it is still true that $\mathrm{tr}(\mathbb{E}[\mathbf{Q}]) = \mathbb{E}[\mathrm{tr}\mathbf{Q}]$ and that, for any matrices A, B, $\mathbb{E}[A\mathbf{Q}B] = A\mathbb{E}[\mathbf{Q}]B$.

Therefore Theorem 12 can be generalized as follows

Theorem 16 (Genralization of Theorem 12). *Given any normal matrix alphabet* $\{Q(\alpha),\ \alpha\in\mathbb{A}\}$ *and any i.i.d. random process* $\{\mathbf{Q}(t)\}_{t\in\mathbb{N}}$ *taking values in this alphabet such that* $tr\mathbb{E}[\mathbf{Q}(0)^T\mathbf{Q}(0)] \leq +\infty$ *is finite, it holds the following*

$$\underline{\sigma}\Big(\mathbb{E}\left[\mathbf{Q}(i-1)\dots\mathbf{Q}(0)\mathbf{Q}(0)^T\dots\mathbf{Q}(i-1)^T\right]\Big) \prec_w \underline{\sigma}\left(\mathbb{E}\left[\mathbf{Q}(0)\mathbf{Q}(0)^T\right]^i\right)\ \forall i\in\mathbb{N}.$$

Remark 3. Note moreover that in the proposed proof it has never been used the fact that $\mathbb{P}(\alpha)$ is such that $\sum_{\alpha\in\mathbb{A}} p_\alpha = 1$. The presented result holds therefore for any non–negative weight function, $p(\alpha) \geq 0\ \forall\alpha \in \mathbb{A}$, even if $\sum_{\alpha\in\mathbb{A}} p_\alpha \neq 1$ ($\int_{\mathbb{A}} p(\alpha)\mathrm{d}\alpha \neq 1$).

4.4 Final Comments and Open Issues

In this chapter we presented a majorization inequality on the singular values of the expectation of matrix–valued random variables' product. It is shown that, if the alphabet is made of normal matrices, then the singular values of the matrix $\mathbb{E}[\mathbf{Q}_i \dots \mathbf{Q}_0\mathbf{Q}_0^T \dots \mathbf{Q}_i^T]$ are submajorized by the singular values of $\mathbb{E}[\mathbf{Q}_0\mathbf{Q}_0^T]^i$.

As a straightforward corollary of this result, we proposed a novel trace inequality.

The proposed result applies to the analysis of jump–Markov linear systems where the transition matrix is an i.i.d. random process. A relevant example of application in this framework is the case of a linear plant controlled by a linear time–varying feedback, in which at each time instant a feedback matrix $K(t)$ is drawn from a fixed set of matrices $K_1, \dots, K_N$ and in which the designer only chose the probability that the various feedback matrices are applied.

Many numerical experiments support the conjecture that the statement of Theorem 12 holds true also in the case of stochastic, possibly non-normal, matrices. We expect an eventual proof of such a conjecture to leverage on arguments different from the ones we used here. In fact, normality is a key assumption in our proof and we do not see how to take advantage of stochasticity in our arguments.

4.A Appendix: Proof of Lemma 9

In this appendix we proof Lemma 9.

Consider two vectors $x, y \in \mathbb{R}^N$, $x = [x_1 \dots x_N]^T$ and $y = [y_1 \dots y_N]^T$ and define $x_0 = y_0 = 0$. As it can easily verified by direct computation, it holds that, $\forall i = 1 \dots n$:

$$x_i y_i - x_{i-1} y_{i-1} = (x_i - x_{i-1})\, y_i + x_{i-1}\,(y_i - y_{i-1})\,. \tag{4.15}$$

We have moreover that

$$\sum_{j=1}^{k} ((x_i - x_{i-1})\, y_i) = x_k y_k - x_0 y_0 - \sum_{j=1}^{k} (x_{i-1}\,(y_i - y_{i-1}))\,. \tag{4.16}$$

In fact, using (4.15), we have

$$x_k y_k - x_0 y_0 = \sum_{j=1}^{k} (x_i y_i - x_{i-1} y_{i-1}) = \sum_{j=1}^{k} ((x_i - x_{i-1})\, y_i) + \sum_{j=1}^{k} (x_{i-1}\,(y_i - y_{i-1}))\,.$$

We are now ready to proof Lemma 9. Note that it can be restated as

Lemma 17 (Restatement of Lemma 9.)**.** *Let x, y, and z be real, non–negative decreasingly ordered vectors. Then if*

$$\sum_{i=1}^{k} y_i \leq \sum_{i=1}^{k} x_i$$

for all $k = 1, \dots, N$, then

$$\sum_{i=1}^{k} y_i z_i \leq \sum_{i=1}^{k} x_i z_i$$

$k = 1, \dots, N$.

Proof.
Define

$$X_k = \sum_{i=1}^{k} x_i \qquad Y_k = \sum_{i=1}^{k} y_i.$$

Define moreover $X_0 = Y_0 = 0$ and $z_0 = z_1$. One has than that

$$X_i - X_{i-1} = x_i \qquad Y_i - Y_{i-1} = y_i \qquad \forall i = 1 \dots N.$$

Consider then

$$\begin{aligned}
&\sum_{i=1}^{k} x_i z_i - \sum_{i=1}^{k} y_i z_i = \sum_{i=1}^{k} (x_i - y_i) z_i \\
&= \sum_{i=1}^{k} \left((X_i - X_{i-1}) - (Y_i - Y_{i-1}) \right) z_i \\
&= \sum_{i=1}^{k} \left((X_i - Y_i) - (X_{i-1} - Y_{i-1}) \right) z_i = \left[\text{see } (4.16) \right] = \\
&= (X_k - Y_k)\, z_k - \underbrace{(X_0 - Y_0)}_{0} z_0 - \sum_{i=1}^{k} (X_i - Y_i)\, (z_i - z_{i-1}) \\
&= \underbrace{(X_k - Y_k)}_{\geq 0} \underbrace{z_k}_{\geq 0} \underbrace{- \sum_{i=1}^{k} \underbrace{(X_i - Y_i)}_{\geq 0} \underbrace{(z_i - z_{i-1})}_{\leq 0}}_{\geq 0} \geq 0.
\end{aligned}$$

That is

$$\sum_{i=1}^{k} x_i z_i \geq \sum_{i=1}^{k} y_i z_i,$$

or equivalently the thesis. □

5

Randomized Gossip Kalman Filter

In this chapter we consider the problem of estimating a random process from noisy measurements, collected by a sensor network. More precisely, as in [40], we consider a prototypical problem of estimation in sensor networks, namely the problem of estimating the state of a scalar random process. We will analyze a distributed two–staged algorithm: the first stage is a Kalman–like estimate update, in which each agent makes use only of its own measurements, while the second phase is devoted to the estimates exchange between neighbor nodes and to the estimates fusion.

To find an optimal way to fuse local estimates is a very difficult problem, that can hardly be handled if the communication graph has cycles. Nevertheless,as we reported in Section 2.3, the problem of finding a distributed algorithm that achieves the same performance of the centralized Kalman filter has been solved when communication are much faster than measurements, [60, 61]. To this aim the estimate fusion problem has been formulated as a consensus problem. In fact, as remarked in [40], since the local estimates mean is a sufficient statistic to compute the optimal estimate, the optimal fusion problem can be solved with consensus techniques. Unluckily, the solution proposed in [60, 61] strongly relies on the assumption that communications are much faster than measurements, so that one can assume that consensus is be reached during two consecutive measurements. Under this assumption, the choice of the Kalman gain to be used at each node is not an issue: the centralized gain has to be used. Obviously, the assumption of fast communications does not hold in all practical cases and the proposed scheme becomes suboptimal.

In [40] and in [62] it has been studied the case of finite number of communications between two subsequent measurements. In both works, inspired by the fact that the mean of the local estimates is a sufficient statistic to compute

the centralized estimate, estimate fusion was implemented as m consensus steps. In [62] and in the subsequent improvement [63], Kalman gain and consensus weights are selected at each time step in order to minimize the estimate error variance at each node in the next step. In [40] the steady state error variance was minimized, for the case of a scalar random walk. Both of the approaches consider a fixed communication scheme, i.e. it is assumed that all the information exchanges prescribed by the communication graph happen between 2 subsequent measurements.

However, in many practical cases this is just a rough model of communications of a sensor network, valid if the time between two subsequent measurements is sufficiently large. In fact, communications in a sensor network often happen according to randomized protocols, such as broadcast [20] or symmetric gossip strategies [33], described in Section 1.3.3. The use of randomized protocols avoids the need of cumbersome communication scheduling, reduces the need of time-synchronization and may allow to reduce power consumption. A further cause of randomness in the communication is the potential unpredictably of the environment where these protocols are implemented: packet losses and collisions are in fact rather common in a sensor network. Moreover nodes failures, arrivals and departures are common events in the large networks under study.

To take these randomnesses into account, in this chapter the estimate fusion will be implemented as m randomized consensus steps. This means to assume that the consensus matrix, rather than being constant as in [40], is drawn at each time instant form an alphabet of matrices compatible with the graph $\mathcal{G}$. For this reason, the performance analysis carried out in [40] does not apply any longer.

This chapter is therefore devoted to the analysis of a distributed Kalman filtering algorithm that makes use of randomized communication strategies, focusing in particular on symmetric gossip. A mean square performance analysis is carried out and an upper–bound on the estimation error variance is derived. This upper–bound is a good performance assessment index and it is assumed therefore as a cost function to be minimized. We will moreover show that problem of minimizing this cost function by choosing the Kalman gain and the selection probability is convex in each of the two variables separately although it is not jointly convex.

Before to move on, we would like to point out that Distributed Kalman filter has attracted the attention of many researchers. Besides the already cited ones, we mention also the works [64], [65] and [66] where a belief propagation approach to distributed Kalman filtering has been proposed.

Chapter organization

The chapter is organized as follows: In Section 5.1 we give a detailed formulation of the estimation problem under consideration and in Section 5.2 we introduce the estimation algorithm under analysis, namely a randomized version of the algorithm proposed in [40]. In Section 5.3.1 a worst case analysis is performed while in Section 5.3.2 the more relevant mean square analysis is carried out. In particular, we give an upper bound on the steady state error variance of the proposed filter, based on the result proposed in Chapter 4. In Section 5.4 we discuss the optimization problem of finding a Kalman gain and a randomized communication strategy, i.e. a selection probability, that minimize a suitable cost function, namely the proposed upper–bound the trace of the steady state error variance. In particular, we show that the optimization problem is convex in each of the two variables separately although it is not jointly convex. In Section 5.5 some simulation results are presented. Finally in Section 5.6 we summarize the results presented in this chapter and discuss some open questions.

5.1 Problem Formulation

Consider a sensor network of N agents, labeled with the elements of the set $\mathcal{V} = \{1 \dots N\}$. Let us describe the communication constraints of the network with a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the edge $(i, j) \in \mathcal{E}$ if and only if i can transmit information to j.

Our goal is to estimate, by means of such a sensor network, a discrete-time scalar random process of the form:

$$x(t+1) = x(t) + w(t)$$

where $w(t)$, known as model noise, is a white noise with variance q. Each node i of the network can collect noisy measurements the state $x(t)$:

$$y_i(t) = x(t) + v_i(t) \tag{5.1}$$

where the measurement noise, $v_i(t)$, is a white noise with variance r_i. It is reasonable to assume that sensors are affected by N independent measurement noises, all independent also from the model noise $w(t)$. Moreover, for simplicity, we consider a network of identical devices, with all nodes having equally reliable sensors, that is, we restrict our analysis to the case $r_i = r$ $\forall i \in \mathcal{V}$.
For ease of notation collect all the N measurements in a vector $y(t) = [y_1(t), \dots, y_N(t)]^T$ and all measurement noises in a vector $v(t) = [v_1(t), \dots, v_N(t)]^T$. Then (5.1) can then be rewritten as:

$$y(t) = \mathbb{1}x(t) + v(t).$$

Since we assumed that all measurement noises are independent and identically distributed $\mathbb{E}[v(t)v^T(t)] = rI_N$.

5.2 Proposed Algorithm

The estimation algorithm we analyze in this work is a randomized version of the one that has been analyzed in [40]. It consists of 2 stages. The first stage is a Kalman–like estimate update. At each time instant, each node collects its own measurement $y_i(t)$ and updates its estimate $\hat{x}_i(t)$:

$$\hat{x}_i^{loc}(t+1) = l\hat{x}_i(t) + (1-l)y_i(t).$$

where $l \in (0, 1)$ is the Kalman gain. Since $\hat{x}_i^{loc}(t)$ has been updated using only local information, it is called *local estimate*. Again, for ease of notation, we collect all the local estimate in the vector $\hat{x}_{loc}(t) = [\hat{x}_1^{loc}(t), \dots, \hat{x}_N^{loc}(t)]$.

The second phase of the algorithm prescribes that during two consecutive measurements, nodes exchange information with their neighbors to improve their local estimates. In contrast with the previous phase, the outcome of this second phase is called *regional estimate*, $\hat{x}_i^{reg}(t)$, or simply $\hat{x}_i(t)$. Once again, we define the vector $\hat{x}(t) = [\hat{x}_1^{reg}(t), \dots, \hat{x}_N^{reg}(t)]$.

Inspired by the fact that the mean of the local estimates is a sufficient statistic to compute the centralized estimate, we implement the estimate fusion phase with m consensus steps, analogously to what it has been done [40] and [62].

$$\hat{x}(t+1) = Q_{m-1}(t) \dots Q_0(t)\hat{x}_{loc}(t+1) = P(t)\hat{x}_{loc}$$

where $P(t) = Q_{m-1}(t) \dots Q_1(t)$, being product of stochastic matrices, is stochastic.

In [40], it has been analyzed the case of a fixed communication strategy, i.e. $Q_1(t) = \dots = Q_{m-1}(t) = Q \ \forall t$ and therefore $P(t) = P = Q^m \ \forall t$. On the contrary, in this work, to take into account the randomness introduced by the use of random protocols and by unpredictable environments, we assume that for all t and i, $Q_i(t)$ is drawn from an alphabet $\{Q_\alpha, \ \alpha \in \mathbb{A}\}$ of stochastic matrices compatible with the graph $\mathcal{G}$. We will call selection probability, $p_{\mathbb{A}} = \{p_\alpha, \ \alpha \in \mathbb{A}\}$, the probability measure on the set $\{Q_\alpha, \ \alpha \in \mathbb{A}\}$, where p_α is the probability that Q_α is drawn. Therefore $Q_i(t)$ and consequently $P(t)$ are i.i.d. random (matrix–valued) processes.

It is reasonable to assume that $\forall i \ Q_i(t)$ is independent form $w(s)$ and $v(r) \ \forall t, s, r$.

In this chapter we will focus on the case of symmetric matrices alphabets: $Q_\alpha = Q_\alpha^T \ \forall \alpha \in \mathbb{A}$ and in particular on the symmetric gossip case.

5.3 Algorithm Analysis

Let us define the local and the regional estimation error:

$$\tilde{x}_{loc}(t) = \mathbb{1}x(t) - \hat{x}_{loc}(t) \qquad \text{and} \qquad \tilde{x}(t) = \mathbb{1}x(t) - \hat{x}(t).$$

After simple computations one gets the following description of the error time evolution:

$$\begin{aligned} \tilde{x}_{loc}(t+1) &= (1-l)\tilde{x}(t) + lw(t) + \mathbb{1}v(t) \\ \tilde{x}(t+1) &= P(t)\tilde{x}_{loc}(t+1) \\ &= (1-l)P(t)\tilde{x}(t) + lP(t)w(t) + \mathbb{1}v(t). \end{aligned}$$

First of all, we will show that the error mean tends to zero when t goes to infinity, for all possible sequence of extracted communication matrix $\left\{P(t)\right\}_{t=1,2,\dots}$. To this aim define the local and regional mean error to be

$$\mu_{loc}(t) = \mathbb{E}\left[\tilde{x}_{loc}(t)|\{P(s)\}_{s=1\dots t}\right] \qquad \mu(t) = \mathbb{E}\left[\tilde{x}(t)|\{P(s)\}_{s=1\dots t}\right].$$

Recalling that w and v are white zero–mean noises, one gets that

$$\begin{aligned} \mu_{loc}(t+1) &= \mathbb{E}\left[(1-l)\tilde{x}(t) + lw(t) + \mathbb{1}v(t)|\{P(s)\}_{s=1\dots t}\right] = (1-l)\mu(t) \\ \mu(t+1) &= \mathbb{E}\left[P(t)\tilde{x}_{loc}(t+1)|\{P(s)\}_{s=1\dots t}\right] \\ &= P(t)\mu_{loc}(t+1) = (1-l)P(t)(t). \end{aligned} \tag{5.2}$$

Proposition 18.
If $|1-l|<1$, for any sequence $\left\{P(t)\right\}_{t=1,2,\dots}$,

$$\lim_{t\to+\infty} \mu_{loc}(t) = 0 \qquad \textit{and} \qquad \lim_{t\to+\infty} \mu(t) = 0.$$

Proof.
Let us study $||\mu(t)||_\infty$, where we recall that, given a vector $\mu \in \mathbb{R}^N$,

$$||\mu||_\infty = \max\{|\mu_1|, \dots, |\mu_N|\}.$$

Given a matrix $M \in \mathbb{R}^{N\times N}$ we consider the induced norm

$$||M||_\infty = \max_{||x||_\infty=1} \left\{ \ ||Mx||_\infty \ \right\}.$$

From the definition it follows immediately that $||Mx||_\infty \leq ||M||_\infty ||x||_\infty$.
Note moreover that, for any stochastic matrix P,

$$||P||_\infty = 1.$$

In fact, for any x such that $||x||_\infty = 1$, i.e. such that $|x_i| \leq 1$, we have that

$$|(Px)_j| = |\sum_{j=1}^N P_{ij}x_j| \leq \sum_{j=1}^N |P_{ij}||x_j| \leq \sum_{j=1}^N |P_{ij}| = \sum_{j=1}^N P_{ij} = 1.$$

Therefore $||Px||_\infty \leq 1$. Moreover, since $||\mathbb{1}||_\infty = 1$ and $P\mathbb{1} = \mathbb{1}$, we have that $||P||_\infty = 1$.

From this fact it follows that

$$\begin{aligned} ||\mu(t)||_\infty &= ||(1-l)^t P(t)P(t-1)\dots P(0)\mu(0)||_\infty \\ &\leq |1-l|^t ||P(t)P(t-1)\dots P(0)||_\infty ||\mu(0)||_\infty \\ &= |1-l|^t ||\mu(0)||_\infty. \end{aligned}$$

If $|1-l|<1$, $||\mu(t)||_\infty \to 0$. We have moreover that

$$||\mu_{loc}(t+1)||_\infty = (1-l)||\mu(t)||_\infty < ||\mu(t)||_\infty.$$

Hence $||\mu(t)||_\infty \to 0$ implies $||\mu_{loc}(t)||_\infty \to 0$. □

One has then to wonder about the behavior of the estimation error variance.

A first question one could ask is if the estimation error variance remains always bounded, independently on sequence of communication matrices $\{P(t)\}_{t=1,2,\dots}$.
One might wonder then if communications always reduce the error variance, that is, if for any outcome of the sequence of communication matrices $\{P(t)\}_{t=1,2,\dots}$ the error variance is reduced with respect to the no communication case: $P(t)=I\ \forall t$.
Both these two questions, concerned with the worst case performances, are treated in Section 5.3.1. Nevertheless, to consider the worst of all possible realizations of the communication matrices leads to a very pessimistic characterization of the algorithm. For this reason, it is more significant to analyze of the mean estimation error variance . We will consider such a problem in Section 5.3.2, where we will give an upper bound on the steady state value of the expected error variance.

5.3.1 Worst Case Analysis

Let us define the local and regional error variance, conditional to the fact that a certain sequence of matrices $\{P(t)\}$ occurred:

$$\Sigma^c_{loc}(t|\{P(s)\}_{s=1,2,\dots}) = \mathbb{E}\left[\tilde{x}_{loc}(t)\tilde{x}^T_{loc}(t)\middle|\ \{P(s)\}_{s=1\dots t}\right]$$
$$\Sigma^c(t|\{P(s)\}_{s=1,2,\dots}) = \mathbb{E}\left[\tilde{x}(t)\tilde{x}^T(t)\middle|\ \{P(s)\}_{s=1\dots t}\right].$$

where the superscript c stands for conditional. For ease of notation, in the following we will refer to the above matrices simply as $\Sigma^c_{loc}(t)$ and $\Sigma^c(t)$. Note that the elements $\Sigma^c_{loc_{ii}}$ and Σ^c_{ii} represent the local and regional estimation error variances of the j-th node.

The analysis is carried out deriving two upper-bounds valid for any possible sequence of matrices $\{P(t)\}$, even the worst possible case.

In the first result we will consider the node with the largest estimation error, $\max_{i=1,\dots,N}\Sigma_{i,i}$, and we will show that communication improves the quality of its estimation. To this aim we will study $||\Sigma^c(t)||_{\max}$, defined as

$$||\Sigma^c(t)||_{\max} = \max_{i,j=1,\dots,N}|\Sigma^c_{ij}(t)|.$$

In fact, for any symmetric positive matrix $\Sigma > 0$ the largest element in absolute value is always on the diagonal[1],

$$||\Sigma||_{\max} = \max_{i,j=1,\dots,N} |\Sigma_{i,j}| = \max_{i=1,\dots,N} |\Sigma_{i,i}|.$$

In the second result we will consider the average quality of the estimation among the nodes:

$$\frac{1}{N}\sum_{i=1}^{N} \Sigma^c_{ii} = \mathrm{tr}\Sigma^c.$$

We will prove that this parameter is always improved by communication if the matrices $P(t)$ are doubly stochastic, while, if the matrices are only stochastic, it might occasionally get worsen by communication.

First of all, let us derive a recursive formula for $\Sigma^c_{loc}(t)$ and $\Sigma^c(t)$. Recalling that w and v are independent for all t and noting that $\tilde{x}(t)$ is independent on $w(t)$ and $v(t)$, since in depends only on $w(s), v(s)$ for $s = 1 \dots t-1$, one can easily see that

$$\begin{aligned}\Sigma^c_{loc}(t+1) &= (1-l)^2\Sigma^c(t) + l^2R + q\mathbb{1}\mathbb{1}^T \\ &= (1-l)^2P(t)\Sigma^c_{loc}(t)P^T(t) + l^2R + q\mathbb{1}\mathbb{1}^T \end{aligned} \tag{5.3}$$

$$\Sigma^c(t+1) = P(t)\Sigma^c_{loc}(t+1)P^T(t) \tag{5.4}$$

$$= (1-l)^2P(t)\Sigma^c(t)P(t) + l^2P(t)RP^T(t) + q\mathbb{1}\mathbb{1}^T. \tag{5.5}$$

The two previous matrices have to be compared with Σ_{nc}, the estimation error variance when no communications occur:

$$\Sigma_{nc} = \Sigma^c(t|\{P(s) = I\ \forall s\}) = \Sigma^c_{loc}(t|\{P(s) = I\ \forall s\}).$$

Σ_{nc} is described by the following recursive equation

$$\Sigma_{nc}(t+1) = (1-l)^2\Sigma_{nc}(t) + l^2R + q\mathbb{1}\mathbb{1}^T \tag{5.6}$$

with initial condition $\Sigma_{nc}(0) = \Sigma^c(0) = \Sigma^c_{loc}(0)$.
Equation (5.6) describes the evolution of a linear time invarying system, which is stable if $l < 1$. From (5.6) it can be easily computed a steady state value for Σ_{nc},

$$\Sigma_{nc}(\infty) = \frac{l^2R + q\mathbb{1}\mathbb{1}^T}{1-(1-l)^2}.$$

[1] Otherwise there would exist an entries $\Sigma_{i,j}$, $i \neq j$ such that $|\Sigma_{i,j}| > |\Sigma_{i,i}|$ and $|\Sigma_{i,j}| > |\Sigma_{j,j}|$. But in this case the minor $\Sigma_{i,i}\Sigma_{j,j} - \Sigma^2_{i,j} = |\Sigma_{i,i}||\Sigma_{j,j}| - |\Sigma_{i,j}|^2 < 0$, which is impossible since $\Sigma > 0$ by hypothesis.

On the contrary equations (5.3) and (5.5) represent the evolution of two time varying systems for which it is not even meaningful to speak about steady state.
Nevertheless, it holds the following result, that links $\Sigma^c(t)$ and $\Sigma^c_{loc}(t)$,$\Sigma_{nc}(t)$:

Proposition 19. *For any sequence* $\left\{P(t)\right\}_{t=1,2,\ldots}$,

$$||\Sigma^c(t)||_{\max} \leq ||\Sigma^c_{loc}(t)||_{\max} \leq ||\Sigma_{nc}(t)||_{\max}$$

Proof.
Note, first of all, that for any stochastic matrix P and for any matrix M:

$$||PM||_{\max} \leq ||M||_{\max}$$

In fact, since $M_{zj} \leq ||M||_{\max}\ \forall z, j$, we have that:

$$\begin{aligned} ||PM||_{\max} = \max_{i,j}\left\{\sum_{z=1}^{N} P_{iz}M_{zj}\right\} &\leq \max_{i,j}\left\{\sum_{z=1}^{N} P_{iz}||M||_{\max}\right\} \\ &= \max_{i,j}\left\{\sum_{z=1}^{N} P_{iz}\right\}||M||_{\max} \end{aligned}$$

but, since P is stochastic, $\sum_{z=1}^{N} P_{iz} = 1\ \forall i$.

Moreover, since $||M||_{\max} = ||M^T||_{\max}$,

$$||MP^T||_{\max} = ||PM^T||_{\max} \leq ||M^T||_{\max} = ||M||_{\max}$$

therefore

$$||PMP^T||_{\max} \leq ||M||_{\max}, \tag{5.7}$$

The fact that $||\Sigma^c(t)||_{\max} \leq ||\Sigma^c_{loc}(t)||_{\max}$ follows then directly form (5.4) and (5.7).

We will prove that $||\Sigma^c_{loc}(t)||_{\max} \leq ||\Sigma_{nc}(t)||_{\max}$ by induction.
The initial step is trivial since,by definition, $\Sigma^c(0) = \Sigma^c_{loc}(0) = \Sigma_{nc}(0)$.
Note furthermore that $\Sigma^c_{loc}(t)$ and $\Sigma_{nc}(t)$, being variance matrices, are both

symmetric and positive definite.

$$
\begin{aligned}
& ||\Sigma^c_{loc}(t)||_{\max} \leq ||\Sigma_{nc}(t)||_{\max} \\
\Leftrightarrow \quad & \max_{ij}\left\{[\Sigma^c_{loc}(t)]_{i,j}\right\} \leq \max_{ij}\left\{[\Sigma_{nc}(t)]_{i,j}\right\} \\
\Leftrightarrow \quad & \max_{i}\left\{[\Sigma^c_{loc}(t)]_{i,i}\right\} \leq \max_{i}\left\{[\Sigma_{nc}(t)]_{i,i}\right\} \\
\Leftrightarrow \quad & \max_{i}\left\{(1-l)^2\left[P(t-1)\Sigma^c_{loc}(t-1)P(t-1)\right]_{i,i} + l^2 r + q\right\} \leq \\
& \leq \max_{i}\left\{(1-l)^2\left[\Sigma_{nc}(t-1)\right]_{i,i} + l^2 r + q\right\} \\
\Leftrightarrow \quad & \max_{i}\left\{[P(t-1)\Sigma^c_{loc}(t-1)P(t-1)]_{i,i}\right\} \leq \max_{i}\left\{[\Sigma_{nc}(t-1)]_{i,i}\right\} \\
\Leftrightarrow \quad & ||P(t-1)\Sigma^c_{loc}(t-1)P(t-1)||_{\max} \leq ||\Sigma_{nc}(t-1)||_{\max}.
\end{aligned}
$$

Which is true by inductive hypothesis, since

$$||P(t-1)\Sigma^c_{loc}(t-1)P(t-1)||_{\max} \leq ||\Sigma^c_{loc}(t-1)||_{\max} \leq ||\Sigma_{nc}(t-1)||_{\max}.$$

□

Proposition 19 shows that the error variance is always bounded, since it is upper–bounded, in a proper norm, by the output of a linear time–invarying stable system. Proposition 19 shows, moreover, that communication always improves the quality of the estimate, in the sense that by communicating we can at least reduce the variance of the worst estimator.

If we restrict our analysis to the case of $P(t)$ doubly stochastic matrix $\forall t$, it holds true the following result:

Proposition 20. *If we restrict to the case of doubly stochastic matrices $P(t)$ it holds the following*

$$\frac{1}{N}\mathrm{tr}\left(\Sigma^c(t)\right) \leq \frac{1}{N}\mathrm{tr}\left(\Sigma^c_{loc}(t)\right) \leq \frac{1}{N}\mathrm{tr}\left(\Sigma_{nc}(t)\right).$$

Proof.
The proof relies on the following lemma

Lemma 21. *Given any positive semi–definite matrix $A \geq 0$ and any doubly stochastic matrix P one has that*

$$\mathrm{tr}(PAP^T) \leq \mathrm{tr}(A).$$

Proof of Lemma 21.
Since A is positive semi–definite, we have that $\lambda_i(A) = \sigma_i(A)$, and therefore

$$\mathrm{tr}(A) = \sum_{i=0}^{N-1} \lambda_i(A) = \sum_{i=0}^{N-1} \sigma_i(A).$$

Analogously, since also PAP^T is positive semi–definite, $\lambda_i(PAP^T) = \sigma_i(PAP^T)$ and

$$\mathrm{tr}(PAP^T) = \sum_{i=0}^{N-1} \lambda_i(PAP^T) = \sum_{i=0}^{N-1} \sigma_i(QAQ^T).$$

Using (4.3), we get

$$\mathrm{tr}(PAP^T) = \sum_{i=0}^{N-1} \sigma_i(PAP^T) \leq \sum_{i=0}^{N-1} \sigma_i(P)\sigma_i(A)\sigma_i(P^T).$$

Note that, for a doubly stochastic matrix $\sigma_{\max}(P) \leq 1$. In fact, $\sigma_{\max}(P) = \lambda_{\max}(PP^T)$ and since PP^T is stochastic $\lambda_{\max}(PP^T) = 1$. Hence

$$\mathrm{tr}(PAP^T) = \sum_{i=0}^{N-1} \sigma_i(PAP^T) \leq \sum_{i=0}^{N-1} \sigma_i(P)\sigma_i(A)\sigma_i(P^T) \leq \sum_{i=0}^{N-1} \sigma_i(A) = \mathrm{tr}(A)$$

□

Proof of Proposition 20:
The fact that $\frac{1}{N}\mathrm{tr}\left(\Sigma^c(t)\right) \leq \frac{1}{N}\mathrm{tr}\left(\Sigma^c_{loc}(t)\right)$ follows directly from (5.4) and the above lemma 21.
We will prove that $\frac{1}{N}\mathrm{tr}\left(\Sigma^c_{loc}(t)\right) \leq \frac{1}{N}\mathrm{tr}\left(\Sigma_{nc}(t)\right)$ by induction.
The initial step is trivial since,by definition , $\Sigma^c(0) = \Sigma^c_{loc}(0) = \Sigma_{nc}(0)$.
Recall that tr is a linear operator, therefore:

$$\begin{aligned}
&\frac{1}{N}\mathrm{tr}\left(\Sigma^c_{loc}(t)\right) \leq \frac{1}{N}\mathrm{tr}\left(\Sigma_{nc}(t)\right) \\
\Leftrightarrow\quad &\frac{1}{N}\mathrm{tr}\big((1-l)^2 P(t-1)\Sigma^c_{loc}(t-1)P(t-1) + l^2 rI + q\mathbb{1}\mathbb{1}^T\big) \leq \\
&\qquad \leq \frac{1}{N}\mathrm{tr}\left((1-l)^2\Sigma_{nc}(t-1) + l^2 rI + q\mathbb{1}\mathbb{1}^T\right) \\
\Leftrightarrow\quad &(1-l)^2\frac{1}{N}\mathrm{tr}\left(P(t-1)\Sigma^c_{loc}(t-1)P(t-1)\right) + \frac{(l^2 r+q)N}{N} \leq \\
&\qquad \leq \frac{1}{N}(1-l)^2\mathrm{tr}\left(\Sigma_{nc}(t-1)\right) + \frac{(l^2 r+q)N}{N} \\
\Leftrightarrow\quad &\frac{1}{N}\mathrm{tr}\left(P(t-1)\Sigma^c_{loc}(t-1)P(t-1)\right) \leq \frac{1}{N}\mathrm{tr}\left(\Sigma_{nc}(t-1)\right)
\end{aligned}$$

Which is true by inductive hypothesis. In fact, using lemma 21:

$$\frac{1}{N}\mathrm{tr}\left(P(t-1)\Sigma^c_{loc}(t-1)P(t-1)\right) \leq \frac{1}{N}\mathrm{tr}\left(\Sigma^c_{loc}(t-1)\right) \leq \frac{1}{N}\mathrm{tr}\left(\Sigma_{nc}(t-1)\right)$$

□

Note that the hypothesis of doubly stochastic matrices is crucial for Lemma 21 to hold, as it can be easily shown considering the following counterexample:
Counterexample.
Let

$$A = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{and} \qquad P = \begin{bmatrix} 1 & 0 \\ 1/2 & 1/2 \end{bmatrix}$$

where P is stochastic but not doubly stochastic. In this case we have

$$\mathrm{tr}(A) = 11 \leq \mathrm{tr}(PAP^T) = \mathrm{tr}\left(\begin{bmatrix} 10 & 5 \\ 5 & \frac{11}{4} \end{bmatrix}\right) = 12.75$$

□

Lemma 21 and Proposition 20 show that the average estimate quality in the network is guaranteed to be improved at each communication if the consensus matrices are doubly stochastic. Such a guarantee can not be given if the algorithm is only stochastic. This means that using a stochastic but not doubly stochastic consensus matrices, the average estimate quality may occasionally decrease.

As we said, the worst case analysis gives a too pessimistic performance assessment. We proceed therefore in the next section with a mean square analysis.

5.3.2 Mean Square Analysis

Let us study then the variance of the (local and regional) estimation error:

$$\Sigma_{loc}(t) = \mathbb{E}\left[\tilde{x}_{loc}(t)\tilde{x}^T_{loc}(t)\right] \qquad \Sigma(t) = \mathbb{E}\left[\tilde{x}(t)\tilde{x}^T(t)\right],$$

Where in the case the expectation is taken also with respect to the randomness introduced by the random extraction of $P(t)$.
Note that

$$\Sigma_{loc}(t) = \mathbb{E}_P\left[\mathbb{E}\left[\tilde{x}_{loc}(t)\tilde{x}^T_{loc}(t)|\{P(s)\}_{s=1,2,\dots}\right]\right] = \mathbb{E}_P\left[\Sigma^{wc}_{loc}(t)\right]$$
$$\Sigma(t) = \mathbb{E}_P\left[\mathbb{E}\left[\tilde{x}(t)\tilde{x}^T(t)|\{P(s)\}_{s=1,2,\dots}\right]\right] = \mathbb{E}_P\left[\Sigma^{wc}(t)\right].$$

To compute a recursive formula for the evolution of these two matrices, recall that $P(t)$, $w(s)$, $v(u)$ are independent $\forall t, s, u$. Note moreover that $\tilde{x}(t)$ is independent from $P(t)$, $w(t)$ and $v(t)$, since it depends only on $P(s)$, $w(s)$ and $v(s)$ for $s = 1, \ldots, t-1$. One can easily see then that

$$\begin{aligned}\Sigma_{loc}(t+1) &= \mathbb{E}\left[\tilde{x}_{loc}(t+1)\tilde{x}_{loc}^T(t+1)\right] \\ &= (1-l)^2\mathbb{E}\left[P(t)\tilde{x}_{loc}(t)\tilde{x}_{loc}^T(t)P^T(t)\right] + l^2 rI + q\mathbb{1}\mathbb{1}^T.\end{aligned}$$

Since

$$\begin{aligned}&\mathbb{E}\left[P(t)\tilde{x}_{loc}(t)\tilde{x}_{loc}^T(t)P^T(t)\right] = \\ &\quad = \mathbb{E}\left[\mathbb{E}\left[P(t)\tilde{x}_{loc}(t)\tilde{x}_{loc}^T(t)P^T(t)|P(t)\right]\right] \\ &\quad = \mathbb{E}\Big[P(t)\mathbb{E}\left[\tilde{x}_{loc}(t)\tilde{x}_{loc}^T(t)|P(t)\right]P^T(t)\Big] \\ &\quad = \mathbb{E}\left[P(t)\Sigma_{loc}(t)P^T(t)\right]\end{aligned}$$

we have that

$$\Sigma_{loc}(t+1) = (1-l)^2\mathbb{E}\left[P(t)\Sigma_{loc}(t)P^T(t)\right] + l^2 rI + q\mathbb{1}\mathbb{1}^T. \tag{5.8}$$

Similarly, one gets

$$\Sigma(t+1) \;=\; (1-l)^2\mathbb{E}\left[P(t)\Sigma(t)P^T(t)\right] + l^2 r\mathbb{E}\left[P(t)P^T(t)\right] + q\mathbb{1}\mathbb{1}^T. \tag{5.9}$$

Equations (5.8) and (5.9) represent a linear time–invariant system, as it can be more clearly recognized defining $\mathrm{vect}(\Sigma(t)) = s(t)$ and recalling that $\mathrm{vect}(ABC) = (C^T \otimes A)\mathrm{vect}(B)$. Equation (5.9) can in fact be rewritten as:

$$\begin{aligned}\mathrm{vect}(\Sigma(t+1)) = s(t+1) = (1-l)^2\mathbb{E}[P(t)\otimes P(t)]s(t) + \\ l^2 r\mathbb{E}[P(t)\otimes P(t)]\mathrm{vect}(I) + q\mathbb{1}_{N^2}\end{aligned}$$

that is precisely a linear time–invariant system forced by a constant input. Note that

$$\mathbb{E}[P(t)\otimes P(t)]\mathbb{1}_{N^2} = \mathbb{E}[(P(t)\mathbb{1}_N)\otimes(P(t)\mathbb{1}_N)] = \mathbb{1}_{N^2}.$$

Since $\mathbb{E}[P(t)\otimes P(t)]$ is stochastic we have that $(1-l)^2\mathbb{E}[P(t)\otimes P(t)]$ is stable.

For ease of notation let us define the linear operator $\mathcal{L}(M) = \mathbb{E}[Q_i(t)MQ_i^T(t)]$. Note that $\mathrm{vect}\big(\mathcal{L}(M)\big) = \mathbb{E}[Q(t)\otimes Q(t)]\mathrm{vect}(M)$ and that

$$\begin{aligned}&\mathbb{E}[P(t)MP^T(t)] = \\ &\mathbb{E}[Q_{m-1}(t)\ldots Q_0(t)MQ_0^T(t)\ldots Q_{m-1}^T(t)] = \mathcal{L}^m(M)\end{aligned}$$

Equation (5.9) can then be rewritten as

$$\Sigma(t+1) = (1-l)^2\mathcal{L}^m(\Sigma(t)) + l^2r\mathcal{L}^m(I) + q\mathbb{1}\mathbb{1}^T$$

Note moreover that $\mathcal{L}(\mathbb{1}\mathbb{1}^T) = \mathbb{E}(Q\mathbb{1}\mathbb{1}^TQ^T) = \mathbb{1}\mathbb{1}^T$.

We have therefore that, for every initial condition, the system reaches an asymptotically stable equilibrium

$$\begin{aligned}
s(\infty) &= \mathrm{vect}\Sigma(\infty) \\
&= \Big(I - (1-l)^2\mathbb{E}[P(t)\otimes P(t)]\Big)^{-1}\Big(l^2r\mathbb{E}[P(t)\otimes P(t)]\mathrm{vect}(I) + q\mathbb{1}_{N^2}\Big) \\
&= \left(\sum_{i=0}^{+\infty}\Big((1-l)^2\mathbb{E}[P(t)\otimes P(t)]\Big)^i\right) l^2r\mathbb{E}[P(t)\otimes P(t)]\mathrm{vect}(I) + \\
&\qquad + \left(\sum_{i=0}^{+\infty}((1-l)^2\mathbb{E}[P(t)\otimes P(t)]^i\right) q\mathbb{1}_{N^2} \\
&= l^2r\sum_{i=0}^{+\infty}(1-l)^{2i}\mathbb{E}[P(t)\otimes P(t)]^{i+1}\mathrm{vect}(I) + q\sum_{i=0}^{+\infty}(1-l)^{2i}\mathbb{1}_{N^2} \\
&= l^2r\sum_{i=0}^{+\infty}(1-l)^{2i}\mathbb{E}[P(t)\otimes P(t)]^{i+1}\mathrm{vect}(I) + \frac{q}{1-(1-l)^2}\mathbb{1}_{N^2}, \quad (5.10)
\end{aligned}$$

that is,

$$\Sigma(\infty) = l^2r\sum_{i=0}^{+\infty}(1-l)^{2i}\mathcal{L}^{m(i+1)}(I) + \frac{q}{1-(1-l)^2}\mathbb{1}\mathbb{1}^T. \quad (5.11)$$

We are in particular interested in computing $\mathrm{tr}\Sigma(\infty)$.

$$\mathrm{tr}\Sigma(\infty) = l^2r\sum_{i=0}^{+\infty}(1-l)^{2i}\mathrm{tr}\mathcal{L}^{m(i+1)}(I) + \frac{qN}{1-(1-l)^2} \quad (5.12)$$

Unluckily, we did not manage to find a closed form for

$$\sum_{i=0}^{+\infty}(1-l)^{2i}\mathcal{L}^{m(i+1)}(I)$$

but the result we proposed in Chapter 4 can be used to obtain an upper–bound for $\mathrm{tr}\mathcal{L}^{m(i+1)}(I)$ that allows to compute an upper–bound on $\mathrm{tr}\Sigma(\infty)$. In fact, Corollary 13 gives immediately the following

Proposition 22. *Given any symmetric matrix alphabet* $\{Q_\alpha\ \alpha \in \mathbb{A}\}$*, then, for all* $i \in \mathbb{N}$*,*

$$\mathrm{tr}\mathcal{L}^i(I) \leq \mathrm{tr}\left(\mathbb{E}\left[Q^2(t)\right]^i\right). \tag{5.13}$$

This upper bound allows us to analyze the $N \times N$ matrix $\mathbb{E}[Q^2(t)]$ rather then the linear operator $\mathcal{L}$, described by the $N^2 \times N^2$ matrix $\mathbb{E}[Q(t) \otimes Q(t)]$. Note, moreover, that $\mathbb{E}[Q^2(t)]$ can be computed quite easily given a communication strategy and a graph while this is not the case for $\mathbb{E}[Q(t) \otimes Q(t)]$, as it has been remarked in [33].

Using the above mentioned upper–bound we get

$$\begin{aligned}
&\mathrm{tr}\Sigma(\infty) = l^2 r \sum_{i=0}^{+\infty}(1-l)^{2i}\mathrm{tr}\mathcal{L}^{m(i+1)}(I) + \frac{qN}{1-(1-l)^2} \\
&\leq l^2 r \sum_{i=0}^{+\infty}(1-l)^{2i}\mathrm{tr}\mathbb{E}\left[Q^2(t)\right]^{m(i+1)} + \frac{qN}{1-(1-l)^2} \\
&= l^2 r \sum_{i=0}^{+\infty}(1-l)^{2i}\sum_{j=0}^{N-1}\lambda_j\left(\mathbb{E}\left[Q^2(t)\right]^{m(i+1)}\right) + \frac{qN}{1-(1-l)^2} \\
&= \sum_{j=0}^{N-1} l^2 r \sum_{i=1}^{+\infty}(1-l)^{2i-2}\lambda_j^{mi}\left(\mathbb{E}\left[Q^2(t)\right]\right) + \frac{qN}{1-(1-l)^2} \\
&= \sum_{j=0}^{N-1} l^2 r \frac{\lambda_j^m\left(\mathbb{E}\left[Q^2(t)\right]\right)}{1-(1-l)^2\lambda_j^m\left(\mathbb{E}\left[Q^2(t)\right]\right)} + \frac{qN}{1-(1-l)^2}.
\end{aligned}$$

5.4 Optimization

Let us start by studying the optimization problem of finding $l \in (0,1)$ and the probability distribution $p_\alpha = P[Q(t) = Q_\alpha]$ such that $\frac{1}{N}\mathrm{tr}\Sigma(\infty)$ is minimized. Numerical experiments show that this optimization problem is not convex. Indeed, it is not convex even the problem of optimizing $\frac{1}{N}\mathrm{tr}\Sigma(\infty)$ only with respect to p_α and with l fixed. This fact is shown in figure 5.1. Since the problem of optimizing $\mathrm{tr}\Sigma(\infty)$ in not convex we rather consider the problem of minimizing the proposed upper–bound on $\mathrm{tr}\Sigma(\infty)$

$$\frac{1}{N}\mathrm{tr}\Sigma(\infty) \leq J(\{p_\alpha\}_{\alpha\in\mathbb{A}}, l)$$

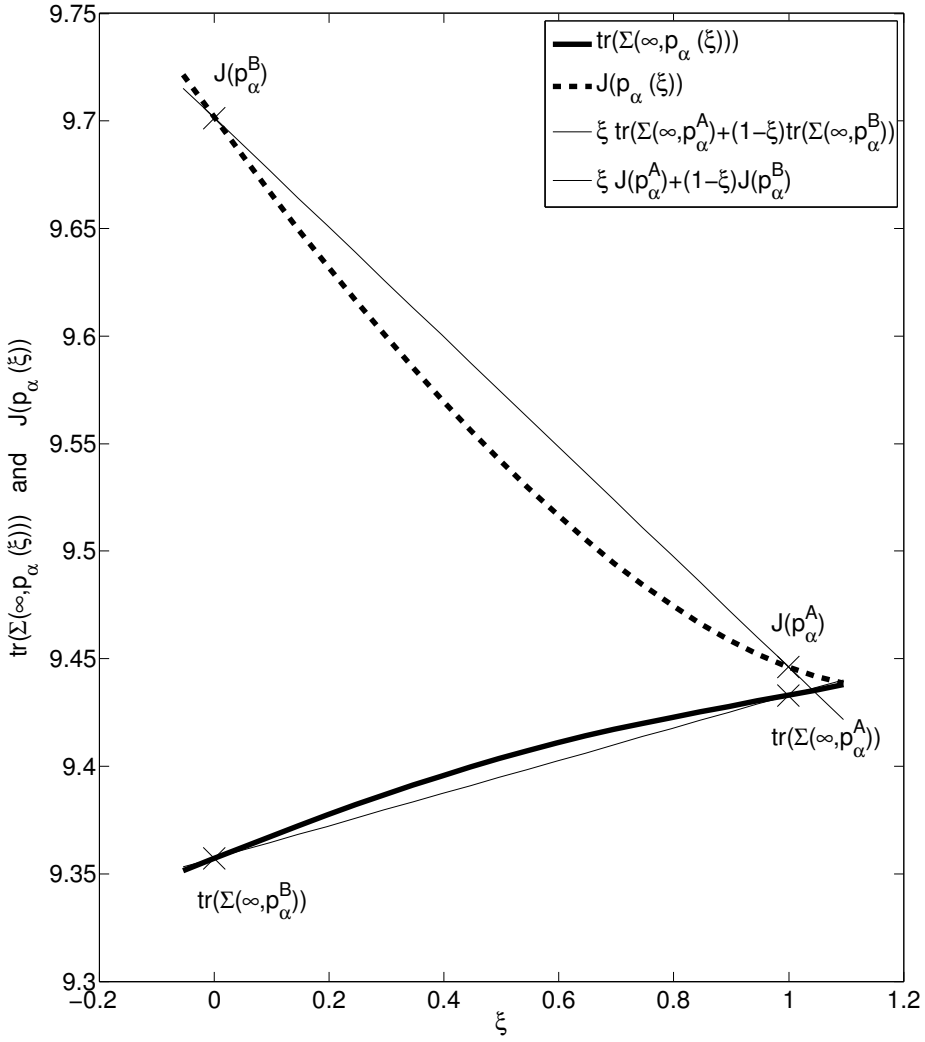


Figure 5.1: Illustration that shows how to find $\{p_\alpha\}_{\alpha\in\mathbb{A}}$ that minimize $\frac{1}{N}\text{tr}\Sigma(\infty, \{p_\alpha\}_{\alpha\in\mathbb{A}})$ is not a convex optimization problem while, on the contrary, the proposed upper-bound $J(p_\alpha)$ is convex (proposition 23). To Both the functions have been represented along the direction connecting two randomly chosen points in the optimization space p_A and p_B.

where

$$J(\{p_\alpha\}_{\alpha\in\mathbb{A}}, l) = \frac{1}{N}\sum_{j=0}^{N-1} l^2 r \frac{\lambda_j^m\left(\mathbb{E}\left[Q^2(t)\right]\right)}{1-(1-l)^2\lambda_j^m\left(\mathbb{E}\left[Q^2(t)\right]\right)} + \\ + \frac{q}{1-(1-l)^2}. \tag{5.14}$$

The above defined function J will be therefore taken as cost function while l and $\{p_\alpha\}_{\alpha\in\mathbb{A}}$, being the design parameters, are going to be our optimization variables. It holds the following result.

Proposition 23. *The problem of minimizing the cost functional $J(\{p_\alpha\}_{\alpha\in\mathbb{A}}, l)$ in the variables l and $\{p_\alpha\}_{\alpha\in\mathbb{A}}$ is convex in l and in $\{p_\alpha\}_{\alpha\in\mathbb{A}}$ separately but it is not jointly convex.*

Proof.
It can be easily verified computing the second derivate, the optimization of the above functional for $l \in (0,1)$ and M fixed is a convex.

In [40] it has been shown that the problem of minimizing the cost function:

$$\tilde{J}(M, l) = \frac{1}{N}\sum_{j=0}^{N-1} l^2 r \frac{\lambda_j^m(M)}{1-(1-l)^2\lambda_j^m(M)} + \frac{q}{1-(1-l)^2} \tag{5.15}$$

over the set of symmetric stochastic matrices, compatible with the graph $\mathcal{G}$, is convex in M for l fixed.

Our cost functional J in (5.14) is the result of the composition of the cost functional $\tilde{J}$ in (5.15) with the map $M(p_\mathbb{A})$:

$$p_\mathbb{A} \longmapsto M(p_\mathbb{A}) = \mathbb{E}_{p_\mathbb{A}}[Q(t)^2] = \sum_{\alpha\in\mathbb{A}} p_\alpha Q_\alpha^2.$$

that goes from $\mathbb{R}^{|\mathbb{A}|}$ the set of stochastic matrices. Note that $M(p_\mathbb{A})$ is linear. Therefore if $\tilde{J}$ in (5.15) is convex, so it is the composed map $J = \tilde{J}(l, M(p_\mathbb{A}))$. We have therefore that the problem of minimizing (5.14) is convex for $l \in (0,1)$ fixed.

To prove that the optimization problem is not jointly convex in l and $\{p_\alpha\}_{\alpha\in\mathbb{A}}$ consider the following counterexample.
Set process parameters as $r = 10$, $q = 1$ and set $N = 2$, $m = 1$.
Let $\mathbb{A} = \{1, 2\}$ and consider the following alphabet $\{Q_\alpha\ \alpha \in \mathbb{A}\}$ of symmetric and stochastic matrices:

$$\{Q_\alpha\ \alpha \in \mathbb{A}\} = \left\{ Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad Q_2 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right\}$$

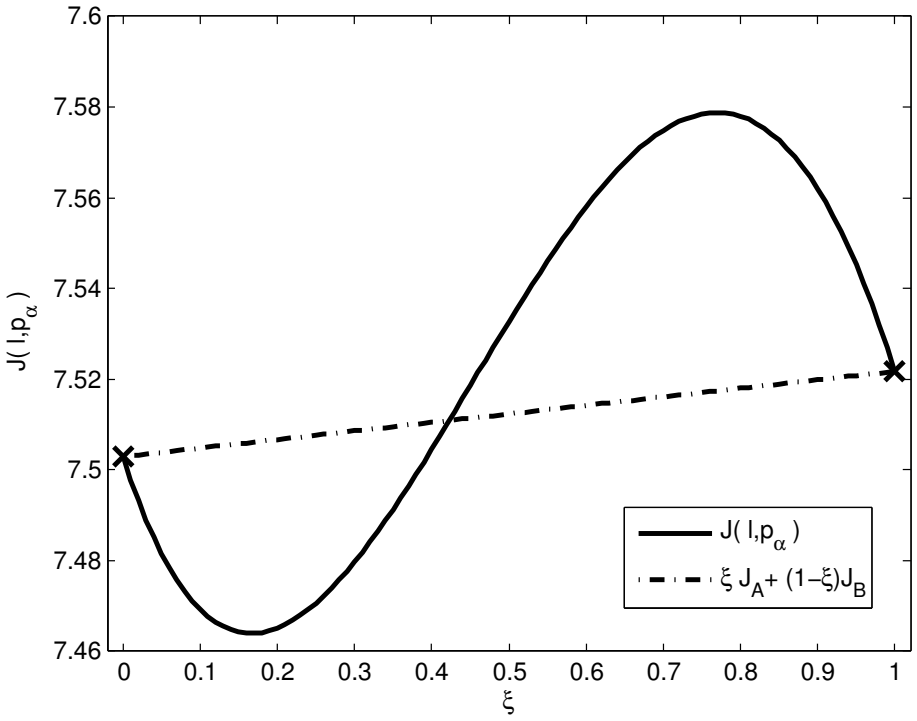


Figure 5.2: Representation of the counterexample presented in the proof of proposition 23. It is depicted the cost function along the direct connecting x_A and x_B: $J(\xi x_A + (1-\xi)x_B)$. Clearly the function is not convex.

Consider the following two points in the optimization space:

$$x_A = (\{p_{A\alpha}\}_{\alpha\in\mathbb{A}}, l_1) = ([0.1 \qquad 0.9],\ 0.75)$$
$$x_B = (\{p_{B\alpha}\}_{\alpha\in\mathbb{A}}, l_2) = ([0.9 \qquad 0.1],\ 0.5)$$

The computation of the cost functional gives us that

$$J(\frac{1}{2}x_A + \frac{1}{2}x_B) = 7.53 \not\leq 7.51 = \frac{1}{2}J(x_A) + \frac{1}{2}J(x_B).$$

This shows that the cost function is not jointly convex in $(\{p_\alpha\}_{\alpha\in\mathbb{A}}, l)$, as depicted also in figure 5.2. □

Therefore, Proposition 23 shows that minimizing J with respect to p_α is an intrinsically easy problem and can be solved using any standard convex optimization technique. Even on a numerical point of view, one can make use of one of the many convex optimization software available.

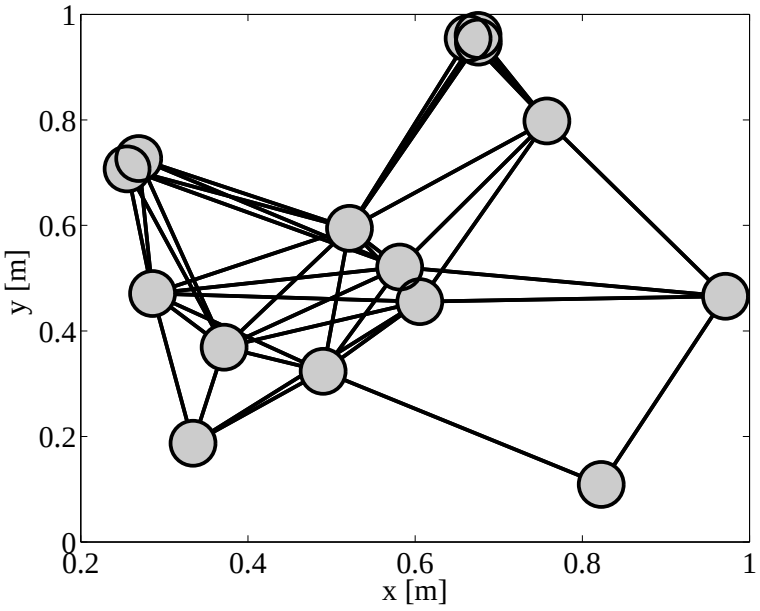


Figure 5.3: Topology of the network under study

On the contrary the non convex problems of minimizing the cost functional $J(\{p_\alpha\}_{\alpha\in\mathbb{A}}, l)$ jointly in the variables l and $\{p_\alpha\}_{\alpha\in\mathbb{A}}$ and of minimizing $\frac{1}{N}\text{tr}\Sigma(\infty)$ are far less simple, the main problem being the fact that an optimization algorithm may end up on a local minimum or present a very low convergence rate.

5.5 Simulation Results

As an example to illustrate the results proposed in the previous sections, we consider a network of $N = 15$ agents. To simulate the behavior of a wireless sensor network, we choose as communication graph a geometric random graph. More precisely, nodes are randomly deployed in a square with side length $1m$ and we assume that two nodes can exchange information if their distance d is less than a visibility radius $r_v = 0.4m$. The topology of the network we obtained is depicted in figure 5.3.

The model noise variance of the process to be estimated is chosen $q = 10$ while the measurement noise variance is chosen $r = 100$.

We consider the symmetric gossip strategy: if the link (i, j) is selected, the corresponding consensus matrix Q_{ij} is:

$$Q_{ij} = I - \frac{1}{2}(e_i - e_j)(e_i - e_j)^T$$

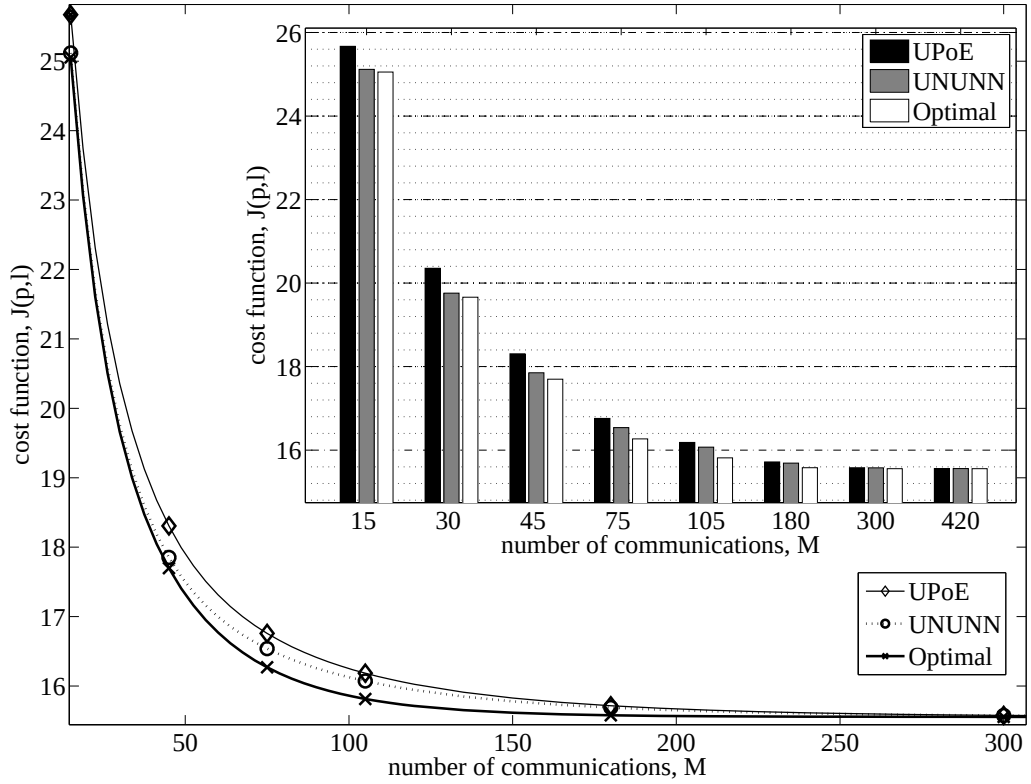


Figure 5.4: Cost function for various selection probabilities

where e_i is the vector having all entries equal to 0 except a 1 in position i. In figure 5.4 it is reported the comparison of three different selection strategies. More precisely it is depicted the value of the cost function (5.14) for $l = 0.5$ and for different values of m, the number of communications between two subsequent measurements. The three different selection strategies we compare are:

- UPoE, Uniform Probability over the Edges: All links are selected with the same probability, $p_{ij}^u = \frac{1}{|\mathcal{E}|}$.

- UNUNN Uniform Node Uniform Neighbor Node: At each time instant one node randomly wakes up, with uniform probably among all nodes. This node picks up randomly one node with uniform probability among all its neighbor nodes. Therefore the selection probability of the link (i, j) is: $p_{ij} = \frac{1}{N} \frac{1}{\text{degree}(i)}$.

- Optimal: p^{opt} is the probability that minimize the cost function (5.14).

Figure 5.4 shows that, at least in this example, optimization does not seem to give a significant improvement in the estimation performance and the easily implementable strategy UNUNN gives performance which is quite close to the optimal selection strategy. Moreover, it can be noted that, over a certain value of m, $\bar{m} \cong 300$, there is no more performance difference between the three strategies. This is due to the fact that when m is large there are enough communications to reach consensus between two subsequent measurements independently of the selection strategy. No further estimation improvement can therefore be obtained. Analogously, if only very few communications are allowed between 2 measurements, the impact of the optimization is very little.

However, it has to be noted that for m in the range $N < m < 10N$, optimization may play an important role. In fact, the same estimation performance is reach by the optimized strategy with a significantly smaller amount of communications with respect to UNUNN. For instance, figure 5.4 shows that the performance achieved by the optimal strategy for $m = 150$ is reached by UNUNN only for $m = 300$.

Finally, we one could wonder if the optimized selection strategy still performs better that UNUNN and UPoE also with respect to the original performance assessment criterion, $\frac{1}{N}\text{tr}\Sigma(\infty)$. All we can say on this issue is that the answer was affirmative in all the simulations we have performed. In figure 5.5 we depict the assessment functions J (thick line) and $\frac{1}{N}\text{tr}\Sigma(\infty)$ (thin line) for the three strategies presented.

5.6 Final Comments and Open Issues

In this chapter we analyzed a randomized version of the distributed Kalman filter proposed in [40]. We carried out a worst-case analysis, that leaded to an over–pessimistic characterization of the algorithm, and a more significant mean–square analysis, proving that the error variance remains bounded and providing an upper–bound for this quantity.

Moreover, we studied the problem of minimizing the proposed upper–bound and we showed that this problem is convex in the selection probabilities and in the Kalman gain separately, but not jointly convex.

Simulations are finally presented. They seem to show that optimization does not give a significant improvement in the estimation performance. Whether or not this is a general fact for the proposed algorithm is an issue that deserves to be explored in detail and that will be object of further investigations.

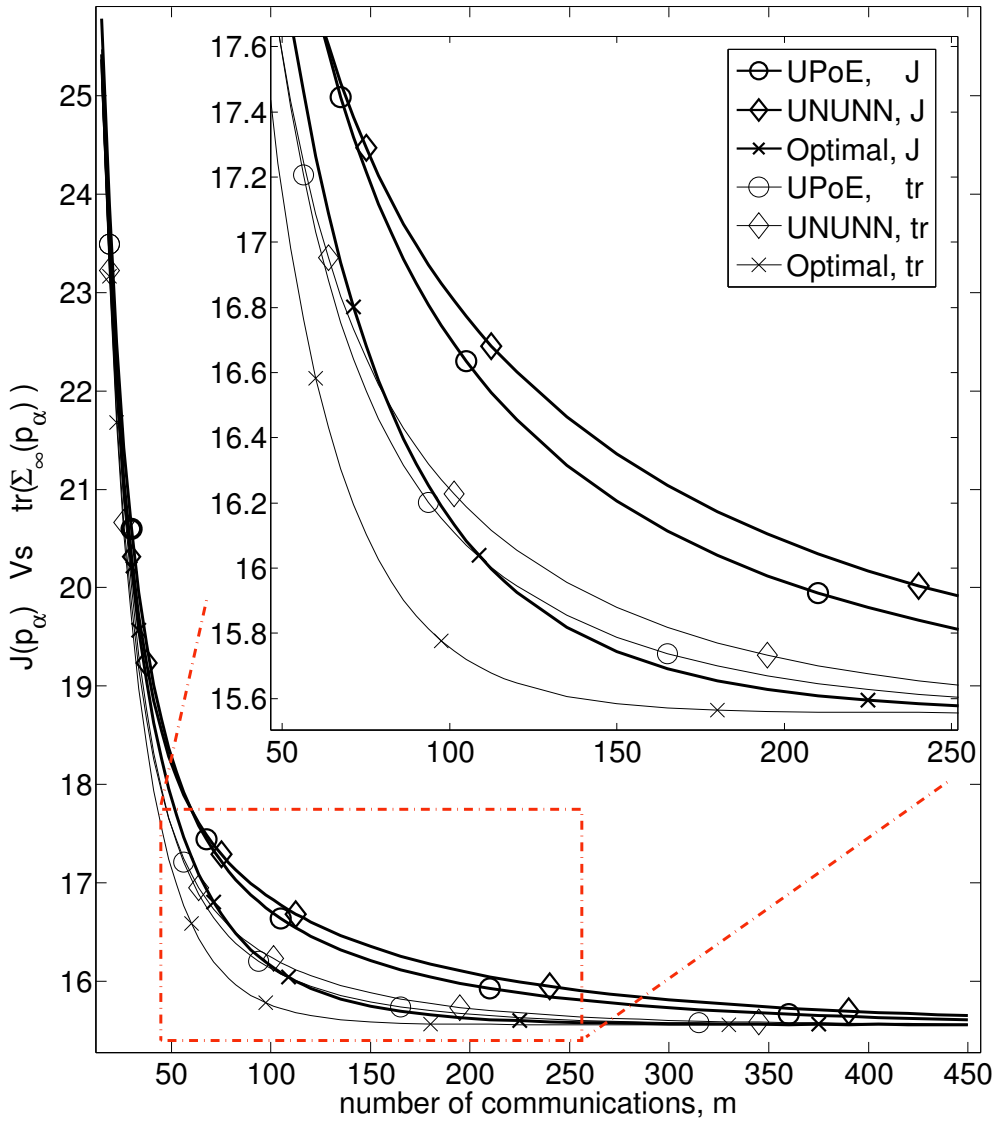


Figure 5.5: J (thick line) and $\frac{1}{N}\mathrm{tr}\Sigma(\infty)$ (thin line) for various selection probabilities

6

A distributed Kalman Smoother for spatially distributed processes

In chapter 5 we considered the problem of estimating, in a distributed fashion, a random process from noisy measurements collected by the nodes of a network. The process to be estimated, $x(t)$, was common for every sensor, that is all the network is trying to compute the same signal. An application that fits in this scenario is, for instance, localization and tracking, where all the network is cooperating to estimate the position of the same moving object.

Yet, many applications involving sensor networks monitoring vast areas do not fit in this framework but rather the set up in which each node i has to estimate a state $x_i(t)$ which is different from the states that other nodes are assigned to estimate, $x_j(t) \neq x_i(t)$ $j \neq i$. Nevertheless there is a correlation between them, $\mathbb{V}(x_i, x_j) \neq 0$, that is exploitable to improve the estimate quality.

As an example, consider a sensor network that has been deployed to monitor rooms' temperature in a building. In this case we are not interested in estimating a common variable, say the average rooms' temperature, but rather we want each node to evaluate the temperature of the zone where it has been placed. The temperature of a room is affected by the temperature of neighbor rooms and one would like to take advantage of this correlation.

Other examples range from environmental and scientific monitoring, where one might spread out in the sea or in the atmosphere cheap sensors to monitor polluting substances' diffusion or environmental changes, to profiling of voltage drop along electrical lines and evaluation sea wave strength. More generally, this framework comes into play whenever it is required to monitor with a sensor network a physical quantity described by partial differential

equations.

The distributed estimation problem in this scenario is rather hard and many works in the literature deals only with the simpler case in which all nodes measure the same quantity, as we did in chapter 5. It is clear that in this new set up we can not use consensus as straightforwardly as we did in chapter 5, since we do not want all nodes to agree upon a common value. We mention here [67], where it is given a remarkable contribution to the scenario of nodes cooperating to estimate different but correlated quantities.

In this chapter we will consider this former scenario of nodes cooperating to estimate different but correlated quantities and, as a first step in the analysis of this challenging set-up, we focus on a simplified case: rather then considering time–varying states $x_i(t)$, we restrict to constant ones: $x_i(t) = x_i$. We have therefore that every node has to estimate x_i from a noisy measurement of it, z_i, and taking advantage of the measure z_j of correlates states x_j, performed by other nodes. This could be for instance the case of a sensor network deployed to estimate the temperature profile of a region.

Moreover we consider the simplest correlation structure between the process at the nodes: x_i is assumed to be independent from x_j conditionally to x_{i-1}, x_{i+1}. This is equivalent to assume that our process is Markov field having as a graphical model a line.

Inference on graphical models such as Markov or Bayesian networks is an important subject of research in many scientific fields such as bioinformatics and signal processing [68, 69]. Some popular methods for inference on such structures include e.g. junction tree algorithms and variational methods as described in [70] as well as Markov chain Monte Carlo methods [71, 72].

In this chapter we focus on simple graphical models defined by state space models underlying Kalman smoothing. Our aim is to reconstruct the state of a Gauss-Markov linear system by means of a decentralized optimization scheme.

The aim of each node is to compute the minimum variance estimate of its own state conditional on all the data acquired by the network. The problem could be solved by using the well known Mayne-Fraser two-filter or the Rauch-Tung-Striebel smoothing algorithm. However, these approaches are essentially serial in nature [73]. An interesting parallel smoothing scheme is instead described in [74]. However, this approach has some important limitations in the scenario of the sensor networks. In fact, after a series of parallel processing of data in subintervals, it calls for a serial and expensive exchange of information between all the nodes of the network.

In this chapter, we present a different distributed smoother since it consists of an iterative scheme where information exchange is limited to nodes which are close to each other.

Chapter organization

The chapter is organized as follows. In section 6.1, the estimation problem is stated and in section 6.2 it is introduced an important notation that will allow us to present the distributed smoothing algorithm, in section 6.3. In section 6.4, we provide a convergence analysis of the algorithm by deriving explicitly the matrix which regulates the dynamics of the error, i.e. the difference between the current estimate and the minimum variance one. In section 6.5, a numerical example regarding the reconstruction of a function and its derivative via cubic smoothing splines is used to test the new algorithm and the theoretical findings. In the case where the state evolves according to a random walk, the convergence rate is explicitly derived as a function of the design parameters of the algorithm, as reported in section 6.6. Interestingly, it is shown that a modest increase in the amount of communications performed at each iteration may lead to an exponential improvement of the performance of the numerical scheme.

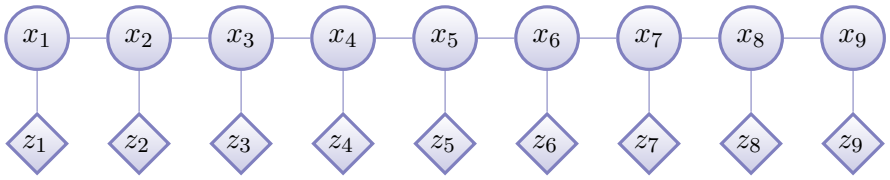


Figure 6.1: Graphical model of the process under study, (6.1) and (6.2), for $N = 9$.

6.1 Problem Description

Consider the graphical model depicted in figure 6.1 where the N non-observable states are denoted by $\{x_k\}$, while $\{z_k\}$ are the noisy measurements. For instance, in the scenario of sensor networks, z_k can be thought of a measurement vector collected by the k-th node whose aim is to estimate x_k. The joint density of $\{x_k\}$ is defined by the following recursive equation:

$$x_k = G_k x_{k-1} + w_k, \qquad \begin{array}{l} w_k \sim \mathcal{N}[0, Q_k], \\ x_0 \sim \mathcal{N}[x^0, Q_0] \end{array} \qquad k = 1, \ \ldots \ , N \tag{6.1}$$

where $G_k \in \mathbb{R}^{n \times n}$ and $w_k \in \mathbb{R}^n$ is zero-mean white Gaussian noise with autocovariance $Q_k \in \mathbb{R}^{n \times n}$. The measurements model is

$$z_k = H_k x_k + v_k, \qquad v_k \sim \mathcal{N}[0, R_k], \tag{6.2}$$

where $H_k \in \mathbb{R}^{m(k) \times n}$ and v_k is zero-mean white normal noise of autocovariance $R_k \in \mathbb{R}^{m(k) \times m(k)}$. We also assume that $\{v_k\}$ and $\{w_k\}$ are all mutually independent.

We would like to obtain an efficient smoothing algorithm where the node in charge of estimating x_k is constrained to exchange information only with its adjacent nodes (i.e. with the nodes handling the states x_{k-1} and x_{k+1}) so as to obtain a high level of parallelism.

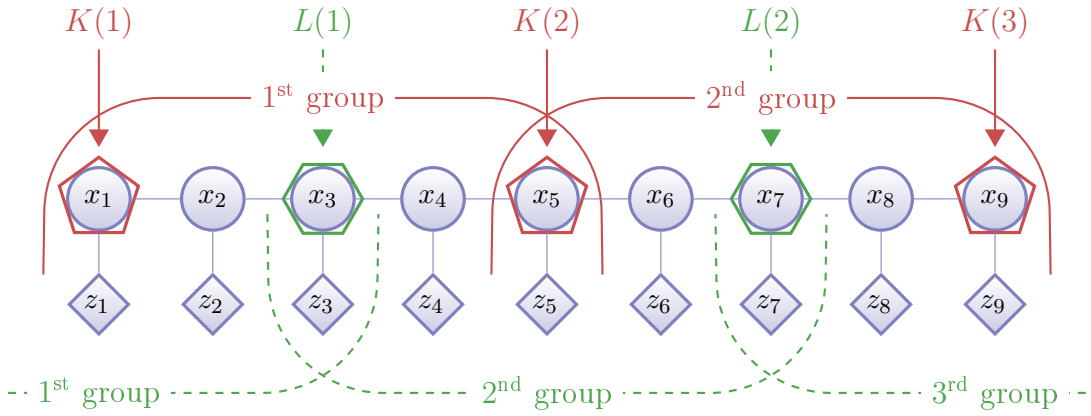


Figure 6.2: Example of the two nodes partitions prescribed by the algorithm. In this case $N = 9, p = 2$ and $J = 4$. The first partition (red solid line) is delimited by nodes $K(j)$ $j = 1, \ldots, p$ while the second partition (green dashed line) is delimited by nodes $L(j)$ $j = 1, \ldots, p$.

6.2 Notation

Let us introduce an important notation that will allow us to present the smoothing algorithm.

As described in detail in the next section, the proposed algorithm prescribes to divide the N nodes in two types of overlapping groups working in parallel.
The first partition consists of p groups which, just for ease of notation, are assumed to contain the same number of nodes, $J+1$, with J an even integer so that

$$N = Jp + 1.$$

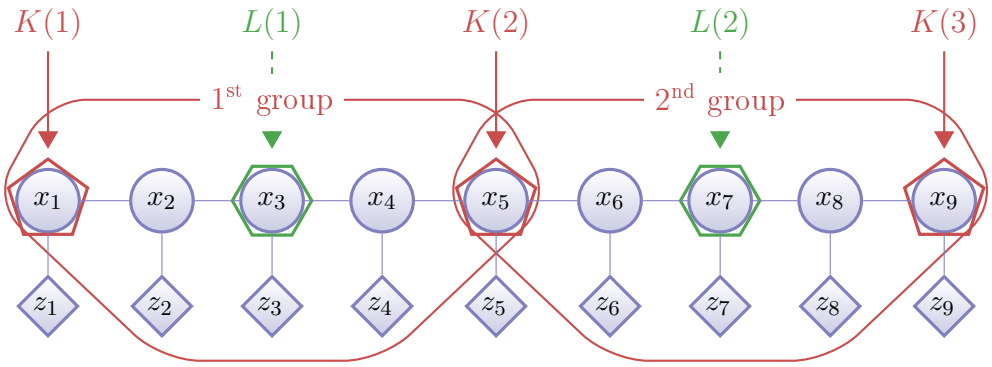
These assumptions can be easily removed without impact on the presented results.

An example is given in Figure 6.2 with $N = 9, p = 2$ and $J = 4$.

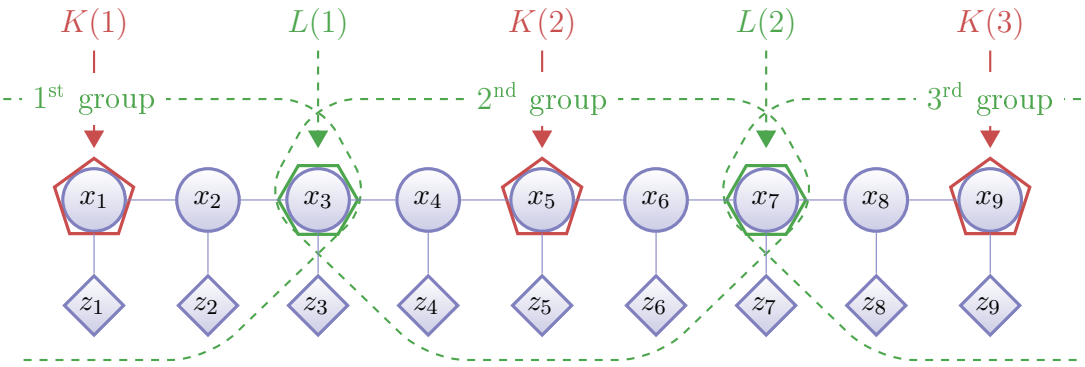
The indices

$$K(j) = 1 + (j-1)J \qquad (j = 1, \cdots, p+1).$$

define the boundaries of these first kind of groups and $K(j)$ is a pointer to the first node of the j-th group that is also the last node of the $(j-1)$-th group. This is graphically shown in of figure 6.2.

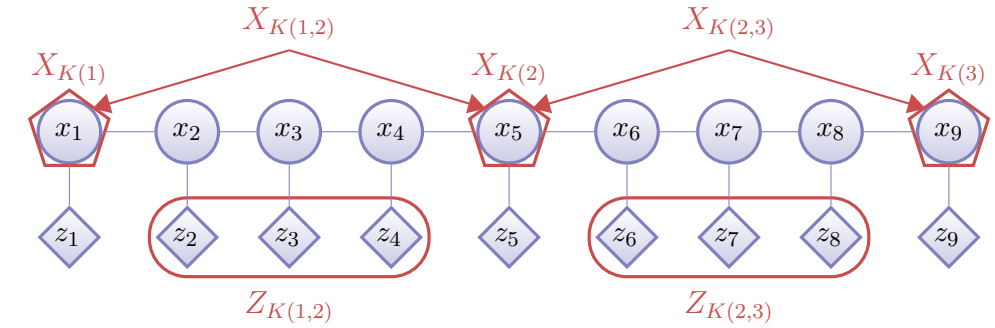


(a) The first partition, delimited by nodes $K(j)$ $j = 1, \ldots, p$.

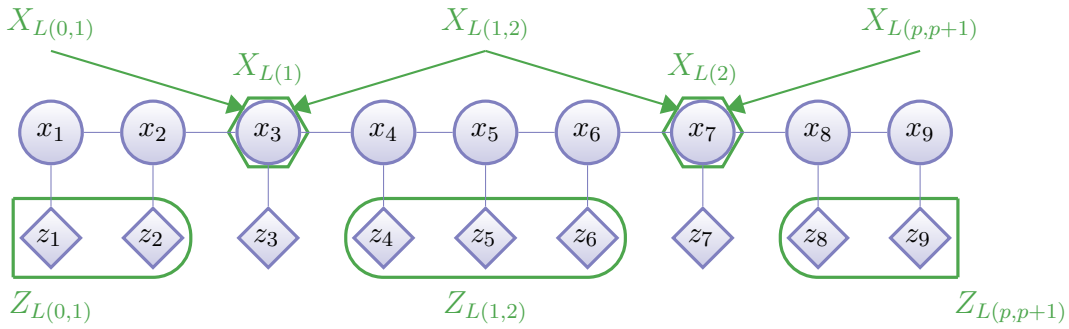


(b) The second partition, delimited by nodes $L(j)$ $j = 1, \ldots, p$.

Figure 6.3: Example of the two nodes partitions prescribed by the algorithm. In this case $N = 9, p = 2$ and $J = 4$.



(a)



(b)

Figure 6.4: Example of the notation introduced. In this case $N = 9, p = 2$ and $J = 4$.

Let X denote the entire state sequence $\{x_k : k = 1, \ldots, N\}$. We define

$$\begin{aligned} X_K &= \{x_{K(1)}, x_{K(2)}, \ldots, x_{K(p+1)}\} \\ X_{K(j,j+1)} &= \{x_{K(j)}, x_{K(j+1)}\} \qquad j = 1, \cdots, p \end{aligned}$$

as illustrated in figure 6.4.
The second partition contains $p+1$ groups whose boundaries are defined by

$$L(j) = 1 + J/2 + (j-1)J \qquad (j = 2, \cdots, p)$$

that are also the middle nodes of the groups present in the previous partition, see figure 6.2. Also in this case, it is useful to define

$$\begin{aligned} X_L &= \{x_{L(1)}, x_{L(2)}, \ldots, x_{L(p)}\} \\ X_{L(j,j+1)} &= \{x_{L(j)}, x_{L(j+1)}\} \qquad j = 1, \cdots, p-1 \\ X_{L(0,1)} &= \{x_{L(1)}\} \\ X_{L(p,p+1)} &= \{x_{L(p+1)}\} \end{aligned}$$

as illustrated in figure 6.4.

Now, let $Z = \{z_k : k = 1, \ldots, N\}$. As it will be clear in the sequel, it is useful to define the following subsets of Z 'contained' between, and not including, the indexes j and k

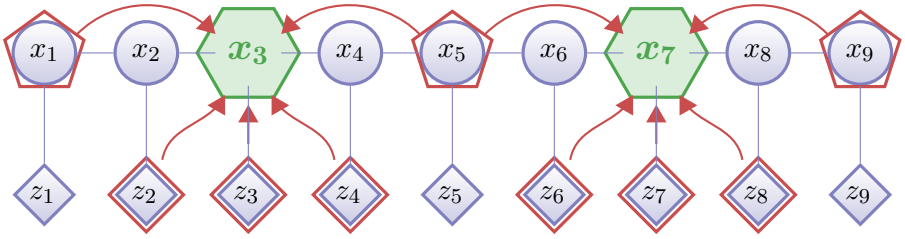
$$\begin{array}{rcll} Z_{K(j,j+1)} & = & \{z_{K(j)+1}, \cdots, z_{K(j+1)-1}\} & j = 1, \cdots, p \\ Z_{L(j,j+1)} & = & \{z_{L(j)+1}, \cdots, z_{L(j+1)-1}\} & j = 1, \cdots, p-1 \\ Z_{L(0,1)} & = & \{z_1, z_2, \cdots, z_{L(1)-1}\} & \\ Z_{L(p,p+1)} & = & \{z_{L(p)+1}, \cdots, z_{N-1}, z_N\} & \end{array}$$

as shown in figure 6.4.

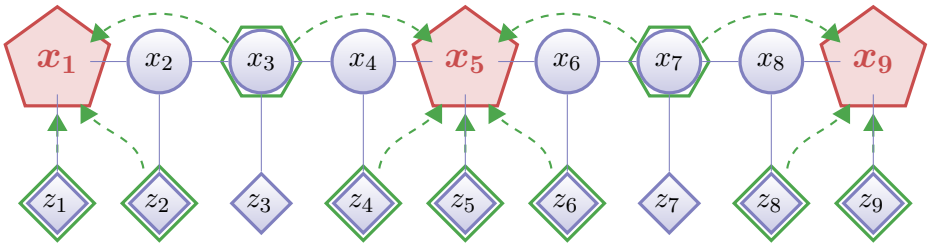
We conclude this section dedicated to the notation recalling that vectors are column vectors, $\mathbb{E}[\cdot]$ denotes the expectation operator, given the random vectors Y and W, $\mathbb{V}[Y, W]$ is their covariance; i.e.,

$$\mathbb{V}[Y, W] = \mathbb{E}\left[(Y - \mathbb{E}[Y])(W - \mathbb{E}[W])^T\right]$$

and we use the notation $\mathbb{V}[Y] = \mathbb{V}[Y, Y]$.
Also we denote compactly $\mathbb{V}[Y|W] = \mathbb{E}[(Y - \mathbb{E}[Y|W])(Y - \mathbb{E}[Y|W])^T]$ the a posteriori estimation–error variance.



(a) **Step 2:** X_3^ℓ and X_3^ℓ are updated as if x_1,x_5 and x_9 were exactly X_1^ℓ,X_5^ℓ,X_9^ℓ.



(b) **Step 3:**X_1^ℓ, X_5^ℓ and X_9^ℓ are updated as if x_3 and x_7 were exactly $X_3^{\ell+1}$, $X_7^{\ell+1}$.

Figure 6.5: Graphical representation the key steps of the proposed algorithm

6.3 Algorithm

The smoothing algorithm is described below.

Step 1. Initialization:
At iteration $\ell = 0$ the second partition is active. Let us consider the $j-th$ group, whose boundaries are nodes $L(j-1)$ and $L(j)$. Node $K(j)$, the middle node of the group, computes an estimate of its state based only the measurements collected by the nodes of its group $Z_{L(j-1,j)}$.

$$x^0_{K(j)} = \mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)}] \qquad \text{for } j = 1, \dots, p+1.$$

Therefore node $K(j)$ is initialized with the best estimate computable using the local measurements.

Step 2:
The first partition is active. Let us consider the $j-th$ group, whose boundaries are nodes $K(j)$ and $K(j+1)$. Node $L(j)$, the middle node of the group, updates the estimate $X^\ell_{L(j)}$ of its state $x_{L(j)}$. In doing so it uses only the measurements collected by the nodes of its group $Z_{K(j,j+1)}$ and assumes that the states $x_{K(j)}$ and $x_{K(j+1)}$ have been exactly estimated at the previous step by nodes $K(j)$ and $K(j+1)$. Namely, it computes its estimate as if

$x_{K(j)}$ and $x_{K(j+1)}$ were exactly $X^{\ell}_{K(j)}$ and $X^{\ell}_{K(j+1)}$. Hence,

$$X^{\ell+1}_{L(j)} = \mathbb{E}[x_{L(j)} \mid Z_{K(j,j+1)} \, , X_{K(j,j+1)} = X^{\ell}_{K(j,j+1)}],$$

Step 3:

The second partition is active. Let us consider the $j-th$ group, whose boundaries are nodes $L(j-1)$ and $L(j)$. Analogously to the previous step, node $K(j)$, the middle node of the group, updates the estimate $X^{\ell}_{K(j)}$ of its state $x_{K(j)}$. In doing so it uses only the measurements collected by the nodes of its group $Z_{L(j-1,j)}$ and assumes that the states $x_{L(j-1)}$ and $x_{L(j)}$ have been exactly estimated at the previous step by nodes $L(j-1)$ and $L(j)$. Namely, it computes its estimate as if $x_{L(j-1)}$ and $x_{L(j)}$ were exactly $X^{\ell+1}_{L(j-1)}$ and $X^{\ell+1}_{L(j+1)}$. Hence,

$$X^{\ell+2}_{K(j)} = \mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)} \, , X_{L(j-1,j)} = X^{\ell+1}_{L(j-1,j)}],$$

Step 4:

The iteration between step 2 and step 3 leads to an improvement of the estimate quality, as we will see more precisely in section 6.4.
Then, fix the exit parameter ε and
if there has been a significant improvement in the estimate

$$|x^{\ell+2}_{K(j)} - x^{\ell}_{K(j)}| \geq \varepsilon$$

then set $\ell = \ell + 2$ and go to step 2.
Otherwise one can assume that the optimal estimate is achieved $X^{\ell+2}_{K(j)} = \mathbb{E}[x_{K(j)} \mid Z]$ and the algorithm can move to Step 5.

Step 5. Final Step:

To conclude, note that from Markov property we have

$$\mathbb{E}[\, x_i \mid Z] = \mathbb{E}[x_i \mid Z_{K(j,j+1)}, X_{K(j,j+1)}].$$

for any node i of the j-th group.
Therefore, compute all the other estimates of the j-th group as

$$\mathbb{E}[x_i \mid Z_{K(j,j+1)} \, , X_{K(j,j+1)} = X^{\ell+2}_{K(j,j+1)}].$$

This concludes the algorithm. □

We point out that in order to perform step 2 node $L(j)$ needs only local informations, $Z_{K(j,j+1)}$ and $X^{\ell}_{K(j,j+1)}$, which can be retrieved from neighbor

nodes. Moreover, once $L(j)$ has received $Z_{K(j,j+1)}$, the node can store it in memory, so that at each iteration only $X^{\ell}_{K(j,j+1)}$ is need. Analogous considerations hold for step 3.

To rewrite compactly the algorithm, note that, from the Markov property for the state sequence, it follows

$$\mathbb{E}[\, x_{L(j)} \mid Z \,,\, X_K] = \mathbb{E}[x_{L(j)} \mid \, Z_{K(j,j+1)} \,,\, X_{K(j,j+1)}] \tag{6.3}$$

$$\mathbb{E}[x_{K(j)} \mid Z \,,\, X_L \,] = \mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)} \,,\, X_{L(j-1,j)}] \tag{6.4}$$

The algorithm becomes then

Algorithm 1. Given an algorithm convergence criteria ε, perform

1. **Initialization:** Set $\ell = 0$ and X^0_K

$$x^0_{K(j)} = \mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)}] \qquad \text{for } j = 1, \ldots, p+1.$$

2. Compute $\quad X^{\ell+1}_L = \mathbb{E}[X_L \mid Z \,,\, X_K = X^{\ell}_K],$

   i.e., X^{ℓ}_L is updated as if X_K was exactly X^{ℓ}_1, X^{ℓ}_K.

3. Compute $\quad X^{\ell+2}_K = \mathbb{E}[X_K \mid Z \,,\, X_L = X^{\ell+1}_L],$

   i.e., X^{ℓ}_K is updated as if X_L was exactly $X^{\ell+1}_L$.

4. **If** $|x^{\ell+2}_{K(j)} - x^{\ell}_{K(j)}| \leq \varepsilon$ for all $j = 1, \ldots, p+1$

   Return $\quad \mathbb{E}[x_k \mid Z \,,\, X_K = X^{\ell+2}_K],$

   as the state estimate for $k = 1, \ldots, N$.

5. Set $\ell = \ell + 2$ and go to step 2

In view of the above equations, the expectation in Step 2 can be computed using p parallel procedures. Each of these parallel procedures solves a smoothing problem over a set of J nodes where the state at the boundaries of the set of nodes is given, as discussed in [75]. The computational complexity of each of these parallel procedures is $O(Jn^3)$. A similar conclusion holds for the expectations in Step 3 and Step 4.

6.4 Convergence analysis

It is worth to recall first of all a well known formula about joint Gaussian vectors (see e.g. [76]), reported in the following lemma.

Lemma 24. *If Y, W are jointly Gaussian random variates, it holds that*

$$\begin{aligned} \mathbb{E}(Y|W) &= \mathbb{E}(Y) + \mathbb{V}(Y,W)\mathbb{V}(W)^{-1}[W - \mathbb{E}(W)] && (6.5)\\ \mathbb{V}(Y|W) &= \mathbb{E}[(Y - \mathbb{E}[Y|W])(Y - \mathbb{E}[Y|W])^T] && (6.6)\\ &= \mathbb{V}(Y) - \mathbb{V}(Y,W)\mathbb{V}(W)^{-1}\mathbb{V}(W,Y) && (6.7) \end{aligned}$$

In the sequel, it is useful to define the following notation for $j = 1, \ldots, p$

$$\begin{aligned} \Xi_j &= \mathbb{V}\left(x_{K(j)}, X_{L(j-1,j)} \mid Z_{L(j-1,j)}\right) \\ &\times \mathbb{V}\left(X_{L(j-1,j)} \mid Z_{L(j-1,j)}\right)^{-1} \\ \Pi_j &= \mathbb{V}\left(x_{L(j)}, X_{K(j,j+1)} \mid Z_{K(j,j+1)}\right) \\ &\times \mathbb{V}\left(X_{K(j,j+1)} \mid Z_{K(j,j+1)}\right)^{-1} \end{aligned}$$

These matrices can be computed using (6.7) (for large values of J it may be more efficient and computationally stable to use [77, corollary 7]).

We use $\delta_K^\ell = X_K^\ell - \mathbb{E}[X_K|Z]$ to denote the error for even values of ℓ. We also use $\delta_{K(j,k)}^\ell$ for the column vector

$$\delta_{K(j,k)}^\ell := (\delta_{K(j)}^\ell, \delta_{K(k)}^\ell)^{\mathrm{T}}$$

We use a similar notation for δ_L^ℓ.

In the following we introduce the matrices $E_1, E_2, \ldots, E_p$, where $E_j \in \mathbb{R}^{n\times 2n}$ if $j \in \{1, p\}$ and $E_j \in \mathbb{R}^{n\times 3n}$ otherwise, where

$$\begin{aligned} E_1 &= \Xi_1\Pi_1 \\ E_j &= \Xi_j \begin{pmatrix} \Pi_j & 0_{n\times n} \\ 0_{n\times n} & \Pi_{i+1} \end{pmatrix} \qquad i = 2, \ldots, p \\ E_{p+1} &= \Xi_{p+1}\Pi_p, \end{aligned}$$

Then, we define the block-tridiagonal matrix R associated with $\{E_i\}$ to be

$$R = \begin{pmatrix} E_1 & & & & \\ & E_2 & & & \\ & \ddots & \ddots & \ddots & \\ & & & E_p & \\ & & & & E_{p+1} \end{pmatrix}, \tag{6.8}$$

hence E_0 specifies the nonzero-entries in the first n rows, E_1 the nonzero-entries in the second n rows and so on. The next proposition characterizes the error dynamics of the distributed smoothing algorithm.

Theorem 25. *Let R be the block-tridiagonal matrix associated with the blocks $\{E_i\}$, as above defined.*

Then, it holds that

1. *the error dynamics of the distributed smoothing algorithm at the nodes $\{K(j)\}$ are regulated by the equation*

$$\delta_K^{\ell+2} = R\delta_K^{\ell} \tag{6.9}$$

2. *R is asymptotically stable, i.e. all its eigenvalues are inside the open complex unit circle*

Proof.
Let's start focusing on the error propagation at a generic node $K(j)$, with $1 < j < p$, when ℓ is 0 or an even number. Step 2 of the algorithm computes

$$X_L^{\ell+1} = \mathbb{E}[X_L \mid Z\ ,\ X_K = X_K^\ell]$$

by means of local computations; see (6.3). Exploiting (6.5), with $Y = x_{L(j)}$ and $W = X_{K(j,j+1)}$ and both conditional on $Z_{K(j,j+1)}$, the linear projection (6.3) admits the following decomposition

$$\begin{aligned}&\mathbb{E}[x_{L(j)} \mid Z_{K(j,j+1)}\ ,\ X_{K(j,j+1)}] = \mathbb{E}[x_{L(j)} \mid Z_{K(j,j+1)}]\\ &+\mathbb{V}(x_{L(j)}, X_{K(j,j+1)} \mid Z_{K(j,j+1)})\\ &\times\mathbb{V}^{-1}(X_{K(j,j+1)} \mid Z_{K(j,j+1)})\\ &\times\left(X_{K(j,j+1)} - \mathbb{E}[X_{K(j,j+1)} \mid Z_{K(j,j+1)}]\right)\end{aligned}$$

Using our definition for Π_j, this becomes

$$\begin{aligned}&\mathbb{E}[x_{L(j)} \mid Z_{K(j,j+1)}\ ,\ X_{K(j,j+1)}] = \mathbb{E}[x_{L(j)} \mid Z_{K(j,j+1)}]\\ &+\Pi_j\left(X_{K(j,j+1)} - \mathbb{E}[X_{K(j,j+1)} \mid Z_{K(j,j+1)}]\right)\end{aligned} \tag{6.10}$$

Taking the expected value $\mathbb{E}[\cdot|Z]$ of both sides of the equation above, we obtain

$$\begin{aligned}&\mathbb{E}[x_{L(j)} \mid Z] = \mathbb{E}[x_{L(j)} \mid Z_{K(j,j+1)}]\\ &+\Pi_j\left(\mathbb{E}[X_{K(j,j+1)} \mid Z] - \mathbb{E}[X_{K(j,j+1)} \mid Z_{K(j,j+1)}]\right)\end{aligned} \tag{6.11}$$

In the places where $X_{K(j,j+1)}$ is a fixed value, substitute $x_{K(j)} = x_{K(j)}^\ell$, and $x_{K(j+1)} = x_{K(j+1)}^\ell$. Note that for these choices,

$$x_{L(j)}^{\ell+1} = \mathbb{E}[x_{L(j)} \mid Z_{K(j,j+1)}\ ,\ X_{K(j,j+1)}]$$

Now, with this choice, subtracting equation (6.11) from (6.10), we obtain

$$\begin{aligned}x_{L(j)}^{\ell+1} - \mathbb{E}[x_{L(j)} \mid Z] &= \Pi_j\left(X_{K(j,j+1)}^\ell - \mathbb{E}[X_{K(j,j+1)} \mid Z]\right)\\ \delta_{L(j)}^{\ell+1} &= \Pi_j\delta_{K(j,j+1)}^\ell\end{aligned} \tag{6.12}$$

We now move to consider Step 3 of the algorithm given by (6.4). We have

$$\begin{aligned}&\mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)}\ ,\ X_{L(j-1,j)}] = \mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)}]\\ &+\mathbb{V}(x_{K(j)}, X_{L(j-1,j)} \mid Z_{L(j-1,j)})\\ &\times\mathbb{V}^{-1}(X_{L(j-1,j)} \mid Z_{L(j-1,j)})\\ &\times\left(X_{L(j-1,j)} - \mathbb{E}[X_{L(j-1,j)} \mid Z_{L(j-1,j)})\right)\end{aligned}$$

Using our definition for Ξ_j, this becomes

$$\begin{aligned}&\mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)}\ ,\ X_{L(j-1,j)}] = \mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)}]\\ &+\Xi_j\left(X_{L(j-1,j)} - \mathbb{E}[X_{L(j-1,j)} \mid Z_{L(j-1,j)})\right)\end{aligned} \tag{6.13}$$

Taking the expected value $\mathbb{E}[\cdot|Z]$ of both sides of the equation above, we obtain

$$\begin{array}{l}\mathbb{E}[x_{K(j)} \mid Z] = \mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)}] \\ +\Xi_j \left(\mathbb{E}[X_{L(j-1,j)} \mid Z] - \mathbb{E}[X_{L(j-1,j)} \mid Z_{L(j-1,j)}]\right)\end{array} \tag{6.14}$$

In the places where $X_{L(j-1,j)}$ is a fixed value, substitute $x_{L(j-1)} = x^{\ell+1}_{L(j-1)}$, and $x_{L(j)} = x^{\ell+1}_{L(j)}$. Note that now for these choices,

$$x^{\ell+2}_{K(j)} = \mathbb{E}[x_{K(j)} \mid Z_{L(j-1,j)}\ , X_{L(j-1,j)}]$$

Now, with this choice, subtracting equation (6.14) from (6.13), we obtain

$$\begin{aligned} x^{\ell+2}_{K(j)} - \mathbb{E}[x_{K(j)} \mid Z] &= \Xi_j \left(X^{\ell+1}_{L(j-1,j)} - \mathbb{E}[X_{L(j-1,j)} \mid Z]\right) \\ \delta^{\ell+2}_{K(j)} &= \Xi_j \delta^{\ell+1}_{L(j-1,j)} \end{aligned}$$

Thus, using equation (6.12), we conclude that

$$\delta^{\ell+2}_{K(j)} = \Xi_j \begin{pmatrix} \Pi_{j-1} & 0 \\ 0 & \Pi_j \end{pmatrix} \begin{pmatrix} \delta^{\ell}_{K(j-1,j)} \\ \delta^{\ell}_{K(j,j+1)} \end{pmatrix}$$

Note that, for $1 < j < N$, the block matrix above is size $n \times 4n$. But since the values $\delta^{\ell}_{K(j)}$ are repeated, one can replace the block matrix by a $n \times 3n$ matrix (as is done in the definition of E_j). Considering $j = 1$ and $j = N$ as special cases (where the matrix above is $n \times 2n$) equation (6.9) is immediately obtained.

Viewing the algorithm as maximizing the likelihood function, it is a special version of coordinated gradient method, see e.g. [78, 79], which is guaranteed to converge to $E[X|Z]$ for any initial point. Hence, all the eigenvalues of R must be less than one [80]. □

6.5 Numerical example

We consider the problem of estimating in a distributed way the derivative of an unknown function f from a finite set of noisy measurements of f. Each measurement is taken by distinct nodes. In particular, let $X_1(t)$ denote the derivative of the unknown function while $X_2(t)$ is its value. Our prior model for f is given by the stochastic differential equation (see [81] or [82])

$$\mathbf{d}X(t) = SX(t)\ \mathbf{d}t + T\ \mathbf{d}B(t) \tag{6.15}$$

where $B(t)$ is Brownian motion (its derivative is white noise) and

$$S = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} , \; T = \begin{pmatrix} 10^{-3} \\ 0 \end{pmatrix}$$

This model provides the basis for a Bayesian interpretation of cubic smoothing splines [83].
For $s \leq t$, we use $\mathbb{V}(t|s)$ to indicate the covariance of $X(t)$ given the value of $X(s)$ which, prior to knowing the system output measurements, satisfies the differential Lyapunov equation (see [84, page 133 equation 4.138])

$$\begin{aligned} \mathbb{V}(s|s) &= 0 \\ \partial_t \mathbb{V}(t|s) &= S\mathbb{V}(t|s) + \mathbb{V}(t|s)S^{\mathrm{T}} + TT^{\mathrm{T}}. \end{aligned}$$

For our particular choice of S and T, we have

$$\mathbb{V}(t|s) = 10^{-3} \begin{pmatrix} (t-s) & \frac{(t-s)^2}{2} \\ \frac{(t-s)^2}{2} & \frac{(t-s)^3}{3} \end{pmatrix}.$$

The distance between sampling points where each sensor is located is denoted by Δt. Thus, the transition model for $k > 1$ is represented by $G_k \in \mathbb{R}^{2\times 2}$ and $Q_k \in \mathbb{R}^{2\times 2}$ where

$$G_k = \begin{bmatrix} 1 & 0 \\ \Delta t & 1 \end{bmatrix}$$

$$Q_k = 10^{-3} \begin{pmatrix} \Delta t & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & \frac{\Delta t^3}{3} \end{pmatrix} .$$

The initial state is given by $x_0 \sim \mathcal{N}[x^0, Q_0]$, where

$$x^0 = X(t_1), \qquad Q_0 = \begin{pmatrix} 100 & 0 \\ 0 & 10 \end{pmatrix}$$

The measurement variance σ^2 is known and direct measurements of $X_2(t)$ are represented by $H_k \in \mathbb{R}^{2\times 2}$ and $R_k \in \mathbb{R}^{1\times 1}$ where

$$H_k = [0 \quad 1] \; , \; R_k = \sigma^2 \; .$$

The specifications for the example are completed by the following choices:

N	$=$	480	number of measurements
Δt	$=$	1	spacing between nodes
σ^2	$=$	1	measurement noise variance
v_k	$\sim$	$\mathcal{N}(0, R_k)$	simulated measurement noise
z_k	$=$	$h_k[X(t_k)] + v_k$	simulated measurement value

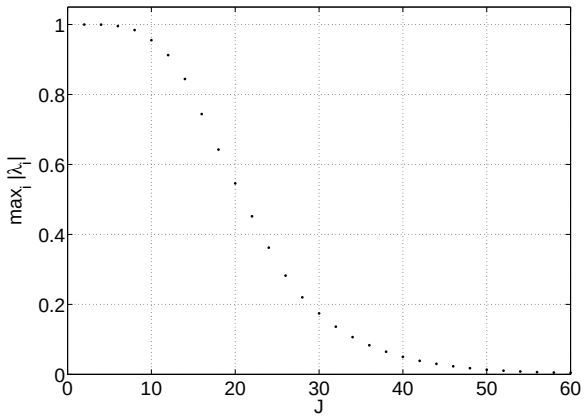


Figure 6.6: Cubic smoothing spline example: maximum absolute value of the eigenvalues $\{\lambda_i\}$ of matrix Γ regulating error dynamics as a function of J defining the blocks size.

Given such specifications, Fig. 6.6 reports the maximum absolute value of the eigenvalues $\{\lambda_i\}$ of the matrix R as a function of J which defines the size of the sensor blocks working in parallel. As a matter of fact, the developed analysis makes available a clear picture on the error dynamics also pointing out the importance of establishing a good trade off between level of parallelism and convergence rate of the iterative algorithm. For instance, when the value of J is low, say $J \leq 8$, a node is stressed with very high frequency to produce a new estimate of its own state and to send it to its adjacent node. However, even if this choice leads to a high level of parallelism, Fig. 6.6 suggests the need of a very large number of iterations for obtaining an acceptable level of accuracy in the estimate. In fact, the maximum modulus of the eigenvalue is very close to 1. The situation changes when J increases and a good trade-off appears between 16 and 30 where eigenvalues vary from 0.7 to 0.2.
To corroborate the theoretical analysis, let's consider a realization of f drawn from the prior and plotted in Fig. 6.7 (solid line). The aim is to reconstruct the derivative of such function from the noisy measurements displayed as circles in the same figure. Fig. 6.8 reports the estimates of the derivative of f obtained by the distributed Kalman smoother when J is 2 (top panels), 16 (middle) or 30 (bottom), for ℓ equal to 0,2 and 4. It is apparent that, in practice, when $J = 2$ the algorithm will never converge to the minimum variance estimate in reasonable time. On the other hand, when $J = 16$, already for $\ell = 2$ the algorithm returns an estimate sufficiently close to the optimum.

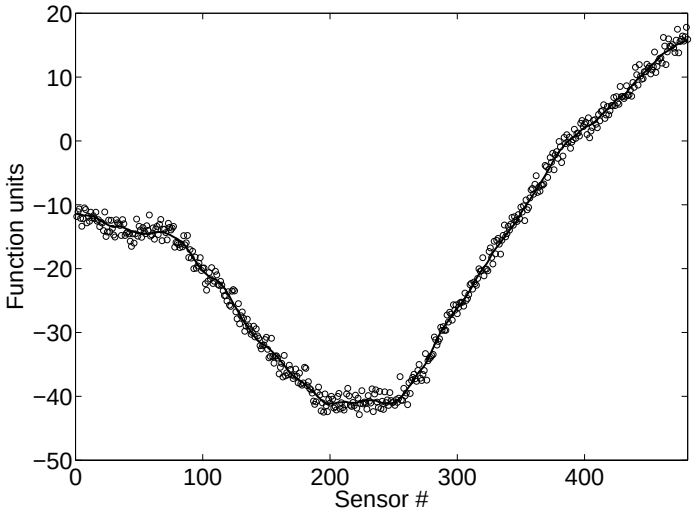


Figure 6.7: Cubic smoothing spline example: true function f (solid line) and noisy measurements (circles).

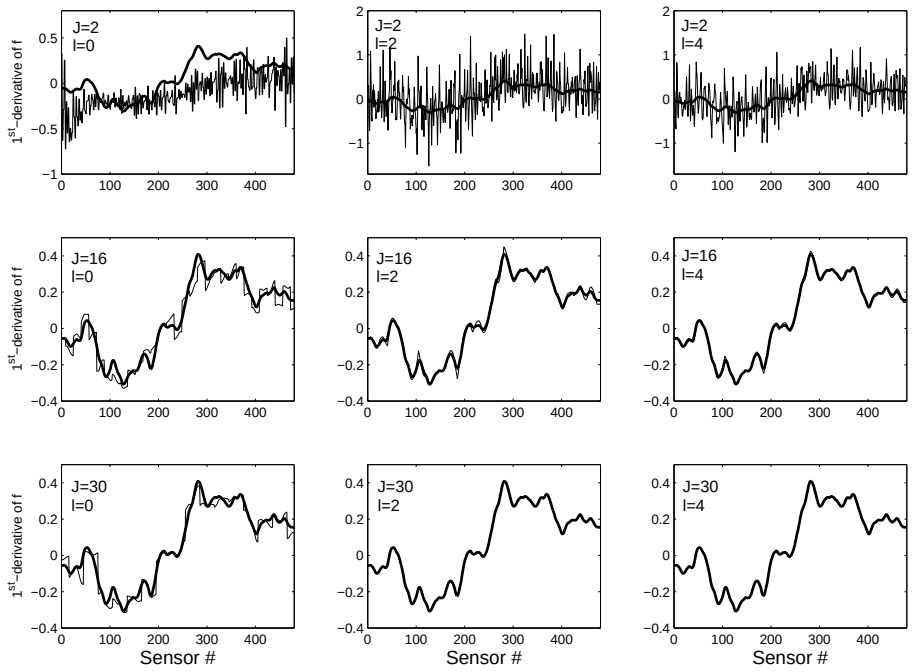


Figure 6.8: Reconstruction of the first derivative of f: minimum variance estimate (thick line), estimates from the distributed smoother (thin line) as a function of iteration number ℓ and size of nodes working in parallel (defined by J).

6.6 Analysis of $\lambda_{\max}(R)$ in a special case

The curve reported in figure 6.6 is particularly interesting since, being non-linear, it suggests that a small increment in the block size J leads to a drastic increment in the speed of convergence of the algorithm. This section is devoted therefore to the analysis of $\lambda_{\max}(R)$ as a function of J. We restrict to a special, yet significant, case of the model (6.1), (6.2), the scalar random walk. We derive an analytical expression for $\Pi_i(J)$, that tells how much an error in the estimate of $x_{K(j,j+1)}$ affects the estimate of $x_{L(j)}$. This expression is rather cumbersome and instead of handle it exactly, we study its asymptotic behavior in J showing that $\Pi_j(J)$ decreases as const $\cdot\, \nu^{-J/2}$ as J goes to infinity, where $\nu > 1$ depends on the process parameters.

We will then show that $\lambda_{\max}(R(J))$ asymptotically decreases as const$\cdot\nu^{-J}$ as J goes to infinity. For large vales of J the behavior of the curve of figure 6.6 is therefore theoretically characterized in this simple case.

Remark 4 (Notation). As we said, in this section we will often deal with asymptotic analysis of functions in J. Recall that we write that $f(J) \sim g(J)$ iff $\lim f(J)/g(J) = 1$ for $J \to +\infty$. Moreover we will denote with $\mathscr{C}$ **a generic** constant whose exact amount is not of interest. To avoid cumbersome notations, we allow $\mathscr{C}$ to represent different constants, even in the same formula.Some care has to be payed by the reader, since this choice might be a little confusing in some cases: for instance $[\mathscr{C} \quad \mathscr{C}]^T$ might represent any vector in $\mathbb{R}^2$, not only vectors of the form $\mathscr{C}[1 \quad 1]^T$. Whenever it is necessary to emphasize the fact that two constants might take different value, we will denote them as $\mathscr{C}_1$ and $\mathscr{C}_2$ rather then with $\mathscr{C}$.

Consider, as a special case of the model (6.1) and (6.2), the scalar random walk:

$$\begin{aligned} x_{k+1} &= x_k + w_k \\ x_0 &= x^0 \qquad\qquad k = 1, \dots N \\ z_k &= x_k + v_k \end{aligned} \tag{6.16}$$

where the model noise w_k is a white noise with variance λ, the initial condition x^0 is a zero mean Gaussian random variable having variance $\mathbb{V}(x^0)$ and we have access to noisy measurements of the state, z_k. We assume that the measurement noise, v_k, is a white noise with variance σ mutually independent of w_k.

Consider then the projectors Π_j and Ξ_j,

$$\Pi_j = \mathbb{V}\left(X_{L(j)}, X_{K(j,j+1)} \mid Z_{K(j,j+1)}\right) \mathbb{V}\left(X_{K(j,j+1)} \mid Z_{K(j,j+1)}\right).$$

Π_j tells us how much an error in the estimate of $x_{K(j)}$ and $x_{K(j+1)}$ affects the estimate of $x_{L(j)}$ given Z and an analogous role is played by Ξ_j. In appendix 6.A we compute them exactly but we report here, in the following proposition, an asymptotic results that describes the behavior of the projector Π_j and Ξ_j a J goes to infinity.

Proposition 26. *Let us define*

$$\nu = 1 + \frac{\lambda}{2\sigma} + \frac{1}{2}\sqrt{\frac{4\lambda}{\sigma} + \frac{\lambda^2}{\sigma^2}} \geq 1$$

Then, for all $j = 1, ..., p$,

$$\Pi_j \sim \mathscr{C}\nu^{-2J} \begin{bmatrix} \mathscr{C}\nu^{\frac{3}{2}J} & \mathscr{C}\nu^{\frac{3}{2}J} \end{bmatrix} \sim \nu^{-\frac{J}{2}} \begin{bmatrix} \mathscr{C}_1 & \mathscr{C}_2 \end{bmatrix}$$

Analogously $j = 2, ..., p \quad \Xi_j \sim \nu^{-\frac{J}{2}} \begin{bmatrix} \mathscr{C}_1 & \mathscr{C}_2 \end{bmatrix}$
and moreover $\Xi_1 \sim \Xi_{p+1} \sim \mathscr{C}\nu^{-\frac{J}{2}}$.

Proof.
See appendix 6.A. □

From the above mentioned result, we have therefore that

$$\begin{aligned} E_j = \Xi_j \begin{bmatrix} \Pi_{j-1} & 0 \\ 0 & \Pi_j \end{bmatrix} &\sim \nu^{-\frac{J}{2}} \begin{bmatrix} \mathscr{C} & \mathscr{C} \end{bmatrix} \nu^{-\frac{J}{2}} \begin{bmatrix} \mathscr{C} & \mathscr{C} & 0 \\ 0 & \mathscr{C} & \mathscr{C} \end{bmatrix} \\ &\sim \nu^{-J} \begin{bmatrix} \mathscr{C} & \mathscr{C} & \mathscr{C} \end{bmatrix} \quad \forall j = 2, \dots, p \end{aligned}$$

and analogously

$$E_1 = \Xi_1\Pi_1 \sim \nu^{-J} \begin{bmatrix} \mathscr{C} & \mathscr{C} \end{bmatrix} \qquad E_{p+1} = \Xi_{p+1}\Pi_p \sim \nu^{-J} \begin{bmatrix} \mathscr{C} & \mathscr{C} \end{bmatrix}$$

The matrix R defined in (6.8) is then a tridiagonal matrix having all entries $r_{i,j} \sim \mathscr{C}\nu^{-J}$, which leads to the following result:

Proposition 27. *All the eigenvalues of* R *have the same asymptotic behavior,*

$$\lambda(R(J)) \sim \mathscr{C}\nu^{-J},$$

in particular $\lambda_{max}(R(J)) \sim \mathscr{C}\nu^{-J}$.

Proof.
The proof follow directly form the following, more general, linear algebra fact:

Lemma 28. *Consider a matrix A whose entries are function of J $a_{i,j}(J)$. If all the entries are asymptotically equivalent to a function $f(J)$ as J goes to infinity, $a_{i,j}(J) \sim \mathscr{C}_{i,j} f(J)$ then also $\lambda(A(J)) \sim \mathscr{C}\nu^{-J}$*

Proof of lemma 28.

$$a_{i,j}(J) \sim f(J) \qquad \Leftrightarrow \qquad \lim_{J\to+\infty} \frac{a_{i,j}(J)}{f(J)} = \mathscr{C}_{i,j}$$

therefore

$$\lim_{J\to+\infty} \frac{1}{f(J)} A = [\mathscr{C}_{i,j}]$$

Recall moreover that the roots of a polynomial are continuous function of its coefficients. This implies that the eigenvalues of a matrix, being the roots of the characteristic polynomial, are a continuous function of the matrix entries. We have hence that

$$\lim_{J\to+\infty} \frac{1}{f(J)} \lambda(A) = \lambda([\mathscr{C}_{i,j}])$$

or equivalently the thesis. □

□

In figure 6.9 we depict $\lambda_{max}(R)$ (solid thick line) and its asymptotic approximation $\mathscr{C}\nu^{-J}$(dashed thin line) versus J, for various values of σ. In the right panel a semilog scale is used, to highlight the good the matching between the true function and its asymptotic approximation.

6.7 Final Comments and Open Issues

We have considered smoothing of Gauss-Markov linear systems via distributed optimization. In the context of a sensor network, our problem amounts to assuming that any node has access to noisy measurements of different but correlated states. Then, the aim is to reconstruct the overall state sequence in a cooperative way, by taking advantage of all the data obtained by the network. A parallel smoothing scheme has been presented together with a convergence analysis. The latter points out the importance, in the algorithm design, of finding the right trade off between parallelism and rate of convergence toward the optimal estimate.

The algorithm presented in this chapter represents the very first step toward more complex algorithms for the estimation of different but correlated time-varying quantities, such as variables that depends both on time and

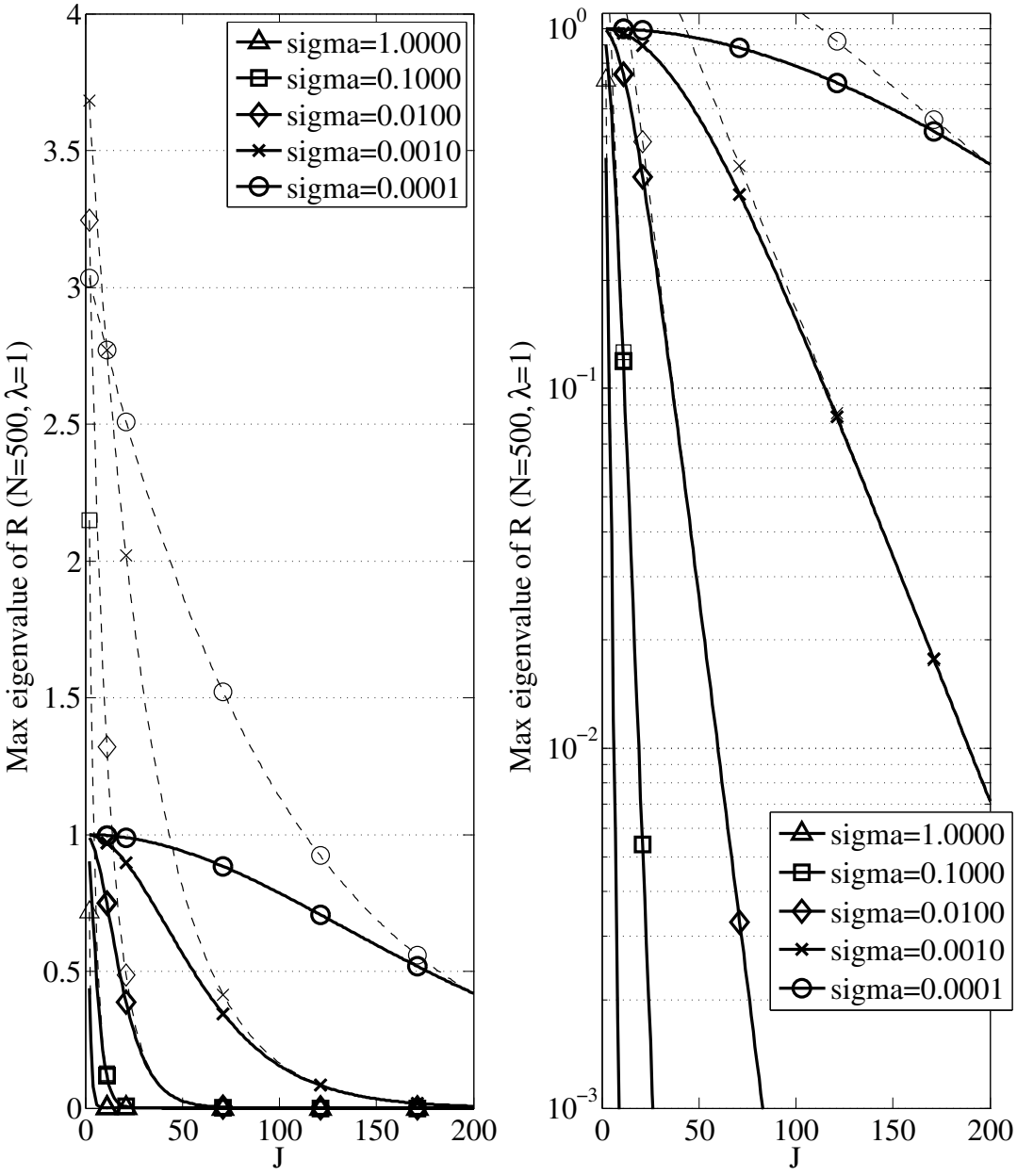


Figure 6.9: $\lambda_{max}(R)$ (solid thick line) and its asymptotic approximation $\mathscr{C}\nu^J$(dashed thin line) versus J, for various values of the parameter σ. In the lower panel the same figure is depicted in a semilog scale. Obtained for $\lambda = \sigma_0 = 1$.

on space. Effective distributed estimation techniques for quantities varying both in time and in space could have relevant impact in many applications, ranging form environmental or scientific monitoring, temperature profiling, estimation of polluting agent diffusion, voltage drop along electrical lines or evaluation sea wave strength. More generally, whenever a sensor network is required to monitor a physical quantity described by partial differential equations these techniques come into play.

Further steps along this research should start from the extension of the proposed method, and its convergence analysis, to more general and complex graphical models. Afterwards the case of a time-varying signal needs to be addressed.

6.A Appendix

This appendix is devoted to the computation of the projectors Π_j and Ξ_j

$$\Pi_j = \mathbb{V}\left(x_{L(j)}, X_{K(j,j+1)} \mid Z_{K(j,j+1)}\right) \mathbb{V}\left(X_{K(j,j+1)} \mid Z_{K(j,j+1)}\right)$$

in the case of noisy measurements of a scalar random walk, (6.16).

6.A.a Exploiting random walk structure

To compute Π_j let us consider the j-th block of nodes.
Let us introduce a new node index, h, such that $k = K(j) + h$. Hence h describes the position of a node of the j–th block with respect to the initial node $K(j)$. The model for the states of the j–th block becomes then

$$\begin{aligned} x^j_{h+1} &= x^j_h + w^j_h \\ x^j_1 &= x_{K(j)} \qquad\qquad h = 1, \ldots, J+1. \\ z_h &= x^j_h + v^j_h \end{aligned} \tag{6.17}$$

In this way the dependence on j is all contained in the initial condition x^j_1. For this reason we will use the indexing h from now on. Let us collect all the $J+1$ states associated with the j–th block nodes in the vector

$$X_j = [x^j_1, \ldots, x^j_{J+1}]$$

and the associated measurements in the vector Z_j. Recall that $\mathbb{V}(w_h) = \lambda$, $\mathbb{V}(v_h) = \sigma$ and denote with $\lambda_0 = \mathbb{V}(x_{K(j)}) = \mathbb{V}(x^0) + K(j)\lambda$.

It is easy to see that:

$$\mathbb{V}(X_j) = \lambda_0 \mathbb{1}\mathbb{1}^T + \lambda \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & 2 & \ldots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \ldots & N \end{bmatrix}$$

and

$$\mathbb{V}(Z_j) = \mathbb{V}(X_j) + \sigma I.$$

Note moreover that $\mathbb{V}(X_j Z_j) = \mathbb{V}(X_j)$.

Alternatively one can note that (6.17) can be rewritten as

$$X_j = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ & 1 & 0 & & \\ & & \ddots & \ddots & \\ & & & 1 & 0 \end{bmatrix} X_j + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \\ 0 \end{bmatrix} x_1 + W_j,$$

where W_j is the vector having entries w_h^j. Therefore,

$$BX_j = e_1 x_0 + W_j,$$

where $e_1 = [1,\ 0, \dots, 0]^T$ and

$$B = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}.$$

It can easily be verified that

$$B^{-1} = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & \ddots & \ddots & \\ 1 & \dots & 1 & 1 \end{bmatrix}$$

and hence, noting that $B^{-1}e_1 = \mathbb{1}$, we get

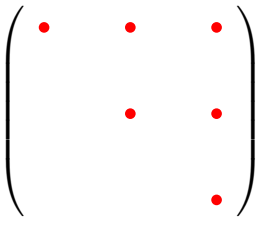
$$\begin{aligned} \mathbb{V}(X_j) &= \lambda_0 B^{-1} e_1 e_1^T B^{-T} + \lambda B^{-1} B^T \\ &= \lambda_0 \mathbb{1}\mathbb{1} + \lambda B^{-1} B^T \end{aligned} \tag{6.18}$$

The evaluation of the projector Π_j requires the knowledge of some elements of the a–posteriori estimation error variance

$$\mathbb{V}(X_j|Z_j) = \mathbb{V}(X_j) - \mathbb{V}(X_j Z_j)\mathbb{V}(Z_j)^{-1}\mathbb{V}(X_j Z_j)^T,$$

specifically of those in positions:

$$(1,1),\ (J+1, J+1),\ (J+1, 1),\ (1, J/2),\ (J/2, J+1),\ (J/2, J/2)$$



Note that

$$\begin{aligned} \mathbb{V}(X_j|Z_j) &= \Big(\mathbb{V}(X_j) - \mathbb{V}(X_j Z_j)\mathbb{V}(Z_j)^{-1}\mathbb{V}(X_j Z_j)\Big) \\ &= \Big(\mathbb{V}(X_j) - \mathbb{V}(X_j)\big(\mathbb{V}(X_j) + \sigma I\big)^{-1}\mathbb{V}(X_j)\Big) \\ &= \Big(\mathbb{V}(X_j)^{-1} + \frac{1}{\sigma} I\Big)^{-1}, \end{aligned}$$

where in the last equality we used the matrix inversion lemma.
Using (6.18) we obtain

$$\mathbb{V}(X_j)^{-1} = (B^{-1}(\lambda_0 e_1 e_1^T + \lambda I)B^{-T})^{-1} = B^T(\lambda_0 e_1 e_1^T + \lambda I)^{-1}B.$$

Since $(\lambda_0 e_1 e_1^T + \lambda I)$ is a diagonal matrix its inverse can be easily computed to be

$$\frac{1}{\lambda}I - \left(\frac{\lambda_0}{\lambda(\lambda_0+\lambda)}\right) e_1 e_1^T.$$

Therefore

$$\begin{aligned}\mathbb{V}(X_j)^{-1} &= -\left(\frac{\lambda_0}{\lambda(\lambda_0+\lambda)}\right) e_1 e_1^T + \frac{1}{\lambda}B^T B \\ &= \frac{1}{\lambda}\begin{bmatrix} 2-\frac{\lambda_0}{\lambda_0+\lambda} & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 1 \end{bmatrix},\end{aligned}$$

where we used the fact that $B^T e_1 = e_1$. The a–posteriori error variance becomes hence:

$$\begin{aligned}\Sigma_{\tilde{X}} &= (\Sigma_X^{-1} + \frac{1}{\sigma}I)^{-1} \\ &= \lambda\begin{bmatrix} 2-\frac{\lambda_0}{\lambda_0+\lambda}+\frac{\lambda}{\sigma} & -1 & & & & & \\ -1 & 2+\frac{\lambda}{\sigma} & -1 & & & & \\ & -1 & 2+\frac{\lambda}{\sigma} & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2+\frac{\lambda}{\sigma} & -1 & \\ & & & & -1 & 1+\frac{\lambda}{\sigma} \end{bmatrix}^{-1}\end{aligned}$$

Thus we are interested to compute some elements of the inverse of a tridiagonal matrix of the form:

$$\begin{bmatrix} a & -1 & & & & \\ -1 & b & -1 & & & \\ & -1 & b & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & b & -1 \\ & & & & -1 & c \end{bmatrix}^{-1} \tag{6.19}$$

where $b \geq 2$.

The next section of this appendix is devoted to this aim.

6.A.b Inversion of a tridiagonal matrix in the form (6.19)

We use the formula:

$$(A^{-1})_{i,j} = (-1)^{j+i}\frac{\mathcal{A}_{j,i}}{\det(A)}$$

i.e.

$$A^{-1} = \frac{1}{\det(A)}\begin{bmatrix} \mathcal{A}_{1,1} & \dots & (-1)^{1+J+1}\mathcal{A}_{1,J+1} \\ \vdots & & \vdots \\ (-1)^{J+1+1}\mathcal{A}_{J+1,1} & \dots & (-1)^{J+1+J+1}\mathcal{A}_{J+1,J+1} \end{bmatrix}^T$$

where $\mathcal{A}_{ji}$, known as adjugate, is the determinant of the square $J \times J$ matrix obtained from A removing the row i and the column j.

As we will see, the computation of the elements of interest of A^{-1} reduces to the computation of the determinant of tree special tridiagonal matrices defined below

$$\overline{M_n} = \begin{bmatrix} a & -1 & & & & \\ -1 & b & -1 & & & \\ & -1 & b & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & b & -1 \\ & & & & -1 & c \end{bmatrix} \tag{6.20}$$

$$M_n = \begin{bmatrix} b & -1 & & & & \\ -1 & b & -1 & & & \\ & -1 & b & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & b & -1 \\ & & & & -1 & c \end{bmatrix} \tag{6.21}$$

$$\widetilde{M_n} = \begin{bmatrix} a & -1 & & & & \\ -1 & b & -1 & & & \\ & -1 & b & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & b & -1 \\ & & & & -1 & b \end{bmatrix} \tag{6.22}$$

where the subscript n specifies the dimension of the matrix: $M_n, \overline{M_n}, \widetilde{M_n} \in \mathbb{R}^{n\times n}$

Let us start, therefore, by determining the determinant of these matrices. Note that:

$$\det M_n = b \det \begin{pmatrix} b & -1 & & & 0 \\ -1 & b & -1 & & \\ & -1 & b & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & b & -1 \\ 0 & & & & -1 & c \end{pmatrix} +$$

$$+ (-1)^{2+1}(-1) \det \begin{pmatrix} b & -1 & & & 0 \\ -1 & b & -1 & & \\ & -1 & b & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & b & -1 \\ 0 & & & & -1 & c \end{pmatrix}$$

$$= b \det(M_{n-1}) + \det \begin{pmatrix} b & -1 & & & 0 \\ -1 & b & -1 & & 0 \\ & -1 & b & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & b & -1 \\ 0 & & & & -1 & c \end{pmatrix}$$

$$= b \det(M_{n-1}) - \det(M_{n-2}).$$

We have therefore a very simple second–order recursive relation that describes $\det M_n$ as n varies. The initial conditions are

$$\det M_1 = c \tag{6.23}$$

$$\det M_2 = bc - 1 \tag{6.24}$$

As it can be easily verified, the recursive relation can be solved to obtain an explicit form for $\det(M_n)$ as a function of n. Define

$$\nu = \frac{1}{2}b + \frac{1}{2}\sqrt{b^2 - 4}$$

and $\mu = \frac{1}{\nu}$, we get:

$$\det(M_n) = c_1\nu^n + c_2\mu^n \tag{6.25}$$

where c_1 and c_2 are two constants that can be easily determined by forcing (6.25) to match the initial conditions (6.23). Note moreover that

$$\nu \geq 1$$

while $\mu \leq 1$.

Since we are interested in the asymptotic behavior of $\det(M_J)$ as J goes to infinity, we note that

$$\det(M_n) \sim c_1\nu^n.$$

Moreover, to our proposes, we are not interested to the exact value of the the const c_1, but rather to the fact that it is **a** constant. We will therefore write

$$\det(M_n) \sim \mathscr{C}\nu^n,$$

according to the notation introduced in the remark 4 of section 6.6 .

Consider then $\det \overline{M_n}$:

$$\det \overline{M}_n = a \det \begin{pmatrix} a & 1 & & & 0 \\ -1 & b & -1 & & \\ & -1 & b & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & b & -1 \\ 0 & & & & -1 & c \end{pmatrix} +$$

$$+ (-1)^{2+1}(-1) \det \begin{pmatrix} a & 1 & & & 0 \\ -1 & b & -1 & & \\ & -1 & b & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & b & -1 \\ 0 & & & & -1 & c \end{pmatrix}$$

$$= a \det(M_{n-1}) - \det(M_{n-2})$$

therefore

$$\det \overline{M}_n \sim \mathscr{C}\nu^n,$$

in fact,

$$\det \overline{M}_n \sim ac_1\nu^{n-1} - c_1\nu^{n-2} = (a\nu - 1)\frac{c_1}{\nu^2}k^n.$$

Note that in this case the symbol $\mathscr{C}$ stands for the constant $(a\nu - 1)\frac{c_1}{\nu^2}$.

Analogously one notices that $\det \widetilde{M}_n$ obeys to the following recursion

$$\det \widetilde{M}_n = b \det \widetilde{M}_{n-1} - \det \widetilde{M}_{n-2} \tag{6.26}$$

$$\det \widetilde{M}_1 = a \tag{6.27}$$

$$\det \widetilde{M}_2 = ab - 1 \tag{6.28}$$

therefore

$$\det \widetilde{M}_n = c_3\nu^n + c_4\mu^n$$

with $c_1 \neq c_3$, $c_2 \neq c_4$, since the initial conditions are different. Also in this case it holds that:

$$\det(\widetilde{M}_n) \sim \mathscr{C}\nu^n.$$

Finally we evaluate the elements of interest in A^{-1}. To this aim, first of all, note that

$$\det A = \det \overline{M_{J+1}} \sim \mathscr{C}\nu^J$$

and

$$\mathcal{A}_{1,1} = \det \begin{pmatrix} a & 1 & & & 0 \\ -1 & b & -1 & & \\ & -1 & b & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & b & -1 \\ 0 & & & & -1 & c \end{pmatrix}$$
$$= \det M_J \sim \mathscr{C}\nu^J.$$

Moreover

$$\mathcal{A}_{J+1,J+1} = \det \begin{pmatrix} a & -1 & & & 0 \\ -1 & b & -1 & & \\ & -1 & b & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & b & -1 \\ 0 & & & & 1 & c \end{pmatrix}$$
$$= \det \widetilde{M}_J \sim \mathscr{C}\nu^J$$

while

$$\mathcal{A}_{\frac{J}{2}+1,\frac{J}{2}+1} = \det \begin{pmatrix} a & -1 & & & 0 & & & \\ -1 & b & -1 & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & -1 & b & -1 & & & \\ 0 & & & -1 & b & -1 & & 0 \\ & & & & -1 & b & -1 & \\ & & & & & \ddots & \ddots & \ddots \\ & & & & & & -1 & b & -1 \\ 0 & & & & 0 & & & -1 & c \end{pmatrix}$$

$$= \det \widetilde{M}_{\frac{J}{2}} \det M_{\frac{J}{2}}$$

$$\sim \mathscr{C}\nu^{J/2}\nu^{J/2} \sim \mathscr{C}\nu^{J}.$$

We get than that

$$\mathcal{A}_{1,J+1} = \det \begin{pmatrix} a & -1 & & & & & 0 \\ -1 & b & -1 & & & & \\ & -1 & b & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & b & -1 & \\ 0 & & & & -1 & c & \end{pmatrix}$$

$$= \det \left[\text{ Upper–triangular Matrix } \right]$$

$$= (-1)^J = 1$$

Is is just a little more complex to determine $\mathcal{A}_{1,\frac{J}{2}+1}$:

$$\mathcal{A}_{1,\frac{J}{2}+1} = \det \begin{pmatrix} a & 1 & & & & & & & 0 \\ -1 & b & -1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & -1 & b & -1 & & & & \\ & & & -1 & b & -1 & & & \\ & & & & -1 & b & -1 & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & -1 & b & -1 \\ & & & & 0 & & & -1 & c \end{pmatrix}$$

$$= (-1)(-1)^{\frac{J}{2}+\frac{J}{2}} \det \begin{pmatrix} a & 1 & & & & & & & 0 \\ -1 & b & -1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & -1 & b & -1 & & & & \\ & & & -1 & b & -1 & & & \\ & & & & -1 & b & -1 & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & -1 & b & -1 \\ & & & & 0 & & & -1 & c \end{pmatrix} +$$

$$+ (-1)(-1)^{\frac{J}{2}+\frac{J}{2}+1} \det \begin{pmatrix} a & 1 & & & & & & & 0 \\ -1 & b & -1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & -1 & b & -1 & & & & \\ & & & -1 & b & -1 & & & \\ & & & & -1 & b & -1 & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & -1 & b & -1 \\ & & & & 0 & & & -1 & c \end{pmatrix}$$

$$\mathcal{A}_{1,\frac{J}{2}+1} = (-1)^{J+1}\det\left[\begin{array}{ccccc|ccccc} -1 & b & -1 & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & \\ & & -1 & b & -1 & & & & & \\ & & & -1 & b & & & & & \\ & & & & -1 & & & & & \\ \hline & & & & & b & -1 & & & \\ & & & & & -1 & b & -1 & & \\ & & & & & & \ddots & \ddots & \ddots & \\ & & & & & & & -1 & b & -1 \\ & & & & & & & & -1 & c \end{array}\right] +$$

$$(-1)^{J+2}\det\left[\begin{array}{ccccc|ccccc} -1 & b & -1 & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & \\ & & -1 & b & -1 & & & & & \\ & & & -1 & b & -1 & & & & \\ & & & & -1 & b & & & & \\ \hline & & & & & 0 & -1 & & & \\ & & & & & 0 & b & -1 & & \\ & & & & & 0 & -1 & b & -1 & \\ & & & & & \vdots & & \ddots & \ddots & \ddots \\ & & & & & & & & -1 & b & -1 \\ & & & & & 0 & & & & -1 & c \end{array}\right]$$

$$= -\det\left[\begin{array}{c|c} \text{Upper Triangular} & \\ \hline & M_{\frac{J}{2}} \end{array}\right] + \det\left[\begin{array}{c|c} \text{Upper Triangular} & * \\ \hline & \text{Singular Matrix} \end{array}\right]$$

$$= (-1)(-1)^{\frac{J}{2}-1}\det(M_{\frac{J}{2}}) + (-1)^{\frac{J}{2}-1}0$$

$$= (-1)^{\frac{J}{2}}\det(M_{\frac{J}{2}}) \sim \mathscr{C}(-1)^{\frac{J}{2}}\nu^{J/2}$$

Analogously one gets

$$\mathcal{A}_{\frac{J}{2},J+1} = \det \begin{pmatrix} a & -1 & & & & & & & 0 \\ -1 & b & -1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & -1 & b & -1 & & & & \\ 0 & & & -1 & b & -1 & & & 0 \\ & & & & -1 & b & -1 & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & -1 & b & -1 \\ & & & & & & & -1 & c \end{pmatrix}$$

$$= (-1)(-1)^{\frac{J}{2}+\frac{J}{2}-1} \det \begin{pmatrix} a & -1 & & & & & & & 0 \\ -1 & b & -1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & -1 & b & -1 & & & & \\ 0 & & & -1 & b & -1 & & & 0 \\ & & & & -1 & b & -1 & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & -1 & b & -1 \\ & & & & & & & -1 & c \end{pmatrix} +$$

$$+ (-1)(-1)^{\frac{J}{2}+\frac{J}{2}-1} \det \begin{pmatrix} a & -1 & & & & & & & 0 \\ -1 & b & -1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & -1 & b & -1 & & & & \\ 0 & & & -1 & b & -1 & & & 0 \\ & & & & -1 & b & -1 & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & -1 & b & -1 \\ & & & & & & & -1 & c \end{pmatrix}$$

$$\mathcal{A}_{\frac{J}{2},J+1} = (-1)^J \det \left[\begin{array}{ccccc|ccccc} a & -1 & & & & & & & & \\ -1 & b & -1 & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & \\ & & -1 & b & -1 & & & & & \\ \hline & & & & & 0 & b & -1 & & \\ & & & & & 0 & -1 & b & -1 & \\ & & & & & \vdots & & \ddots & \ddots & \ddots \\ & & & & & 0 & & & -1 & b \\ & & & & & 0 & & & & -1 \end{array}\right] +$$

$$+ (-1)^{J+2} \det \left[\begin{array}{cccc|cccc} a & -1 & & & & & & \\ -1 & b & -1 & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & -1 & b & & & & \\ \hline & & & & -1 & b & -1 & \\ & & & & & \ddots & \ddots & \ddots \\ & & & & & & -1 & b \\ & & & & & & & -1 \end{array}\right]$$

$$= \det \left[\begin{array}{c|c} \text{Upper Triangular} & * \\ \hline & \text{Singular Matrix} \end{array}\right] - \det \left[\begin{array}{c|c} \overline{M}_{\frac{J}{2}} & \\ \hline & \text{Upper Triangular} \end{array}\right]$$

$$= (-1)^{J/2-1} 0 + (-1)^{J/2} \det(\widetilde{M}_{\frac{J}{2}})$$

$$\sim \mathscr{C}(-1)^{J/2} \nu^{J/2-1}$$

Summarizing we get

$$
\begin{aligned}
(A^{-1})_{1,1} &= \frac{\det M_J}{\det(\overline{M_{J+1}})} \sim \mathscr{C} \\
(A^{-1})_{J+1,J+1} &= \frac{\det \widetilde{M}_J}{\det(\overline{M_{J+1}})} \sim \mathscr{C} \\
(A^{-1})_{J/2+1,J/2+1} &= \frac{(-1)^{\frac{J}{2}+\frac{J}{2}}\mathcal{A}_{J/2,J,2}}{\det(A)} = \frac{\det \widetilde{M}_{J/2} \det M_{J/2}}{\det(\overline{M_{J+1}})} \sim \mathscr{C} \\
(A^{-1})_{1,J+1} = (A^{-1})_{J+1,1} &= \frac{(-1)^{1+J+1}\mathcal{A}_{1,J+1}}{\det(A)} = \frac{1}{\det(\overline{M_{J+1}})} \sim \mathscr{C}\nu^{-J} \\
(A^{-1})_{J/2+1,1} = (A^{-1})_{1,J/21} &= \frac{(-1)^{\frac{J}{2}+1}\mathcal{A}_{1,\frac{J}{2}+1}}{\det(A)} = \frac{\det M_{J/2}}{\det(\overline{M_{J+1}})} \sim \mathscr{C}\nu^{-J/2} \\
(A^{-1})_{J/2+1,J+1} = (A^{-1})_{J+1,J/2+1} &= \frac{(-1)^{\frac{J}{2}+J+1}\mathcal{A}_{1,\frac{J}{2}+1}}{\det(A)} = \frac{\det \widetilde{M}_{J/2}}{\det(\overline{M_{J+1}})} \sim \mathscr{C}\nu^{-J/2}
\end{aligned}
$$

6.A.c Computing Π_j

Using the results obtained so far we get

$$
\begin{aligned}
\Pi_j &= \mathbb{V}([x_1\ x_{J+1}], x_{J/2+1}|Z)\mathbb{V}^{-1}([x_1\ x_{J+1}]|Z) \\
&= \begin{bmatrix} \dfrac{\det M_{J/2}}{\det \overline{M_{J+1}}} & \dfrac{\det \widetilde{M}_{J/2}}{\det \overline{M_{J+1}}} \end{bmatrix} \cdot \begin{bmatrix} \frac{\det M_J}{\det \overline{M_{J+1}}} & \frac{1}{\det \overline{M_{J+1}}} \\ \frac{1}{\det \overline{M_{J+1}}} & \frac{\det \widetilde{M}_J}{\det \overline{M_{J+1}}} \end{bmatrix}^{-1} \\
&= \frac{1}{\det \overline{M_{J+1}}} \begin{bmatrix} \det M_{J/2} & \det \widetilde{M}_{J/2} \end{bmatrix} \cdot \left(\frac{1}{\det \overline{M_{J+1}}} \begin{bmatrix} \det M_J & 1 \\ 1 & \det \widetilde{M}_J \end{bmatrix} \right)^{-1} \\
&= \begin{bmatrix} \det M_{J/2} & \det \widetilde{M}_{J/2} \end{bmatrix} \cdot \frac{1}{\det M_J \det \widetilde{M}_J - 1} \begin{bmatrix} \det \widetilde{M}_J & -1 \\ -1 & \det M_J \end{bmatrix} \\
&= \frac{1}{\det M_J \det \widetilde{M}_J - 1} \begin{bmatrix} \det M_{J/2} \det \widetilde{M}_J - \det \widetilde{M}_{J/2} \\ \det \widetilde{M}_{J/2} \det M_J - \det M_{J/2} \end{bmatrix}^T
\end{aligned}
$$

which has the asymptotic behavior reported in Proposition 26:

$$
\Pi_j \sim \mathscr{C}\nu^{-2N} \begin{bmatrix} \mathscr{C}\nu^{\frac{3}{2}J} \\ \mathscr{C}\nu^{\frac{3}{2}J} \end{bmatrix}^T \sim \nu^{-\frac{J}{2}} \begin{bmatrix} \mathscr{C}_1 \\ \mathscr{C}_2 \end{bmatrix}^T .
$$

Since the asymptotic behavior of Π_j is independent on the variance of the initial condition, it is also independent on j and therefore on the index of the

first node of the group $K(j)$. We have also that

$$\Xi_j \sim \Pi_j \quad \forall j = 2, \ldots, p.$$

It is easy to see moreover that

$$\Xi_1 = \mathbb{V}(x_{J/2+1}, x_1|Z)\mathbb{V}^{-1}(x_{J/2+1}|Z_{L(1,2)}) \sim \mathscr{C}\nu^{-\frac{J}{2}}$$

and that, analogously, $\Xi_{p+1} \sim \mathscr{C}\nu^{-\frac{J}{2}}$.

7 Conclusions

Recently, we have assisted to a great technological improvement that has made available, at very contained prices, microsensors with embedded communication and processing functions. Networks of a number of these microsensors, interacting with each other and cooperating to reach a given common objective, can be used in a great variety of applications and promise to revolutionize our daily life. Some of the most promising applications for sensor networks are monitoring of vast areas, cooperative estimation and detection.

Nevertheless, these sensors have usually simple hardware, low computational power, little communication capabilities and they often have to work making a careful usage of energy, which in turn is a very critical resource in battery powered devices. For this reason, the development of efficient distributed estimation and data fusion algorithms becomes crucial to avoid unmanageable computational and communicational burden on network bottleneck nodes.

In this thesis we addressed some issues in this emerging field, both presenting new algorithms and carrying out the analysis of solutions recently appeared in literature. Most of the algorithms considered leverage on consensus steps to distribute the computation among nodes. Consensus algorithms are therefore used as a key module to build new estimation algorithms and the rich consensus theory becomes a precious and profitable toolbox for our analysis.

For this reason we opened the thesis by reviewing, in **Chapter 1**, some material on *graph theory* and *consensus theory*. In particular, we summarize some convergence results both for the time-invariant and time-varying case, with special attention to the important class of randomized algorithms.

Probabilistic convergence results are presented and two important examples, namely symmetric gossip and broadcast, are discussed and compared.

In Chapter 2 we considered the problem of computing global quantities, i.e. quantities that are function of all the data collected in the network. We reported a result that characterizes a class of global functions that can be computed by means of consensus algorithms and we showed that many problems of interest fit this class. In particular we describe how to cast into a consensus problem the problem of computing generalized means, least square estimate, maximum likelihood estimate and also Kalman estimate.

Nevertheless, the equivalence between the centralized algorithm and the decentralized consensus-based algorithm relies on an asymptotic result. It is not clear what is the behavior of these algorithms for finite number of iterations t.

The fact that consensus has to be reached for the algorithm to work, has its most relevant impact in the Kalman filer case. Indeed, in this example the algorithm prescribes to perform a whole consensus algorithm run within two subsequent measurements. Hence, the time available to the consensus algorithm depends on the application and if it is not sufficient to reach consensus, then the strategy is not guaranteed to achieve the centralized Kalman filter estimate.

For this reason, when the time between two subsequent measurements is small, other strategies might be preferable. One of the possible alternative strategies has been presented and analyzed in Chapter 5: it minimizes the steady state estimation variance assuming that only a given amount m of consensus iterations is exploitable.

In Chapter 3 we focused on one of the most important applications of wireless sensor networks: localization and tracking. In particular, we studied two problems arising when localization is based on the strength of the radio signal received. Specifically, we first proposed a distributed strategy to minimize the effects of unknown constant offsets in the reading of the Radio Strength Signal Indicator (RSSI), due to uncalibrated sensors.

The field of application of the proposed solution is definitely wider than the few applications presented in this work. For example, the offset removal algorithm could also be used to detect malfunctioning sensors by observing the magnitude of the compensation offset $\hat{o}_i$.

Then, we considered the problem of estimating the channel parameters for a generic wireless sensor network in a distributed manner. To do so, we formulated the estimation problem as global least-square problem, that we solved using the distributed algorithm presented in Chapter 2. The pro-

posed algorithms do not require any knowledge on the global topology of the network nor on the total number of nodes.

Finally, we applied these algorithms to experimental data collected from an indoor wireless sensor network.

In **Chapter 4** we presented a new majorization inequality on the singular values of the expectation of matrix–valued random variables' product. This inequality, devised during the analysis the algorithm presented in Chapter 5, is a general linear algebra result that can be useful in the analysis of the convergence rate of general jump–Markov linear systems.

It is shown that, if the alphabet is made of normal matrices, then the singular values of the matrix $\mathbb{E}[\mathbf{Q}_i \dots \mathbf{Q}_0 \mathbf{Q}_0^T \dots \mathbf{Q}_i^T]$ are submajorized by the singular values of $\mathbb{E}[\mathbf{Q}_0 \mathbf{Q}_0^T]^i$.

As a straightforward corollary of this result, we proposed a novel trace inequality.

In **Chapter 5** we considered the problem of estimating a random process from noisy measurements, collected by a sensor network.

We analyzed a randomized version of the distributed Kalman filter proposed in [40]. In fact, in [40] it has been considered only the case of fixed communication strategies, with constant consensus matrix. However, in many practical cases this is just a rough model of communication in a sensor network, that usually happens according to a randomized strategy. This strategy is more properly modeled by assuming that the consensus matrices are drawn, according to a selection probability, from an alphabet of matrices compatible with the communication graph, at each time instant.

We carried out a worst-case analysis, that leaded to an over–pessimistic characterization of the algorithm, and a more significant mean–square analysis, proving that the error variance remains bounded and providing an upper–bound for this quantity.

This upper–bound is a good performance assessment index and therefore it is assumed as a cost function to be minimized. Moreover we show that the problem of minimizing this cost function by choosing the Kalman gain and the selection probability is convex in each of the two variables separately although it is not jointly convex. On the contrary, even the problem of minimizing the trace of the estimation-error variance with respect to selection probability is not convex.

Finally simulations were presented and the results discussed.

In **Chapter 6** we considered the set up of a network of nodes cooperating to estimate different but correlated quantities. We assumed these quantities

to be constant in time. We presented a cooperative smoothing algorithm for Gauss-Markov linear models whose aim is to reconstruct the overall state sequence in a cooperative way, by taking advantage of all the data obtained by the network.

A convergence analysis was carried out, fully characterizing the gap between distributed and optimal estimate. This points out the importance, in the algorithm design, of finding the right trade-off between parallelism and rate of convergence toward the optimal estimate. In the simple yet significant case of a random walk, this issue has been further investigated. The convergence rate has been studied as a function of J and we derived a simple approximation of it for large values of J, suggesting that the it increases exponentially with J.

We presented also some illustrative simulations. Even if the process was more complicated than a random walk, simulations showed an exponential improvement in rate of convergence as J increases, very similar to the one we characterized in the simpler case.

7.1 Open Issues

Many intriguing issues still deserve to be explored and the material presented here suggests, we hope, some further stimulating questions. We would like to conclude this dissertation summarizing the issues we believe to be the most interesting and relevant.

A first remark is that, although the optimal solution to many estimation problems depends on the average of the initial conditions, there are algorithms which do not guarantee convergence to the average, nonetheless providing good performances. Therefore, there is a definite need to better understand the trade-offs between performance, rate of convergence, communication complexity and noise sensitivity for different consensus strategies on real wireless sensor networks.

Indeed, we showed how it is possible to cast a wide class of problems into the consensus framework, including problems in which the agents have to actually agree on a common estimate of few parameters (like in the least-square fitting), and problems in which every agent has to estimate its own parameter (like in the offset-removal algorithm). This duality deserves further investigation, possibly leading to a tighter relation between consensus algorithms [14] and asynchronous parallel iterative methods for the solution

of system of linear equations [55].

There is also an intriguing issue on a pure linear algebra point of view: many numerical experiments support the conjecture that the majorization inequality proposed in Theorem 12 holds true also in the case of stochastic, possibly non-normal, matrices. Nevertheless, we expect an possible proof of such a conjecture to leverage on arguments different from the ones we used here. In fact, normality is a key assumption in our proof and we do not see how to take advantage of stochasticity in our arguments.

Moreover the analysis of the distributed Kalman Filter algorithm proposed in Chapter 5 is restricted to the case of a random walk and an extension to more complicated models would certainly be of interest, although by no mean trivial.

Simulations presented in Chapter 5 show that optimization of the selection probabilities does not give a significant improvement in the estimation performance. Whether or not this is a general fact for the proposed algorithm is an issue that deserves to be explored in detail.

More on a long-term perspective, a systematic answer to the previous question would represent a first important step toward the understanding of a crucial issue: is optimization an effective and meaningful approach to complex networks of thousands of interacting devices?

The algorithm presented in Chapter 6 represents the very first step toward more complex algorithms for the estimation of different but correlated time-varying quantities, such as variables that depends both on time and on space. The tremendous practical relevance of this set-up makes its investigation an intriguing challenge. In fact, effective distributed estimation techniques for quantities varying both in time and in space could have relevant impact in many applications, ranging form environmental or scientific monitoring, temperature profiling, estimation of polluting agent diffusion, voltage drop along electrical lines or evaluation sea wave strength. More generally, whenever a sensor network is required to monitor a physical quantity described by partial differential equations these techniques come into play.

Further steps along this research should start from the extension of the proposed method, and its convergence analysis, to more general and complex graphical models. Afterwards the case of a time-varying signal needs to be addressed.

Bibliography

[1] Special Issue Proceedings of the IEEE, editor. *Sensor networks and applications*, volume 91, number 8, August 2003.

[2] Saverio Bolognani, Simone Del Favero, Luce Schenato, and Damiano Varagnolo. Distributed sensor calibration and least-square parameter identification in wsns using consensus algorithms. In *Proc. 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 1191–1198, September 23–26, 2008.

[3] Saverio Bolognani, Simone Del Favero, Luca Schenato, and Damiano Varagnolo. Consensus-based distributed sensor calibration and least-square parameter identification in wsns. *International Journal of Robust and Nonlinear Control*, to appear.

[4] Simone Del Favero and Sandro Zampieri. Distributed estimation through randomized gossip Kalman filter. In *Combined 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, Shanghai, China*, December 16 - 18 2009.

[5] Simone Del Favero and Sandro Zampieri. A majorization inequality and it application to distributed kalman filtering. *Automatica*, submitted.

[6] B. Bell and G. Pillonetto. A distributed kalman smoother. In *Proceedings of the 1st IFAC Workshop on Estimation and Control of Networked Systems - NecSys'09, Venice, Italy*, 2009.

[7] G. Pillonetto, S. Del Favero, and B. Bell. Unconstrained and inequality constrained distributed kalman smoother. *Automatica*, submitted.

[8] J.N. Tsitsiklis and M. Athans. Convergence and asymptotic agreement in distributed decision problems. *IEEE Transactions on Automatic Control*, 29(1):42–50, 1984.

[9] J. N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation*. PhD thesis, MIT, LIDS, November 1984.

[10] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, Sep. 1986.

[11] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. on Automatic Control*, 48(6):988–1001, June 2003.

[12] H. Tanner, A. Jadbabaie, and G.J. Pappas. Flocking in fixed and switching networks. *IEEE Transaction on Automatic Control*, submitted.

[13] A. Tahbaz-Salehi and A. Jadbabaie. On consensus in random networks. In *Proceedings of Allerton Conference on Communications, Control and Computing*, pages 1315–1321, September 2006.

[14] R. Olfati Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in multi-agent networked systems. *Proceedings of IEEE*, 95(1):215–233, January 2007.

[15] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri. Communication constraints in the average consensus problem. *Automatica*, 44(3):671–684, March 2008.

[16] Ruggero Carli. *Topics on the average consensus problem*. PhD thesis, Università di Padova, 2008.

[17] Qing Huia and Wassim M. Haddad. Distributed nonlinear control algorithms for network consensus. *Automatica*, 44(9):2375 – 2381, 2008.

[18] Leonidas Georgopoulos and Martin Hasler. Nonlinear Average Consensus. In *Proceedings of the 2009 International Symposium on Nonlinear Theory and its Applications*, pages 10–13, Sapporo, Hokaido, 2009. IEICE.

[19] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, September 2007.

[20] F. Fagnani and S. Zampieri. Randomized consensus algorithms over large scale networks. *IEEE Journal on Selected Areas of Communications*, to be published.

[21] Akshay Kashyap, Tamer Basar, and R. Srikant. Quantized consensus. *Automatica*, 43(7):1192 – 1203, 2007.

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

[23] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(1):65–78, September 2004.

[24] L. Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, Feb. 2005.

[25] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 49(9):1520–1533, Sept. 2004.

[26] F. R. Gantmacher. *The theory of matrices*. New York: Chelsea, 1959.

[27] O.L.V. Costa, M.D. Fragoso, and R.P. Marques. *Discrete-Time Markov Jump Linear Systems (Probability and its Applications)*. Springer, 2005.

[28] Garin F. and Zampieri S. Performance of consensus algorithms in large-scale distributed estimation. In *Proceedings of the European Control Conference 2009*, pages 755–760, 23-26 Aug. 2009.

[29] Carli R., Garin F., and Zampieri S. Quadratic indices for the analysis of consensus algorithms. In *Proceedings of Information Theory and Applications Workshop*, pages 96–104, 8-13 Feb. 2009.

[30] M. Cao, A. S. Morse, and B. D. O. Anderson. Reaching a consensus in a dynamically changing environment–a graphical approach. *SIAM Journal on Control and Optimization*, submitted.

[31] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proccedings of the Information Processing for Sensor Networks (IPSN'05)*, 2005.

[32] S. Boyd L. Xiao and S. Lall. Distributed average consensus with time-varying metropolis weights. Available at `http://www.stanford.edu/~boyd/papers/avg_metropolis.html`, 2006.

[33] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory/ACM Transactions on Networking*, 52(6):2508–2530, June 2006.

[34] F. Fagnani and S. Zampieri. Randomized consensus algorithms over large scale networks. *submitted to IEEE Trans. on Selected areas in communication*, 2007.

[35] F. Fagnani and S. Zampieri. Average consensus with packet drop communication. *SIAM J. on Control and Opt.*, to be published.

[36] D. Bauso, L. Giarré, and R. Pesenti. Nonlinear protocols for optimal distributed consensus in networks of dynamic agents. *Systems and Control Letters*, 2006.

[37] J. Cortés. Distributed algorithms for reaching consensus on general functions. *Automatica*, 44:726–737, 2008.

[38] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1997.

[39] D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Distributed sensor fusion using dynamic consensus. In *Proccedings of the 16th IFAC World Congress*, July 2005.

[40] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri. Distributed Kalman filtering based on consensus strategies. *IEEE Journal in Selected Areas of Communications*, 26(4):622 – 633, May 2008.

[41] Konrad Lorincz and Matt Welsh. Motetrack: a robust, decentralized approach to rf-based location tracking. *Personal and Ubiquitous Computing*, 11(6):489–503, 2006.

[42] Umberto Spagnolini and A.V. Bosisio. Indoor localization by attenuation maps: model-based interpolation for random medium. In *ICEAA Intl. Conf. on Electromagnetics in Adv. Application*, Sept. 2005.

[43] Lingxuan Hu and David Evans. Localization for mobile sensor networks. In *Tenth Annual International Conference on Mobile Computing and Networking*. MobiCom, 2004.

[44] K. Whitehouse and D. Culler. Calibration as parameter estimation in sensor networks. In *Proceedings of the 1st ACM Intl. Workshop on Wireless sensor networks and applications*, pages 59–67, 2002.

[45] N. Patwari, A. O. Hero III, M. Perkins, N.S. Correal, and R.J. O'Dea. Relative location estimation in wireless sensor networks. *IEEE transaction on signal processing*, 2002.

[46] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[47] M. Gudmundson. Correlation model for shadow fading in mobile radio systems. *Electronics Letters*, 27(23):2145–2146, November 1991.

[48] Chipcom AS. *SmartRF CC2420 Datasheet, rev. 1*, November 2003.

[49] Moteiv Corporation. *Tmote Sky Datasheet, rev. 1.0.4*, November 2006.

[50] G. Zanca and F. Zorzi. Measurements on CC2420 radio chipset. Technical report, Department of Information Engineering, University of Padova, Italy, 2008.

[51] Ilaria Solida. *Localization services for IEEE802.15.4/Zigbee devices. Mobile Node Tracking (in Italian)*. Master's thesis, Department of Information Engineering, University of Padova, 2007.

[52] J. L. Doob. *Stochastic Processes*. John Wiley & Sons, Inc., New York, 1953.

[53] M. Dominguez-Duran, D. Claros, C. Urdiales, F. Coslado, and F. Sandoval. Dynamic calibration and zero configuration positioning system for WSN. In *IEEE MELECON 08*, May 2008.

[54] F. Fagnani and S. Zampieri. Randomized consensus algorithms over large scale networks. *Information Theory and Applications Workshop*, pages 150–159, February 2007.

[55] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[56] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1990.

[57] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1991.

[58] Rajendra Bathia. *Matrix Analysis*. Springer, 1997.

[59] Albert W Marshall and Ingram Olkin. *Inequalities: Theory of Majorization and Its Applications*, volume 143 of *Mathematics in science and engineering*. Academic press, New York, 1979.

[60] R. Olfati-Saber. Ultrafast consensus in small-world networks. In *Proceedings of the 2005 American Control Conference (ACC'05)*, volume 4, pages 2371– 2378, 2005.

[61] R. Olfati-Saber. Distributed kalman filter with embedded consensus filters. In *Proceedings of the 44th IEEE Conference on Decision and Control, and European Control Conference*, December 2005.

[62] Peter Alriksson and Anders Rantzer. Distributed Kalman filtering using weighted averaging. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, Kyoto, Japan, July 2006.

[63] Peter Alriksson and Anders Rantzer. Model based information fusion in sensor networks. In *Proceedings of the 17th IFAC World Congress*, Seoul, Korea, July 2008.

[64] A. Speranzon, C. Fischione, and K.H. Johansson. Distributed and collaborative estimation over wireless sensor networks. In *Proceedings of the IEEE Conference on Decision and Control (CDC'06)*, pages 1025–1030, December 2006.

[65] Ioannis D. Schizas, Georgios B. Giannakis, Stergios I. Roumeliotis, and Alejandro Ribeiro. Anytime optimal distributed Kalman filtering and smoothing. *Statistical Signal Processing, 2007. SSP '07. IEEE/SP 14th Workshop on*, pages 368–372, Aug. 2007.

[66] Danny Bickson, Ori Shental, and Danny Dolev. Distributed Kalman filter via Gaussian belief propagation. *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 628–635, Sept. 2008.

[67] U.A. Khan and J.M.F. Moura. Distributing the kalman filter for large-scale systems. *Signal Processing, IEEE Transactions on*, 56(10):4919–4935, Oct. 2008.

[68] M.I. Jordan. *Learning in Graphical Models*. The M.I.T. Press, Cambridge, MA, 1998.

[69] H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*. Chapman and Hall, 2005.

[70] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

[71] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov chain Monte Carlo in Practice*. London: Chapman and Hall, 1996.

[72] W.K. Hastings. Monte carlo sampling methods using markov chain and their applications. *Biometrika*, 57:97–109, 1970.

[73] A. Gelb. *Applied Optimal Estimation*. The M.I.T. Press, Cambridge, MA, 1974.

[74] A.H. Tewfik, A.S. Willsky, and B.C. Levy. Parallel smoothing. *Systems and Control Letters*, 14:253–259, 1990.

[75] B.M. Bell, J. Burke, and G. Pillonetto. An inequality constrained nonlinear kalman-bucy smoother by interior point likelihood maximization. *Automatica*, 45:25–33, 2009.

[76] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, N.J., USA, 1979.

[77] B.M. Bell and G. Pillonetto. Approximating the bayesian inverse for nonlinear dynamical systems. *Journal of Physics: Conference Series 124*, 2008.

[78] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Athena Scientific, Belmont, Massachusets, 1997.

[79] D.G. Luenberger. *Linear and Nonlinear Programming*. Kluwer, 2003.

[80] T. Kailath. *Linear Systems*. Prentice Hall, 1979.

[81] B. Oksendal. *Stochastic Differential Equations*. Springer, 2003.

[82] G. De Nicolao and G. Ferrari Trecate. Regularization networks for inverse problems: a state space approach. *Automatica*, 39:669–676, 2003.

[83] G. Wahba. *Spline models for observational data*. SIAM, Philadelphia, 1990.

[84] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, New York, 1970.

List of Figures

List of Tables