High dimensional consensus in large-scale networks: Theory and applications

Usman Ahmed Khan

August 28, 2009

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering

Thesis Advisor: Professor José Manuel Fonseca de Moura

Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213

Dedicated to my parents,
Khalid and Aroona

Abstract

In this thesis, we develop the theory of *High Dimensional Consensus* (HDC), a general class of distributed algorithms in large-scale networks. HDC relies only on (i) local information, (ii) local communication, and (iii) low-order computation, and, hence, is ideally suited to implement network tasks under resource constraints, e.g., in sparse networks with a limited computation budget. HDC, in general, is iterative because the underlying sparsity of the network limits the information flow. Each HDC iteration is a simple (linear or non-linear) update at each node in the network. In this context, HDC partitions the network nodes into two classes: (i) sensors, nodes that update their state as some function of their neighboring nodes; and (ii) anchors, nodes whose states are fixed. HDC includes as special cases several existing well-known algorithms, for example, average-consensus and the Jacobi algorithm. We show also how to cast a banded matrix inversion algorithm in the HDC framework.

Using HDC, we derive a novel sensor localization algorithm that is distributed, iterative, and linear. With this algorithm, each sensor (with unknown location) updates its location estimate as a convex combination of its neighbors, where the coefficients of the convex combination are the barycentric coordinates computed locally by Cayley-Menger determinants. We show that this localization algorithm converges to exact sensor locations, if all the sensors lie in the convex hull of a minimal number, $m + 1$, of anchors (with known locations), where m is the dimension of the space.

We divide our theoretical contributions regarding HDC into two parts: (i) analysis of HDC; and (ii) synthesis of HDC. The analysis part studies the convergence of HDC, establishes the conditions under which HDC converges, and derives its convergence rate. It shows that the HDC converges under very general conditions, in particular, linear non-convex updates, where the updating coefficients may be negative. The synthesis part designs the HDC, for example, its parameters and weights, under network constraints, such that it converges to a pre-specified state. The thesis presents practical applications of HDC to very diverse areas including: (i) average-consensus with non-linear updates; (ii) distributed sensor localization; (iii) distributed banded matrix inversion; (iv) distributed estimation in complex dynamical systems; and (v) modeling, estimation, and inference in electrical power grids.

Contents

List of Figures

Chapter 1

Introduction

Advances in integrated electronics, radio frequency (RF) technologies, and sensing devices have made it possible to deploy large number of cheap sensors in order to implement key tasks in complex large-scale dynamical systems, such as, monitoring, tracking, estimation, and control. A centralized protocol to implement such tasks, although possibly optimal, is neither robust nor scalable because (i) the large-scale systems are very high-dimensional, and thus require extensive computations to implement a centralized protocol; and (ii) the span of the geographical region, over which the large-scale system is deployed, poses a large communication burden to implement a centralized procedure. A computationally efficient implementation is to employ a distributed algorithm that relies only on local communication and low-order computation. A distributed algorithm views a complex systems as a network of dynamic agents interacting over a graph. Hence, design and analysis of distributed algorithms is vital for efficient and scalable operation of large-scale complex infrastructures.

As a motivation, consider a detection problem where the state of the environment is monitored "locally" by sensors; each sensor makes a measurement, based on which it may make a local decision—the current state of the sensor. A problem of interest is how to fuse these local decisions. A centralized detection scheme is to send these states to a fusion center where the optimal detector is formulated; this has been considered extensively in the literature since the early work in [2, 3, 4, 5], and, more recently, [6, 7]. This centralized architecture, which may have several advantages, is neither

robust (as it has a single point of failure) nor scalable when the size of the system grows, because of heavy computational burden at the center and resource (bandwidth, power, etc.) constraints at the sensors. An alternative architecture, under resource constraints, is a web like network topology equipped with a distributed inference algorithm, where each sensor updates its own local detector by interacting with its neighboring sensors, such that the local decisions converge to the optimal centralized decision.

Distributed algorithms have been studied widely in the literature. Early references include [3, 8, 9, 10], which provide a generic theory for developing and analyzing distributed and parallel algorithms. Recently, there has been renewed interest in the sensor network community on the so called "consensus" problems and its various generalizations. Consensus can be broadly interpreted as a distributed protocol in which the sensors collaborate to reach an agreeable state. Agreement and consensus have been important problems in distributed computing, [11, 12]. The problem of dynamic load balancing for distributed multiprocessors leads to an algorithm that is essentially consensus. In the multi-agent and control literature, reference [13] develops a model for emergent behavior, schooling, and flocking described in [14, 15].

Much research effort has been directed to the average-consensus problem [13, 16, 17, 18]. In average-consensus, the goal is to converge to the average of the initial states at the sensors. Reference [16] identifies the algebraic connectivity of the underlying graph as controlling the convergence rate of the continuous-time average-consensus algorithm. For additional comments and a survey of consensus for multi-agent coordination see [19, 20], and the references there in. Consensus and its generalizations, [21, 22, 18, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32], form the building block of a large number of distributed protocols, including flocking, [33, 34], multi-vehicle coordination, [20], and distributed estimation [35, 36, 37, 38, 39].

In this thesis, we consider such distributed algorithms in broad generality and provide a systematic study of classes of these algorithms. We explore average-consensus algorithms in a larger context where the sensors may converge to an arbitrary weighted combination of their initial states (not just their average). We term such distributed algorithms *High Dimensional Consensus* (HDC). To capture in the same framework

a broader class of algorithms, we consider the notion of anchors. Anchors are the nodes in the network that maintain a fixed state throughout the algorithm. The anchors may play the role of leaders of the system that drive the algorithm in a certain direction, see also [40]. The sensors[1] iteratively update their states as some function (linear or non-linear) of the states of their neighboring nodes. It is essential that the updating functions are computed using only local information that is available at the sensor itself or at the neighboring nodes. This requirement assures that the algorithms remain decentralized and distributed.

We develop a comprehensive theory of HDC. We divide this development into two parts: (i) analysis of HDC; and (ii) synthesis of HDC. In the analysis part, we address the question: given the HDC parameters (i.e., the updating functions at the nodes) and the underlying connectivity graph, determine: (a) under what conditions does the HDC converge; (b) to what state does the HDC converge; and (c) what is the convergence rate. In the synthesis part, we address the question: given the desired state to which HDC should converge and the sparsity of the underlying communication graph, *learn* the HDC parameters, so that HDC converges to the desired state. We then focus on relevant practical applications where HDC is specialized to perform several networked tasks. The applications we provide include (i) distributed sensor localization; (ii) distributed banded matrix inversion; (iii) distributed estimation in complex dynamical systems; and (iv) modeling, estimation, and inference in electric power systems.

In the context of electric power systems, we provide a structure-preserving model that we term as *cyber-physical model*. In this model, we view the electric power system as a network of interacting dynamical agents. We divide the nodes in this network into physics-based modules and cyber-based modules. Physics-based modules, e.g., generators, are the nodes that can be described completely using their physical descriptions (e.g., partial differential equations). On the other hand, cyber-based modules are the nodes that cannot be described using underlying physics because they represent a diverse mixture of several sub-modules appearing in and/or disappearing from the

[1] In the sequel, unless specifically stated or obvious from the context, a sensor implies a non-anchor node.

network at random times. We use statistical identification techniques to model these modules. An example is aggregate electrical load that is a mixture of several household and commercial appliances/equipments randomly turned on and off; modeled by an autoregressive process. Such representation of the electric power systems provides a foundation for *distributed and computationally efficient* implementation of several key tasks, such as, load balancing, solving power flows, dynamic estimation, control, and, inference. Clearly, HDC is fundamental in implementing these tasks.

Parts of this thesis have been partially supported by U.S. National Science Foundation ITR Project Number CNS-0428404. The results in this thesis have been published[2] in the following journals, book chapters, and proceedings, [35, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57].

We now summarize our main contributions in the next section.

1.1 Contributions

We now summarize the main contributions of this thesis.

1. **Distributed sensor localization:** We introduce a novel distributed localization algorithm, DILOC, for sensor localization that requires a minimal number of anchors. Anchors are the nodes that know their exact locations, thus, provide an absolute frame of reference to the algorithm. DILOC is a linear, iterative algorithm, where each sensor updates its location estimate as a convex combination of its carefully chosen neighbors. The weights of the linear combinations are the barycentric coordinates computed from local distances and Cayley-Menger determinants. We study several enhancements of the localization problem including dynamic network topologies, localization in mobile networks, and localization with noisy distance measurements. In addition, we characterize the qualitative behavior of DILOC in randomly deployed sensor networks. Since DILOC updates are convex (non-negative weights that sum to one), the convergence of DILOC can be mapped to the convergence of absorbing

[2]Some material is in the process of publication.

Markov chains to a steady state distribution.

DILOC provides an interesting setup for distributed algorithms. It splits the network nodes into sensors (nodes that update their states) and anchors (nodes whose states are fixed). As the thesis shows, this setup can be extended by generalizing DILOC to non-convex and non-linear updates to address problems different from localization. In other words, we can relax the convexity, linearity, and non-negativity assumptions in DILOC. This leads to a generalized class of distributed algorithms that we term High Dimensional Consensus (HDC).

2. **High Dimensional Consensus (HDC):** We introduce HDC that is a unified framework for distributed algorithms in large-scale networks. HDC updates the state at each node as a function (linear or non-linear) of the states at the neighboring nodes. In HDC, the updates can have negative coefficients or be non-convex. Under appropriate conditions, the network reaches an agreement or a consensus, i.e., the network reaches a *limiting state* (a function of the initial states of the nodes). HDC includes several existing and well-known algorithms as special cases when we restrict the updating functions to be of a particular form, for example, the Jacobi algorithm [58], and linear average-consensus [17]. Clearly, the sensor localization algorithm we introduce is a special case of HDC. In the context of HDC, we address the following issues.

    - *Analysis: Convergence.* We establish appropriate conditions for convergence of the HDC, and derive the limiting state of the network given the updating functions.
    - *Synthesis: Design.* We design the updating functions in HDC such that we achieve a desired limiting state.
    - *Random environments:* We study the behavior and performance of HDC in random environments (communication noise, data packet drops, and imperfect knowledge of the underlying system).

3. **Non-linear average-consensus:** To speed the convergence of linear average-consensus [17], we consider a non-linear version. We define general conditions

under which the non-linear HDC converges to average-consensus and study HDC with sinusoidal updating functions.

4. **Distributed Kalman filter:** For large-scale sparsely-connected dynamical systems, we provide a distributed estimation algorithm based on the spatial decomposition of the dynamical systems into sub-systems. At the heart of our estimation algorithm lies a distributed banded matrix inversion algorithm that we term Distributed Iterated Collapse Inversion (DICI) algorithm. DICI is a special case of HDC (appended with a non-linear collapse operator) that assimilates the local error covariances among the sub-systems in a computationally efficient and completely decentralized fashion.

5. **Applications to electric power grids:** We study the following applications of HDC and in the context of large-scale electric power systems:

   - *Estimation and modeling:* We explore the practical significance of our distributed Kalman filter in the context of a structure-preserving model of the electric power system that we term cyber-physical model. We show that cooperation among the (potentially unobservable) sub-systems, derived from the cyber-physical model, leads to the observability of the overall system.
   - *Phase-angle estimation:* We provide a distributed inference algorithm for phase-angle estimation that is based on the HDC algorithm and borrows some concepts from our distributed localization algorithm. In this context, we study the minimal number of Phasor Measurement Units (PMUs) and their optimal placement.

In the next section, we set some standard notation.

1.2 Notation

Consider a network of N nodes described by a communication graph, $\mathcal{G} = (\Theta, \mathbf{A})$, where $\Theta = \{1, \ldots, N\}$ is the set of vertices. The interconnections among the nodes

are given by the adjacency matrix, $\mathbf{A} = \{a_{lj}\}$, where

$$a_{lj} = \begin{cases} 1, & l \leftarrow j, \\ 0, & \text{otherwise}, \end{cases} \tag{1.1}$$

and $l \leftarrow j$ implies that node j can send information to node l. We define $\mathcal{K}(l)$ as the neighbors of node l, i.e.,

$$\mathcal{K}(l) \triangleq \{j \mid a_{lj} = 1\}. \tag{1.2}$$

We partition this network into K anchors and M sensors, such that $N = K + M$. As highlighted before, the anchors are the nodes whose states are fixed, and the sensors are the nodes that update their states as some functions of the states of their neighboring nodes. Let $\kappa = \{1, \ldots, K\}$ be the set of anchors and let $\Omega = \{K + 1, \ldots, N\}$ be the set of sensors. The set of all of the nodes is then denoted by $\Theta = \kappa \cup \Omega$. The notion of anchors and sensors induces a natural partitioning on this neighborhood, i.e., the set of neighbors that are sensors,

$$\mathcal{K}_\Omega(l) = \mathcal{K}(l) \cap \Omega, \tag{1.3}$$

and the set of neighbors that are anchors,

$$\mathcal{K}_\kappa(l) = \mathcal{K}(l) \cap \kappa. \tag{1.4}$$

Similarly, we define the extended neighborhood at node l,

$$\mathcal{D}_l \triangleq \{l\} \cup \mathcal{K}(l). \tag{1.5}$$

Let K be the total number of edges in $\mathcal{G}$. Let $\mathbf{C} = \{c_{lk}\}_{l=1,\ldots,N}^{k=1,\ldots,K}$ be the $N \times K$ incidence matrix of $\mathcal{G}$ where its kth column represents the kth edge, $(i, j) \in \mathcal{G}$, such that $c_{ik} = 1$ and $c_{jk} = -1$. The Laplacian, $\mathbf{L}$, of $\mathcal{G}$ is then defined as

$$\mathbf{L} = \mathbf{C}\mathbf{C}^T. \tag{1.6}$$

If w_k is a weight associated to the kth edge in $\mathcal{G}$, then the weighted Laplacian matrix is defined as

$$\mathbf{L}_w = \mathbf{C}\mathbf{W}\mathbf{C}^T, \tag{1.7}$$

where $\mathbf{W}$ is a $K \times K$ diagonal matrix such that the kth element on its diagonal (that represents the kth edge in $\mathcal{G}$) is w_k. Note that the Laplacian, $\mathbf{L}$, is symmetric and positive-semidefinite. Hence, its eigenvalues are real and non-negative. If $\mathbf{W} \geq 0$ (where $\geq$ denotes element-wise inequality), then $\mathbf{L}_w$ is also symmetric, positive-semidefinite, see [59] for details. For a connected graph, $\mathcal{G}$, we have, $\lambda_2(\mathbf{L}) > 0$, where $\lambda_2(\mathbf{L}) > 0$ is the second largest eigenvalue of $\mathbf{L}$. If $\lambda_2(\mathbf{L}) > 0$ and $\mathbf{W} > 0$, then $\lambda_2(\mathbf{L}_w) > 0$ [59].

As a graph can be characterized by its adjacency matrix, to every matrix we can associate a graph. For a matrix, $\mathbf{\Upsilon} = \{\upsilon_{lj}\} \in \mathbb{R}^{N\times N}$, we define its associated graph by $\mathcal{G}^{\mathbf{\Upsilon}} = (V^{\mathbf{\Upsilon}}, \mathbf{A}^{\mathbf{\Upsilon}})$, where $V^{\mathbf{\Upsilon}} = \{1, \ldots, N\}$ and $\mathbf{A}^{\mathbf{\Upsilon}} = \{a_{lj}^{\mathbf{\Upsilon}}\}$ is given by

$$a_{lj}^{\mathbf{\Upsilon}} = \begin{cases} 1, & \upsilon_{lj} \neq 0, \\ 0, & \upsilon_{lj} = 0. \end{cases} \tag{1.8}$$

Let $\lambda_{i(\mathbf{P})}$ denotes the ith eigenvalue of a matrix, $\mathbf{P} \in \mathbb{R}^{M\times M}$. We use $\rho(\mathbf{P})$ to denote the spectral radius of $\mathbf{P}$, defined as

$$\rho(\mathbf{P}) = \max_i |\lambda_{i(\mathbf{P})}|. \tag{1.9}$$

For any matrix induced norm, $\|\cdot\|$, recall that

$$\rho(\mathbf{P}) \;\; = \;\; \lim_{q\to\infty} \|\mathbf{P}^q\|^{1/q}. \tag{1.10}$$

The notation set in this section is, mostly, standard for the rest of the thesis, unless specifically modified in the appropriate chapter or obvious from the context.

1.3 Problem statement

For $k \in \kappa$, let

$$\mathbf{u}_k(t) = [u_{k,1}, \ldots, u_{k,m}], \tag{1.11}$$

be an m-dimensional row-vector that denotes the state of the kth anchor at time t. Similarly, for $l \in \Omega$, let

$$\mathbf{x}_l(t) = [x_{l,1}, \ldots, x_{l,m}], \tag{1.12}$$

be an m-dimensional row-vector that denotes the state of the lth sensor at time t. The general form of the HDC is given by

$$\begin{array}{lll} \mathbf{u}_k(t+1) = & \mathbf{u}_k(t), & k \in \kappa, \\ x_{l,j}(t+1) = & f_l\left(x_{\mathcal{D}_l,j}\left(t\right)\right) + g_l\left(u_{\mathcal{K}_\kappa(l),j}\left(t\right)\right), & l \in \Omega,\ j \in \{1, \ldots, m\}, \end{array} \tag{1.13}$$

where $x_{\mathcal{D}_l,j}$ is the jth component of the sensor states in $\mathcal{D}_l$, $u_{\mathcal{K}_\Omega(l),j}$ is the jth component of the anchor states in $\mathcal{K}_\kappa(l)$, and, $f_l : \mathbb{R}^{|\mathcal{D}l|\times 1} \mapsto \mathbb{R}$ and $g_l : \mathbb{R}^{|\mathcal{K}_\kappa(l)|\times 1} \mapsto \mathbb{R}$ are real-valued linear or non-linear vector functions. Clearly, a single iteration of the HDC at sensor l and time t, takes the information from its neighboring nodes (sensors and anchors) and uses the updating functions, f_l and g_l, to compute the state of sensor l at time $t+1$. Furthermore, anchors do not update their state as seen from the iteration for the anchors states, $\mathbf{u}_k(t),\ k \in \kappa$. Under appropriate conditions, HDC converges to

$$\lim_{t\to\infty} \mathbf{x}_l(t+1) = w_l\left(\mathbf{x}(0), \mathbf{u}(0)\right), \tag{1.14}$$

for some appropriate real-valued vector function, $w_l : \mathbb{R}^N \mapsto \mathbb{R}$. Two interesting problems can be considered in this context that we describe in the following.

Analysis (Forward) problem: Given the functions, f_l and g_l at each sensor, the network initial conditions, $\mathbf{x}_l(0),\ \forall\ l \in \Omega$ and $\mathbf{u}_k(0),\ \forall\ k \in \kappa$, and the underlying communication graph, $\mathcal{G}$, determine (i) the conditions under which the HDC converges; (ii) to what state does the HDC converge; and (iii) what is the convergence rate.

Synthesis (Inverse) problem: Given the desired state, $w_l\left(\mathbf{x}(0), \mathbf{u}(0)\right), \forall\ l \in \Omega$, to which HDC should converge and the sparsity of the underlying communication

graph, $\mathcal{G}$, *design* the HDC parameters, so that HDC converges to the desired state.

1.4 Summary

In this section, we summarize the rest of the thesis.

- **Chapter 2:** In Chapter 2, we provide a novel framework for the analysis and synthesis of non-linear distributed average-consensus algorithms. In particular, we consider $m = 1$-dimensional sensor states (scalars) with no anchors. The HDC updates are of the form: at each sensor l,

$$x_l(t+1) = f_l\left(x_l(t), x_{\mathcal{K}_\Omega(l)}(t)\right). \tag{1.15}$$

  In the analysis part, we establish the conditions on the local updating functions, f_l, such that (1.15) converges to the average of the initial sensor states, i.e.,

$$\lim_{t\to\infty} x_l(t+1) = \frac{1}{M}\sum_{i=0}^{M} x_i(0), \qquad \forall\ l. \tag{1.16}$$

  In the synthesis part, we design the local functions, f_l, such that we achieve the above limiting state. In particular, we choose the local functions as a weighted sum of sinusoids. In this case, key design questions include designing the frequency and domain of the sinusoids along with the appropriate bounds on the weights in the weighted sum.

- **Chapter 3:** In Chapter 3, we consider the local functions, f_l and g_l, to be linear. In particular, the HDC updates are of the form: at each sensor l,

$$\mathbf{x}_l(t+1) = \sum_{j\in\mathcal{D}_l} p_{lj}\mathbf{x}_j(t) + \sum_{k\in\mathcal{K}_\kappa(l)} b_{lk}\mathbf{u}_k(t). \tag{1.17}$$

  In the analysis part, we establish the conditions on the local updating coefficients, p_{lj}s and b_{lk}s, such that (1.17) converges to some linear functions of the

initial anchor states, i.e.,

$$\lim_{t\to\infty} \mathbf{x}_l(t+1) = \sum_{j=0}^{K} w_{lj}\mathbf{u}_j(0), \qquad \forall\ l. \tag{1.18}$$

As a side note, average-consensus is a special case here by taking b_{lj}s to be zero and[3] $\mathbf{u}_j(0) = \mathbf{x}_j(0)$.

We then formulate the HDC in random environments, i.e., with communication link failures, communication noise, and imperfect system knowledge. We provide a modification to HDC that is robust to such random environments.

In the synthesis part, we fix the w_{lj}s in (1.18) and learn the HDC parameters, p_{lj}s and b_{lk}s, so that HDC converges to (1.18).

- **Chapter 4:** In Chapter 4, we specialize HDC to distributed sensor localization in m-dimensional Euclidean spaces ($\mathbb{R}^m$) using a minimal number, $K = m + 1$, of anchors (that know their exact locations and hence, do not update their state). The m-dimensional sensor state, $\mathbf{x}_l(t)$, $l \in \Omega$, becomes the location estimate of the sensor l at time t. We choose the HDC parameters, p_{lj}s and b_{lk}s, to be the barycentric coordinates computed locally from inter-sensor distance measurements and Cayley-Menger determinants. We show the limiting state of the distributed localization algorithm (as established in Chapter!3) is the exact sensors' locations. We extend our algorithm to networks of mobile agents and also consider sensor localization in random environments, i.e., under data packet drops, communication noise, and noisy distance measurements.

- **Chapter 5:** In Chapter 5, we specialize HDC to provide a distributed algorithm for banded matrix inversion that we call Distributed Iterate Collapse Inversion (DICI) algorithm. The DICI algorithm exploits the structure of banded matrices by appending a non-linear collapse step to the HDC. We show that the computation complexity of the DICI algorithm is independent of the size of the matrix, at each sensor.

[3]For precise statements on this comparison, see Chapter 3.

- **Chapter 6:** In Chapter 6, we use the DICI algorithm to implement a distributed estimation (Kalman filter) algorithm for large-scale complex dynamical systems. Our algorithm spatially decomposes the large-scale system into several coupled local sub-systems. We implement local Kalman filters at these sub-systems and use the HDC and DICI algorithms for observation fusion and covariance assimilation. The resulting distributed estimator is computationally efficient, scalable, and completely decentralized.

- **Chapter 7:** In Chapter 7, we consider applications in electric power systems (smart grids). We provide a novel modeling paradigm that we term as *cyber-physical model*. Cyber-physical models are structure-preserving; they provide the key structure required to implement the distributed Kalman filter of Chapter 6 on electric power systems. Finally, we provide a distributed inference algorithm for phase-angle estimation in electric power systems.

- **Chapter 8:** In Chapter 8, we conclude this thesis and recapitulate our contributions.

Note that each chapter contains a detailed set of references, along with comparisons and contrasts of the work presented in the chapter with existing work in that domain.

Chapter 2

Nonlinear average-consensus algorithm

In this chapter, we specialize HDC to present a distributed average-consensus algorithm with non-linear updates. In average-consensus, there are no anchors (recall that the anchors do not update their state), since each node in the network should converge to the average of the network initial conditions. Hence, we use the standard form of HDC (1.13) without anchors and consider the analysis and the synthesis problems. In the analysis problem, we establish conditions on the local updating functions, f_l, such that HDC converges to the average of the network initial conditions. In the synthesis problem, we design one such updating function, along with its appropriate parameters. In particular, we use the updating functions as a weighted sum of sinusoids and derive appropriate bounds on its parameters (frequency, domain) and the weights (of the weighted sum) to achieve the desired convergence properties. By simulations, we show that the convergence rate of the non-linear algorithm outperforms the convergence rate of the conventional linear case.

Parts of this chapter have been presented in [55].

2.1 Introduction

Recently, there has been significant interest in the linear distributed average-consensus (LDAC) problem [17]. LDAC computes the average of several scalars distributed over a sensor network. These scalars become the initial conditions of the LDAC algorithm. LDAC updates the state at each sensor by a *linear* combination of the states at the neighboring sensors. Under appropriate conditions [17, 20], the state at each sensor converges to the average of the initial conditions. So far, the related literature has focussed on linear updates, where the convergence rate only depends on the algebraic network connectivity (second largest eigenvalue of the graph Laplacian).

In this chapter, we introduce a distributed average-consensus algorithm with *non-linear* updates. Our algorithm is a special case of HDC introduced in (1.13) without anchors. Average-consensus does not require anchors, since each node in the network should converge to the average of the network initial conditions. In this context, we address (i) the analysis problem, and (ii) the synthesis problem. In the analysis problem, we establish the *conditions* required for any non-linear updating function to achieve average-consensus. On the other hand, in the synthesis problem, we fix a suitable non-linear function and design its parameters such that it satisfies the aforementioned conditions.

In particular, we select the updating functions as a weighted sum of sinusoids and show that satisfy the required conditions by appropriately choosing their frequency and domain. The state update in the non-linear distributed average-consensus (NLDAC) consists of the sensor's previous state added to a linear combination of the sine of the state differences among the neighboring nodes. Due to the non-linearity introduced, the convergence rate now depends on the actual states of the nodes. As will be shown in the chapter, this fact makes the convergence rate of NLDAC faster, by appropriate tuning of the combining weights.

Our work can be tied to results on networks of coupled oscillators (see, for example, [60, 61, 62, 63, 64]). These works are concerned with qualitative properties of such networks. In contrast, we propose schemes to design algorithms with desirable convergence properties (in our case, for average-consensus), and our methodology is

different. The framework presented here goes beyond average-consensus and is likely to find applications in other areas of distributed signal processing, e.g., distributed phase-locked loops [65], or large-scale power networks [66], where such form of dynamics arise naturally.

For simplicity of the exposition, we assume that each sensor, l, possesses a scalar quantity, y_l, such that $y_l \in [-\pi/4 + \varepsilon, \pi/4 - \varepsilon]\ \forall\ l$ with $\varepsilon \in \mathbb{R}_{>0}$. This is needed because the domain of the sine function should be such that the $\cos(y_l - y_j) \neq 0,\ \forall\ l$ and j. This condition is on the cosine rather than the sine function because, as will be shown, the cosine determines the convergence rate as it is the derivative of the sine. In general, when $y_l \notin [-\pi/4 + \varepsilon, \pi/4 - \varepsilon]$ for some l, we only require the y_l's to be bounded, i.e., $|y_l| < M,\ \forall\ l$, since we can always choose the frequency, ζ, such that $\cos(\zeta(y_l - y_j))$ does not vanish. The resulting convergence rate involves the additional degree of freedom, ζ.

We now describe the rest of the chapter. Section 2.2 discusses the problem formulation and introduces the non-linear distributed average-consensus (NLDAC). We analyze the algorithm in Section 2.3 and derive the conditions for convergence. We then address the synthesis of NLDAC and design the non-linear updates in Section 2.4. In Section 2.5, we present simulations, and finally, Section 2.6 concludes the chapter.

2.2 NLDAC: Problem formulation

Along the lines of the notation described in Chapter 1, consider a network of N nodes communicating over a graph $\mathcal{G}$, where each node, l, possesses a scalar quantity, y_l. In addition, we assume that the communication graph, $\mathcal{G}$, is undirected, i.e., its adjacency matrix, $\mathbf{A}$, is symmetric. We consider distributed updates on $\mathcal{G}$ of the following form: at each node l, we have

$$x_l(t+1) = f_l(x_{\mathcal{D}_l}(t)), \qquad x_l(0) = y_l, \tag{2.1}$$

where $\mathcal{D}_l$ is the extended neighborhood of sensor l (as defined in (1.5)), $x_{\mathcal{D}_l}(t)$ denotes the states at the nodes in $\mathcal{D}_l$, and $f_l : \mathbb{R}^{|\mathcal{D}_l|} \mapsto \mathbb{R}$ is a non-linear real-valued vector

function, such that the above algorithm converges to

$$\lim_{t\to\infty} x_l(t+1) = \frac{1}{N}\sum_{j=0}^{N} x_j(0) = \frac{1}{N}\sum_{j=0}^{N} y_j, \qquad \forall l, \tag{2.2}$$

i.e., to the average of the scalar quantities, y_l, the nodes possess. In the conventional linear distributed average-consensus (LDAC) algorithm, with constant weight, μ, [17], we have

$$f_l\left(x_{\mathcal{D}_l}\right) = x_l(t) - \mu \sum_{j\in\mathcal{K}(l)} \left(x_l(t) - x_j(t)\right), \qquad \forall l. \tag{2.3}$$

In this chapter, we allow the functions, f_l, to be non-linear and address the following problems:

Analysis problem: Given the NLDAC updates (2.1) on $\mathcal{G}$, establish the conditions on f_l such that (2.1) converges to (2.2).

Synthesis problem: Given the desired convergence to (2.2), design the functions, f_l, such that the functions satisfy the conditions established in the analysis problem. In particular, we choose

$$f_l\left(x_{\mathcal{D}_l}(t)\right) = x_l(t) - \mu \sum_{j\in\mathcal{K}(l)} \sin\left(\zeta\left(x_l(t) - x_j(t)\right)\right), \qquad \forall l, \tag{2.4}$$

where μ is a weight that is constant across the entire network and ζ is the frequency. It turns out that choosing $\zeta = 1$ requires the network initial conditions, $\mathbf{x}(0)$, to lie in a certain range. When this is not the case, we provide an appropriate generalization to choose the frequency, ζ.

Matrix form: We now write (2.1) in matrix form. Define the column vectors,

$$\mathbf{x}(t) \triangleq \left[x_1(t), \ldots, x_N(t)\right]^T, \tag{2.5}$$

$$\mathbf{f}(\mathbf{x}(t)) \triangleq \left[f_1(\cdot), \ldots, f_N(\cdot)\right]^T. \tag{2.6}$$

Clearly, we have

$$\mathbf{f} : \mathbb{R}^N \mapsto \mathbb{R}^N. \tag{2.7}$$

With the above notation, algorithm (2.1) can be written compactly as

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t)). \tag{2.8}$$

From (2.2), we can now define the error in the iterations (2.8) as the following vector:

$$\mathbf{e}(t+1) \triangleq \mathbf{x}(t+1) - x_{\text{avg}}\mathbf{1}, \tag{2.9}$$

where

$$x_{\text{avg}} = \frac{1}{N}\sum_{l=1}^{N} x_l(0), \tag{2.10}$$

and $\mathbf{1}$ is a vector of N 1's. In the next section, we consider the analysis problem.

2.3 NLDAC algorithm: Analysis problem

In this section, we consider the analysis problem and provide appropriate conditions on the non-linear updating function, $\mathbf{f}$, required for average-consensus. To this aim, we give the following theorem. To state our result, we let

$$\mathbf{J} = \frac{\mathbf{1}\mathbf{1}^T}{N}. \tag{2.11}$$

Theorem 1: If $\mathbf{f}$ is continuously differentiable, such that

(i) the functions, f_l, are sum preserving , i.e.,

$$\sum_l f_l(\mathbf{x}(t)) = \sum_l x_l(t) = \sum_l y_l; \tag{2.12}$$

(ii) for any $c \in \mathbb{R}$, $\mathbf{x}^* = c\mathbf{1}$ is a fixed point of (2.8), i.e.,

$$\mathbf{x}^* = \mathbf{f}(\mathbf{x}^*); \tag{2.13}$$

(iii) for some $\theta(t) \in [0,1]$ with $\eta(t) = \theta(t)\mathbf{x}(t) + (1-\theta(t))\, x_{\text{avg}}\mathbf{1}$, we have

$$\|\mathbf{f}'(\eta(t)) - \mathbf{J}\| < 1, \qquad \forall\ t, \tag{2.14}$$

where the $N \times N$ matrix, $\mathbf{f}'$, denotes the derivative of the vector function, $\mathbf{f}$, with respect to the vector $\mathbf{x}$;

then (2.8) converges to

$$\lim_{t\to\infty} \mathbf{x}(t+1) = \mathbf{J}\mathbf{x}(0). \tag{2.15}$$

Proof: Since $\mathbf{f}$ is sum preserving, and $\mathbf{1}^T\mathbf{x}(t)$ is the sum of all the states at time t, we have $\mathbf{1}^T\mathbf{x}(t) = \mathbf{1}^T\mathbf{x}(t-1) = \ldots = \mathbf{1}^T\mathbf{x}(0)$. Hence, $\mathbf{1}^T\mathbf{x}(t)/N = x_{\text{avg}}$, and (i) implies

$$\mathbf{J}\mathbf{x}(t) = x_{\text{avg}}\mathbf{1}. \tag{2.16}$$

From (ii), we note that $x_{\text{avg}}\mathbf{1}$ is the fixed point of (2.8), i.e.,

$$x_{\text{avg}}\mathbf{1} = \mathbf{f}(x_{\text{avg}}\mathbf{1}). \tag{2.17}$$

We can now write the norm of the error vector in (2.9) as

$$\begin{aligned} \|\mathbf{e}(t+1)\| &= \|\mathbf{f}(\mathbf{x}(t)) - \mathbf{f}(x_{\text{avg}}\mathbf{1}) - \mathbf{J}(\mathbf{x}(t) - x_{\text{avg}}\mathbf{1})\|, \\ &= \|\mathbf{g}(\mathbf{x}(t)) - \mathbf{g}(x_{\text{avg}}\mathbf{1})\|, \end{aligned} \tag{2.18}$$

where $\mathbf{g} : \mathbb{R}^N \to \mathbb{R}^N$ is defined as

$$\mathbf{g}(\mathbf{x}) \triangleq \mathbf{f}(\mathbf{x}) - \mathbf{J}\mathbf{x}. \tag{2.19}$$

Since $\mathbf{f}$, and, in turn, $\mathbf{g}$, is continuously differentiable: from the mean-value theorem, there exists some $\theta(t) \in [0,1]$ with $\eta(t) = \theta(t)\mathbf{x}(t) + (1-\theta(t))\, x_{\text{avg}}\mathbf{1}$, such that

$$\begin{aligned} \|\mathbf{e}(t+1)\| &= \|\mathbf{g}'(\eta(t))\left(\mathbf{x}(t)) - x_{\text{avg}}\mathbf{1}\right)\|, \\ &\leq \|\mathbf{g}'(\eta(t))\|\|\mathbf{e}(t)\|, \\ &= \|\mathbf{f}'(\eta(t)) - \mathbf{J}\|\|\mathbf{e}(t)\|. \end{aligned} \tag{2.20}$$

From (iii), we have

$$\lim_{t\to\infty} \|\mathbf{e}(t+1)\| = 0, \tag{2.21}$$

and (2.15) follows. ■

Theorem 1 establishes the conditions required for any non-linear updating function, $\mathbf{f}$, such that NLDAC reaches average-consensus. Designing such a function lies under the purview of the synthesis problem that we consider in the following.

2.4 NLDAC algorithm: Synthesis problem

The synthesis problem entails choosing a suitable function and verifying the three conditions established in Theorem 1. In general, the conditions (i) and (ii) in Theorem 1 are relatively easy to verify, whereas, condition (iii) may impose additional design requirements on the network initial condition, $\mathbf{x}(0)$, and the parameters of the selected function. We now define the updating functions, f_l, to be of the following form:

$$f_l(x_{\mathcal{D}_l}(t)) \triangleq x_l(t) - \mu \sum_{j\in\mathcal{K}(l)} \sin\left[\zeta(x_l(t) - x_j(t))\right], \tag{2.22}$$

where μ is a weight that is constant across the entire network and ζ is the frequency. For simplicity of the exposition, we assume that $\zeta = 1$. As we will show in Section 2.4.3, this assumption does not lose generality and can be relaxed easily.

In the next subsections, we show that (2.22) satisfy the properties described in Theorem 1 and subsequently derive the conditions on μ and the network initial conditions, $\mathbf{x}(0)$, under which the following update at node l

$$x_l(t+1) = x_l(t) - \mu \sum_{j\in\mathcal{K}(l)} \sin(x_l(t) - x_j(t)), \tag{2.23}$$

with $x_l(0) = y_l$ converges to (2.2).

2.4.1 Theorem 1 for sinusoidal updating functions

We prove (i) in Theorem 1 for sine functions in the following lemma.

Lemma 1: The functions f_l given by (2.22) are sum preserving, i.e.,

$$\sum_l f_l(\mathbf{x}(t)) = \sum_l x_l(t) = \sum_l y_l. \tag{2.24}$$

Proof: We start with the L.H.S of (2.24). We have

$$\sum_l f_l(\mathbf{x}(t)) = \sum_l \left(x_l(t) - \mu \sum_{j \in \mathcal{K}(l)} \sin(x_l(t) - x_j(t)) \right). \tag{2.25}$$

In order to establish (2.24), it suffices to show that

$$\sum_l \mu \sum_{j \in \mathcal{K}(l)} \sin(x_l(t) - x_j(t)) = 0. \tag{2.26}$$

Since we assumed the communication graph, $\mathcal{G}$, to be undirected, we have

$$j \in \mathcal{K}(l) \Rightarrow l \in \mathcal{K}(j). \tag{2.27}$$

Fix an l and a $j \in \mathcal{K}(l)$, then there are two terms in the L.H.S of (2.26) that contain both l and j. For these two terms, we have

$$\mu \sin(x_l(t) - x_j(t)) + \mu \sin(x_j(t) - x_l(t)) = 0, \tag{2.28}$$

due to the fact that the sine function is odd. The above argument is true $\forall l$, and (2.24) follows. Clearly, from (2.24), we also have

$$\sum_l x_l(t+1) = \sum_l x_l(t). \tag{2.29}$$

■

In the following lemma, we establish (ii) for Theorem 1 for sine functions.

Lemma 2: Let $\mathbf{1}$ denote an $N \times 1$ column-vector of 1's. For any $c \in \mathbb{R}$, $\mathbf{x}^* = c\mathbf{1}$ is

a fixed point of (2.8).

Proof: The proof is straightforward and relies on the fact that $\sin(0) = 0$. ■

Before we proceed to establish (iii) in Theorem 1 for the sine function, we provide two important lemmas. Lemma 3 proves a result on the range of network initial conditions, $\mathbf{x}(0)$, whereas Lemma 4 computes $\mathbf{f}'$ when f_l is given by (2.22). We then continue the development in Section 2.4.2 and provide the main result of this chapter in Section 2.4.3.

Range of the NLDAC initial conditions, $\mathbf{x}(0)$

In the next lemma, we establish a condition under which $x_l(t)$ lies in a certain range.

Lemma 3: Let the NLDAC vector of initial conditions, $\mathbf{x}(0)$, be such that

$$x_l(0) \in [-\pi/4 + \varepsilon, \pi/4 - \varepsilon], \ \forall \ l, \tag{2.30}$$

where $\varepsilon \in \mathbb{R}_{>0}$ is a sufficiently small real number. Let $d_{\max}$ denote the maximum degree of the underlying communication graph, $\mathcal{G}$, and let $x_l(t)$ denotes the NLDAC updates (2.23). Then for μ in the range

$$0 < \mu \leq \frac{\pi}{2d_{\max}}, \tag{2.31}$$

we have

$$x_l(t) \in [-\pi/4 + \varepsilon, \pi/4 - \varepsilon], \ \forall \ l, t. \tag{2.32}$$

Proof: We use the following bounds to prove this lemma.

$$\frac{2}{\pi}x \leq \sin(x) \leq x, \qquad 0 \leq x \leq \frac{\pi}{2}. \tag{2.33}$$

For any arbitrary node, l, partition its neighbors, $\mathcal{K}(l)$, into $\mathcal{K}_L(l,t)$ and $\mathcal{K}_G(l,t)$, where $\mathcal{K}_L(l,t) = \{j \in \mathcal{K}(l) \mid x_l(t) > x_j(t)\}$, and $\mathcal{K}_G(l,t) = \{j \in \mathcal{K}(l) \mid x_l(t) < x_j(t)\}$.

We can write the NLDAC iterations (2.23) as

$$\begin{aligned} x_l(t+1) &= x_l(t) - \mu \sum_{j\in\mathcal{K}_L(l,t)} \sin(x_l(t) - x_j(t)) + \mu \sum_{j\in\mathcal{K}_G(l,t)} \sin(x_j(t) - x_l(t)), \\ &\leq x_l(t) - \mu \sum_{j\in\mathcal{K}_L(l,t)} \frac{2}{\pi}(x_l(t) - x_j(t)) + \mu \sum_{j\in\mathcal{K}_G(l,t)} (x_j(t) - x_l(t)), \\ &= \left(1 - \frac{2\mu}{\pi}|\mathcal{K}_L(l,t)| - \mu|\mathcal{K}_G(l,t)|\right) x_l(t) \\ &+ \frac{2\mu}{\pi} \sum_{j\in\mathcal{K}_L(l,t)} x_j(t) + \mu \sum_{j\in\mathcal{K}_G(l,t)} x_j(t), \end{aligned} \tag{2.34}$$

where the inequality follows from (2.32) and the bound in (2.33). Similarly, we can show

$$\begin{aligned} x_l(t+1) &\geq x_l(t) - \mu \sum_{j\in\mathcal{K}_L(l,t)} (x_l(t) - x_j(t)) + \mu \sum_{j\in\mathcal{K}_G(l,t)} \frac{2}{\pi}(x_j(t) - x_l(t)), \\ &= \left(1 - \mu|\mathcal{K}_L(l,t)| - \frac{2\mu}{\pi}|\mathcal{K}_G(l,t)|\right) x_l(t) \\ &+ \mu \sum_{j\in\mathcal{K}_L(l,t)} x_j(t) + \frac{2\mu}{\pi} \sum_{j\in\mathcal{K}_G(l,t)} x_j(t). \end{aligned} \tag{2.35}$$

Combining (2.34) and (2.35), $x_l(t+1)$ remains bounded above and below by a convex combination of $x_j(t), j \in \mathcal{D}_l$, when

$$0 \leq \frac{2\mu}{\pi}|\mathcal{K}_L(l,t)| + \mu|\mathcal{K}_G(l,t)| \leq 1, \tag{2.36}$$

$$0 \leq \mu|\mathcal{K}_L(l,t)| + \frac{2\mu}{\pi}|\mathcal{K}_G(l,t)| \leq 1. \tag{2.37}$$

The L.H.S is trivially satisfied in both of the above equations. When μ is such that

$$\mu \leq \frac{1}{\frac{2}{\pi}\left(|\mathcal{K}_L(l,t)| + |\mathcal{K}_G(l,t)|\right)}, \tag{2.38}$$

$$\leq \frac{\pi}{2d_{\max}}, \tag{2.39}$$

we note that R.H.S is also satisfied for both of the above equations. Hence, with μ satisfying the above equation, $x_l(t+1)$ is bounded above and below by a convex combination of $x_j(t), j \in \mathcal{D}(l)$. Hence, if $x_j(0) \in [-\pi/4+\varepsilon, \pi/4-\varepsilon]$, for $j \in \mathcal{D}(l)$, so does its convex combination and the lemma follows. ■

Computing $\mathbf{f}'$

We now compute $\mathbf{f}'$. Let $\mathbf{D}(\mathbf{x})$ be a $K \times K$ diagonal matrix such that the kth element on its diagonal is $\cos(x_i - x_j)$, where i and j are those vertices that represent the edge described by the kth column of the incidence matrix, $\mathbf{C}$ (formally defined in Section 1.2).

Lemma 4: Let the derivative of the vector function, $\mathbf{f}(\mathbf{x})$, with respect to the vector $\mathbf{x}$ be denoted by the matrix, $\mathbf{f}'(\mathbf{x})$, i.e.,

$$\mathbf{f}'(\mathbf{x}) = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \left\{ \frac{\partial \mathbf{f}_i(\mathbf{x})}{\partial x_j} \right\}_{i,j=1,\ldots,N}, \tag{2.40}$$

then

$$\mathbf{f}'(\mathbf{x}) = \mathbf{I} - \mu \mathbf{C}\mathbf{D}(\mathbf{x})\mathbf{C}^T. \tag{2.41}$$

Proof: We have

$$\frac{\partial \mathbf{f}_i(\mathbf{x})}{\partial x_j} = \begin{cases} 1 - \mu \sum_{j \in \mathcal{K}(i)} \cos(x_i - x_j), & i = j, \\ \mu \cos(x_i - x_j), & i \neq j, (i,j) \in \mathcal{G}, \\ 0, & i \neq j, (i,j) \notin \mathcal{G}. \end{cases} \tag{2.42}$$

With the above, we note that

$$\left\{ \frac{\mathbf{f}'(\mathbf{x}) - \mathbf{I}}{-\mu} \right\}_{ij} = \begin{cases} \sum_{j \in \mathcal{K}(i)} \cos(x_i - x_j), & i = j, \\ -\cos(x_i - x_j) & i \neq j, (i,j) \in \mathcal{G}, \\ 0, & i \neq j, (i,j) \notin \mathcal{G}. \end{cases} \tag{2.43}$$

is a weighted Laplacian matrix with the corresponding weight for each edge, $(i,j) \in \mathcal{G}$, coming from the matrix $\mathbf{D}(\mathbf{x})$. ■

2.4.2 Error analysis

In this subsection, we perform the error analysis for the NLDAC iterations (2.23), (recall that the error is defined in (2.9)). To this aim, the following lemma provides an upper bound on the norm of $\mathbf{e}(t)$.

Lemma 5: For some $\theta(t) \in [0, 1]$, we have

$$\|\mathbf{e}(t+1)\| \leq \|\mathbf{I} - \mu\mathbf{C}\mathbf{D}(\theta(t)\mathbf{e}(t))\mathbf{C}^T - \mathbf{J}\|\|\mathbf{e}(t)\|. \tag{2.44}$$

Proof: From (2.20) and Lemma 4, we have

$$\|\mathbf{e}(t+1)\| \leq \|\mathbf{I} - \mu\mathbf{C}\mathbf{D}(\eta(\mathbf{t}))\mathbf{C}^T - \mathbf{J}\|\|\mathbf{e}(t)\|. \tag{2.45}$$

Recall (from Lemma 4) that the elements of $\mathbf{D}(\eta(t))$ are of the form $\cos(\eta_i(t) - \eta_j(t))$. We have

$$\begin{aligned} \eta_i(t) - \eta_j(t) &= \theta(t)x_i(t) + (1-\theta(t))\,x_{\text{avg}} - \big(\theta(t)x_j(t) + (1-\theta(t))\,x_{\text{avg}}\big)\,, \\ &= \theta(t)\,(x_i(t) - x_j(t))\,. \end{aligned} \tag{2.46}$$

Hence, we can write $\mathbf{D}(\eta(t))$ as $\mathbf{D}(\theta(t)\mathbf{e}(t))$ and (2.44) follows. ■

The above lemma establishes that if (iii) in Theorem 1 for the sine update holds, then $\lim_{t\to\infty}\|\mathbf{e}(t)\| = 0$. To show that $\|\mathbf{I} - \mu\mathbf{C}\mathbf{D}(\theta(t)\mathbf{e}(t))\mathbf{C}^T - \mathbf{J}\| < 1$, we perform the eigen-analysis of $\mathbf{I} - \mu\mathbf{C}\mathbf{D}(\theta(t)\mathbf{e}(t))\mathbf{C}^T - \mathbf{J}$ in the following.

Eigen-analysis of $\mathbf{f}'(\theta(t)\mathbf{e}(t)) - \mathbf{J}$

To proceed, we need the following notation. Let

$$\mathbf{L}_t = \mathbf{C}\mathbf{D}(\theta(t)\mathbf{e}(t))\mathbf{C}^T. \tag{2.47}$$

Define $\mathcal{L}$ as the set of all possible $\mathbf{L}_t$ when the algorithm is initialized with the initial conditions, $\mathbf{x}_0$, i.e.,

$$\mathcal{L} = \{\mathbf{L}_t \,|\mathbf{x}(0) = \mathbf{x}_0, \mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))\}\,. \tag{2.48}$$

Recall (from Section 1.2) that the weighted Laplacian, $\mathbf{L}_t$, is symmetric, positive-semidefinite, when the diagonal matrix, $\mathbf{D}(\theta(t)\mathbf{e}(t))$, is non-negative. Let

$$\mathbf{Q}_t = [\mathbf{q}_1(t), \mathbf{q}_2(t), \ldots, \mathbf{q}_N(t)] \tag{2.49}$$

be the matrix of N linearly independent eigenvectors of $\mathbf{L}_t$ with the corresponding eigenvalues denoted by $\lambda_i(\mathbf{L}_t), i = 1, \ldots, N$. Without loss of generality, we assume that[1]

$$0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N, \tag{2.50}$$

and $\mathbf{q}_1(t) = \mathbf{1}$ with the corresponding eigenvalue $\lambda_1(\mathbf{L}_t) = 0$. We further define

$$p_2 \triangleq \inf_{\mathcal{L}} \lambda_2(\mathbf{L}_t), \tag{2.51}$$

$$p_N \triangleq \sup_{\mathcal{L}} \lambda_N(\mathbf{L}_t). \tag{2.52}$$

Lemma 6: The eigenvectors of the matrix $\mathbf{I} - \mu\mathbf{L}_t - \mathbf{J}$ are the column vectors in the matrix $\mathbf{Q}_t$ and the corresponding eigenvalues are 0 and $1 - \mu\lambda_i(\mathbf{L}_t),\ i = 2, \ldots, N$.

Proof: The first eigenvector of the matrix $\mathbf{I} - \mu\mathbf{L}_t - \mathbf{J}$ is $\mathbf{q}_1(t) = \mathbf{1}$ with the eigenvalue 0. This can be shown as

$$(\mathbf{I} - \mu\mathbf{L}_t - \mathbf{J})\mathbf{q}_1 = \mathbf{1} - \mu\mathbf{L}_t\mathbf{1} - \mathbf{J}\mathbf{1} = \mathbf{0}. \tag{2.53}$$

That the rest of the eigenvectors, $\mathbf{q}_2(t), \ldots, \mathbf{q}_N(t)$, of $\mathbf{L}_t$, are also the eigenvectors of $\mathbf{I} - \mu\mathbf{L}_t - \mathbf{J}$ can be established by the fact that $\mathbf{J}$ is rank 1 with eigenvector $\mathbf{1}$ and the identity matrix can have any set of linearly independent vectors as its eigenvectors. ■

Using the above lemma, we now have the following result.

Lemma 7: Let (2.32) hold. Let the network communication graph be connected, i.e., $\lambda_2(\mathbf{L}) > 0$, then $p_2 > 0$, for $0 < \mu \leq \pi/2d_{\max}$.

Proof: Consider $\mathbf{L}_w$ as defined in (1.7) with $\mathbf{W} > 0$ (element-wise inequality, also recall from Section 1.2 that $\mathbf{W}$ is a $K \times K$ diagonal matrix where the kth element

[1]Note that $\mathbf{L}_t$ is symmetric, positive-semidefinite so its eigenvalues are non-negative reals.

on its diagonal (that represents the kth edge, $(i,j) \in \mathcal{G}$) is w_k). We have

$$\mathbf{z}^T \mathbf{L}_w \mathbf{z} = \sum_{(i,j)\in\mathcal{G}} w_k (z_i - z_j)^2, \tag{2.54}$$

for any $\mathbf{z} \in \mathbb{R}^N$. Since $w_k > 0, \forall (i,j) \in \mathcal{G}$, the quadratic form (2.54) is 0 if and only if $\mathbf{z} = c\mathbf{1}$, for any $c \in \mathbb{R}$. So $\mathbf{L}_w$ has only one eigenvalue of 0 with eigenvector $\mathbf{1}$. Hence, $\lambda_2(\mathbf{L}_w) > 0$. Also, note that $\lambda_2(\mathbf{L}_w)$ is a continuous function of w_k's [67]. So the infimum of $\lambda_2(\mathbf{L}_w)$ is attainable if the elements of $\mathbf{W}$ lie in a compact set. To this end, define

$$\mathcal{C} = \{\mathbf{W} \in \mathbb{R}^{K\times K} \mid w_{ii} \in [\cos(\pi/2 - 2\varepsilon), 1], w_{ij} = 0 (i \neq j)\}, \tag{2.55}$$

where $\varepsilon \in \mathbb{R}_{>0}$, and note that $\mathbf{D}(\theta(t)\mathbf{e}(t)) \in \mathcal{C}$ from Lemma 4. Since

$$\mathbf{L}_t = \mathbf{C}\mathbf{D}(\theta(t)\mathbf{e}(t))\mathbf{C}^T \tag{2.56}$$

and $\mathbf{D}(\theta(t)\mathbf{e}(t)) \in \mathcal{C}$, we note that

$$p_2 = \inf_{\mathcal{L}} \lambda_2(\mathbf{L}_t) \geq \inf_{\mathbf{W}\in\mathcal{C}} \lambda_2(\mathbf{L}_w). \tag{2.57}$$

We now use a contradiction argument to show $p_2 > 0$. Assume on the contrary that $p_2 = 0$. Then $\inf_{\mathbf{W}\in\mathcal{C}} \lambda_2(\mathbf{L}_w) = 0$ and there exists some $\mathbf{W} \in \mathcal{C}$ such that $\lambda_2(\mathbf{L}_w) = 0$. But, since $\mathbf{W} > 0$ and $\mathcal{G}$ is connected, i.e., $\lambda_2(\mathbf{L}) > 0$, for all $\mathbf{W} \in \mathcal{C}$, we have $\lambda_2(\mathbf{L}_w) > 0$ (see Section 1.2), which is a contradiction. Hence, $p_2 > 0$. ■

2.4.3 Main result

We now present the convergence of NLDAC in (2.23) in the following theorem.

Theorem 2: Let the NLDAC vector of initial conditions be denoted by $\mathbf{x}(0)$ such

that (2.32) holds. Let the network communication graph, $\mathcal{G}$, be connected and undirected, i.e., $\lambda_2(\mathbf{L}) > 0$. If μ is such that

$$0 < \mu < \frac{2}{p_N}, \tag{2.58}$$

then

$$\lim_{t\to\infty} \|\mathbf{e}(t)\| = 0. \tag{2.59}$$

Proof: From (2.52) and (2.58), we have for $i = 2, \ldots, N$

$$1 - \mu\lambda_i(\mathbf{L}_t) \geq 1 - \mu p_N > 1 - \frac{2}{p_N} p_N = -1. \tag{2.60}$$

From (2.51), we have for $i = 2, \ldots, N$

$$1 - \mu\lambda_i(\mathbf{L}_t) \quad \leq \quad 1 - \mu p_2 < 1, \tag{2.61}$$

from (2.58) the fact that $p_2 > 0$ from Lemma 7. Combining (2.60) and (2.61), we have

$$-1 < 1 - \mu\lambda_i(\mathbf{L}_t) < 1, \qquad i = 2, \ldots, N, \tag{2.62}$$

With the above equation, we have $|1 - \mu\lambda_i(\mathbf{L}_t)| < c < 1,\ i = 2, \ldots, N$, for some $c \in [0, 1)$. Thus, the error norm is given by

$$\|\mathbf{e}(t+1)\| \leq \max_{2\leq i\leq N} |1 - \mu\lambda_i(\mathbf{L}_t)| \|\mathbf{e}(t)\| < c\|\mathbf{e}(t)\|, \tag{2.63}$$

and (2.59) follows. ■

We further note that [59]

$$p_N \leq 2d_{\max}. \tag{2.64}$$

Combining (2.64) with (2.58), we have convergence for

$$0 < \mu < \frac{1}{d_{\max}} \leq \frac{2}{p_N}. \tag{2.65}$$

Remarks: We state relevant remarks.

(i) We explain our assumption in (2.32) with the help of Fig. 2.1. When the data, $x_l(0),\ \forall\ l$, lies in the interval $[-\pi/4+\varepsilon, \pi/4-\varepsilon]$, for $\varepsilon \in \mathbb{R}_{>0}$, then $x_l(t)$ also lies in the same interval $\forall\ l, t$, from Lemma 3. Thus, the state differences, $x_l(t) - x_j(t)$, for all $l, t, j \in \mathcal{D}(l)$ lie in the interval $[-\pi/2 + 2\varepsilon, \pi/2 - 2\varepsilon]$.

Hence, $\cos(x_l(t) - x_j(t)),\ \forall\ l, t$, must lie in the interval $[\cos(\pi/2 - 2\varepsilon), 1]$, which is strictly greater than 0 for $\varepsilon > 0$, as shown in Fig. 2.1. With $\cos(x_l(t) - x_j(t)) \in [\cos(\pi/2-2\varepsilon), 1],\ \forall\, t$, we note that $\mathbf{L}_t$ does not lose the sparsity (zero-one) pattern of $\mathbf{L}$ and hence, $\lambda_2(\mathbf{L}_t) > 0,\ \forall\ t$, if the underlying communication graph, $\mathcal{G}$, is connected, i.e., $\lambda_2(\mathbf{L}) > 0$.

Clearly, if (2.32) does not hold but the initial data has a known bound, we can introduce a frequency parameter, ζ, in the sine function such that $\cos\left(\zeta(x_l(t) - x_j(t))\right) \in [\cos(\pi/2 - 2\varepsilon), 1]$. Hence, the assumption in (2.32) does not lose generality.

(ii) Note that p_N may not be known or easily computable a priori. In such case, one may work with $0 < \mu < 1/d_{\max}$ as established in (2.65), which is readily determined.

(iii) Choosing μ away from the upper bound in (2.65), results into a divergence of the algorithm, as we also elaborate in the simulations.

(iv) The NLDAC algorithm can also be formulated with non-linear functions other than sinusoids, under the properties established in Theorem 1. Appropriate conditions for μ and the network initial conditions can be derived in these cases. For instance, one such family of functions are

$$f_l(x_{\mathcal{D}_l}(t)) = x_l(t) - \mu \sum_{j \in \mathcal{K}(l)} (x_l(t) - x_j(t))^p, \tag{2.66}$$

where $p > 0$ is odd. We may require an additional condition that the functions $f_l(\cdot)$ remain bounded for all time t by taking $f_l(\cdot)$ to be the minimum of some constant and (2.66). Clearly, $p = 1$ results into LDAC.

(v) Note that the upper bound on μ established in Lemma 3 is subsumed in the upper bound on μ used in (2.65).

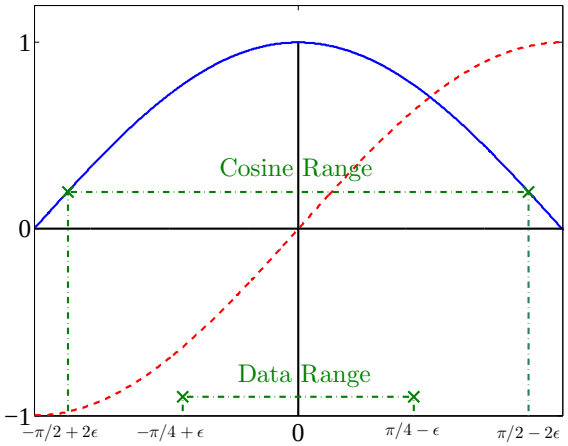


Figure 2.1: Figure corresponding to Remark (i).

2.5 Simulations

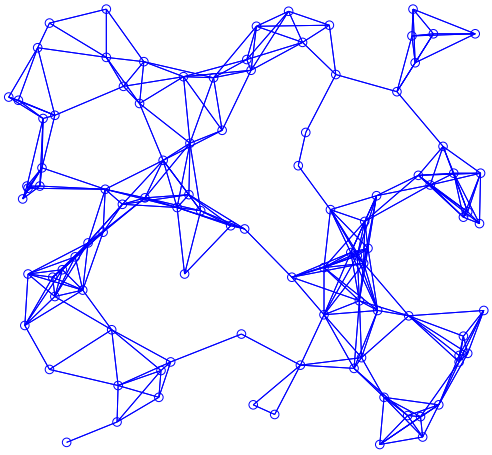
We consider a network of $N = 100$ nodes, shown in Fig. 2.2(a). We implement the conventional linear distributed average-consensus algorithm with optimal constant weights [17], i.e., we choose

$$\mu_{\text{LIN}}^{\text{OPT}} = \frac{2}{\lambda_2(\mathbf{L}) + \lambda_N(\mathbf{L})}, \tag{2.67}$$

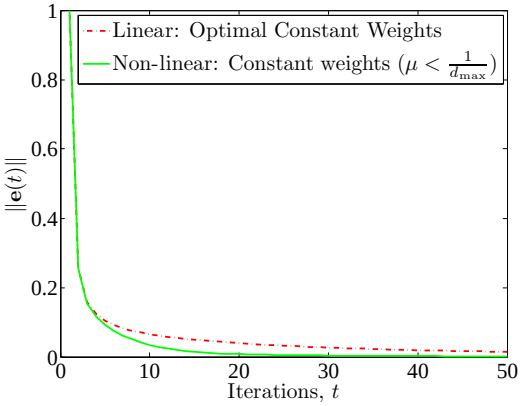
in (2.3). The error norm in this case is shown in Fig. 2.2(b) as a red dotted curve. To show the performance of the NLDAC algorithm (2.23), we choose the following values of μ

$$\mu = \left\{ \frac{0.99}{d_{\max}}, \frac{2}{d_{\max}}, \frac{1}{2d_{\max}}, \frac{1}{3d_{\max}} \right\} \tag{2.68}$$

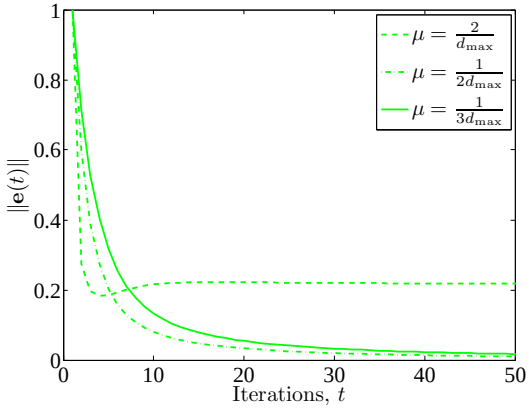
and show the error norm in Fig. 2.2(b) and Fig. 2.2(c). The simulations confirm the convergence of NLDAC with $0 < \mu < 1/d_{\max}$ and show the divergence of NLDAC when μ violates the bound. The convergence of NLDAC is faster than LDAC (with optimal constant weight) as shown in Fig. 2.2(b).



(a)



(b)



(c)

Figure 2.2: (a) An $N = 100$ node network. (b) Comparison of the NLDAC with LDAC using constant optimal edge weights. (c) The error norm of NLDAC for different values of μ.

2.6 Conclusions

In this chapter, we provide a non-linear distributed algorithm for average-consensus, which is a special case of HDC without anchors. We address the analysis and synthesis problems for this algorithm. In the analysis problem, we establish the conditions required on the non-linear updating functions for the NLDAC to reach average-consensus. We then study the synthesis problem by choosing a suitable updating function. In particular, we use a weighted sum of sinusoids as the consensus updating function. We prove appropriate bounds on the weights, domain, and frequency of this particular update such that the conditions established in the analysis are satisfied. With the choice of sine function, the convergence rate of the NLDAC algorithm now depends on the cosine of the state differences (as cosine is the derivative of the sine) and, thus, depends on the actual state values. Due to this dependence, the convergence rate has additional degrees of freedom as the convergence rate in LDAC only depends on the network connectivity. We provide simulations to assert the theoretical findings. The results can be extended to the any non-linear function as long as it fits the paradigm of Theorem 1.

Chapter 3

High dimensional consensus (Linear case)

In this chapter, we present high dimensional consensus (HDC) when the updating functions are linear. Linear HDC is a general class of linear distributed algorithms for large-scale networks that generalizes average-consensus and includes other interesting distributed algorithms, like sensor localization, leader-follower algorithms in multi-agent systems, or distributed Jacobi algorithm. In HDC[1], the network nodes are partitioned into 'anchors,' nodes whose states are fixed over the HDC iterations, and 'sensors,' nodes whose states are updated by the algorithm. The chapter considers two problems in this context: (i) *the analysis problem*; and, (ii) *the synthesis problem*. The analysis problem establishes the conditions under which HDC converges, the limiting state to which it converges, and what is its convergence rate.

The *synthesis* or design problem *learns* the weights or parameters of the HDC so that the algorithm converges to a desired pre-specified state. This generalizes the well-known problem of designing the weights in average-consensus. We pose *learning* as a constrained non-convex optimization problem that we cast in the framework of multi-objective optimization (MOP) and to which we apply Pareto optimality. We derive the solution to the learning problem by proving relevant properties satisfied by the MOP solutions and by the Pareto front. Finally, the chapter shows how the MOP

[1]In this chapter, HDC implies the linear case.

approach leads to interesting tradeoffs (speed of convergence versus performance) arising in resource constrained networks. Simulation studies illustrate our approach for a leader-follower architecture in multi-agent systems.
Parts of this chapter have been presented in [53, 45, 43, 56].

3.1 Introduction

In this chapter, we present the *high dimensional consensus* (HDC) with linear updates. HDC provides a unified framework for the analysis and design of linear distributed algorithms for large-scale networks; examples include distributed Jacobi algorithm [58], average-consensus [13, 17, 20], distributed sensor localization [41], distributed matrix inversion [48], and leader-follower algorithms [53, 45]. These applications arise in many resource constrained large-scale networks, e.g., sensor networks, teams of robots, and also in cyber-physical systems like the smart grid in electric power systems. We view these systems as a collection of nodes interacting over a sparse communication graph. The nodes, in general, have strict communication and computation constraints so that only local communication and low-order computation is feasible at each node.

In HDC, the network nodes are partitioned into anchors and sensors. Anchors do not update their state over the HDC iterations, while the sensors iteratively update their states by a linear, possibly convex, combination of their neighboring nodes' states. The weights of this linear combination are the parameters of the HDC. For example, in distributed sensor localization, the state at each node is its current position estimate and the HDC parameters are the collection of barycentric coordinates. Anchors are the nodes that know their precise locations and the remaining non-anchor nodes (sensors) do not know their locations. Using HDC, each sensor updates its state, i.e., its location, in a distributed fashion, which converges to the exact sensor locations, see Chapter 4 for details.

We pose the following two problems in the context of HDC:

Analysis: (Forward problem) Given the HDC parameters (i.e., the coefficients of the linear combination) and the underlying connectivity graph, determine: (i) when

does the HDC converge; (ii) to what state does the HDC converge; and (iii) what is the convergence rate.

Learning: (Synthesis problem) Given the desired state to which HDC should converge and the sparsity of the underlying communication graph, *learn* the HDC parameters, so that HDC converges to the desired state. A widely studied example in this context is the average consensus problem where every sensor is an anchor in itself and the goal is to converge to the average of the initial (anchors') states. Here the question of interest is whether there exists a distributed iterative algorithm, i.e., is it possible to design the HDC parameters, such that the sensor states converge to the desired average. This problem of designing link weights for average-consensus has seen much recent activity, see for example [17, 68].

In this chapter, we take a step further in this direction, i.e., we would like the sensors to converge to specific linear combinations of the anchors' states, which may be different for different sensors and not necessarily a simple average. The goal then is to design the HDC parameters so that the state of the HDC converges to the desired linear combinations. Due to the network sparsity constraints, it may not be possible for HDC to converge exactly to the desired state. Our formulation leads to interesting tradeoffs between the speed of convergence and the error in the limiting HDC state, i.e., the final state and the desired state. Clearly, the learning problem is inverse (synthesis) to the analysis (forward) problem.

We formulate learning as a constrained non-convex optimization problem that we cast in the framework of a multi-objective optimization problem (MOP) [69]. We prove that the optimal solution to the HDC learning problem is a Pareto optimal (P.O.) solution, which we extract from the Pareto front (locus of all P.O. solutions.) We exploit the structure of our problem to prove smoothness, convexity, strict decreasing monotonicity, and differentiability properties of the Pareto front. Although, in general, it is computationally infeasible to determine the Pareto front as it requires extensive iterative procedures [69]; with the help of the established properties and the structure of our problem, we derive an efficient procedure to generate the Pareto front, and find the solution to the learning problem. This solution is found by a rather expressive geometric argument. We illustrate our approach by finding the optimal

HDC parameters for a leader-follower problem in multi-agent systems.

We now describe the rest of the chapter. In Section 3.2, we provide the problem formulation. We discuss the forward problem (analysis of HDC) in Section 3.3. An example of the forward problem is the distributed Jacobi algorithm that we present in Section 3.4. Section 3.5 discusses the robustness of HDC in random environments. We then present the synthesis problem (learning in large-scale networks) in Sections 3.6–3.8. We present the simulations in Section 3.9 and finally, Section 3.10 concludes the chapter.

3.2 Problem formulation

Consider a sensor network with N nodes communicating over a network described by a directed graph, $\mathcal{G} = (\Theta, \mathbf{A})$. Let $\mathbf{u}_k \in \mathbb{R}^{1\times m}$ be the state associated to the kth anchor, and let $\mathbf{x}_l \in \mathbb{R}^{1\times m}$ be the state associated to the lth sensor. We are interested in studying linear iterative algorithms of the form[2]

$$\mathbf{u}_k(t+1) = \mathbf{u}_k(t), \qquad k \in \kappa, \tag{3.1}$$

$$\mathbf{x}_l(t+1) = \sum_{j\in\mathcal{K}_\Omega(l)\cup\{l\}} p_{lj}\mathbf{x}_j(t) + \sum_{k\in\mathcal{K}_\kappa(l)} b_{lk}\mathbf{u}_k(0), \tag{3.2}$$

for $l \in \Omega$, where: $t \geq 0$ is the discrete-time iteration index; and p_{lj}'s and b_{lk}'s are the state updating coefficients. We assume that the updating coefficients are constant over the components of the m-dimensional state[3], $\mathbf{x}_l(t)$. We term distributed linear iterative algorithms of the form (3.1)–(3.2) as *High Dimensional Consensus (HDC)* algorithms [45, 53].

[2]In average-consensus [17], $m = 1$, and there are no anchors. The algorithm takes the form of (3.2) without the second term in the sum.

[3]An example when the updating coefficients are constant over the state dimensions is the distributed sensor localization problem when the updating coefficients are the barycentric coordinates that are the same over all dimensions (vertical and horizontal coordinates) of a node's state (location estimate). If these coefficients are not constant then there are two HDC algorithms running side-by-side, one with parameters $\mathbf{P}_1, \mathbf{B}_1$ and the other with parameters $\mathbf{P}_2, \mathbf{B}_2$. In this case, the two HDC algorithms can be dealt with separately.

For the purpose of analysis, we write the HDC (3.1)–(3.2) in matrix form. Define

$$\mathbf{U}(t) = \left[\mathbf{u}_1^T(t), \ldots, \mathbf{u}_K^T(t)\right]^T, \qquad \mathbf{X}(t) = \left[\mathbf{x}_{K+1}^T(t), \ldots, \mathbf{x}_N^T(t)\right]^T, \tag{3.3}$$

$$\mathbf{P} = \{p_{lj}\} \in \mathbb{R}^{M\times M}, \qquad \mathbf{B} = \{b_{lk}\} \in \mathbb{R}^{M\times K}. \tag{3.4}$$

Note that $\mathbf{U}(t) \in \mathbb{R}^{K\times m}$ and $\mathbf{X}(t) \in \mathbb{R}^{M\times m}$. With the above notation, we write (3.1)–(3.2) concisely as

$$\begin{bmatrix} \mathbf{U}(t+1) \\ \mathbf{X}(t+1) \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B} & \mathbf{P} \end{bmatrix} \begin{bmatrix} \mathbf{U}(t) \\ \mathbf{X}(t) \end{bmatrix}, \tag{3.5}$$

$$\triangleq \mathbf{C}(t+1) = \boldsymbol{\Upsilon}\mathbf{C}(t). \tag{3.6}$$

Note that the graph, $\mathcal{G}^{\boldsymbol{\Upsilon}}$, associated to the $N \times N$ iteration matrix, $\boldsymbol{\Upsilon}$, must be a subgraph of $\mathcal{G}$. In other words, the sparsity of $\boldsymbol{\Upsilon}$ is dictated by the sparsity of the underlying sensor network. In the iteration matrix, $\boldsymbol{\Upsilon}$: the submatrix, $\mathbf{P}$, collects the updating coefficients of the M sensors with respect to the M sensors; and the submatrix, $\mathbf{B}$, collects the updating coefficients of the M sensors with respect to the K anchors. From (3.5), the matrix form of the HDC in (3.2) is

$$\mathbf{X}(t+1) = \mathbf{P}\mathbf{X}(t) + \mathbf{B}\mathbf{U}(0), \qquad t \geq 0. \tag{3.7}$$

We now formally state the analysis and synthesis problems.

Analysis: (Forward problem) Given an N-node sensor network with a communication graph, $\mathcal{G}$, the matrices $\mathbf{B}$, and $\mathbf{P}$, and the network initial conditions, $\mathbf{X}(0)$ and $\mathbf{U}(0)$; what are the conditions under which the HDC converges? what is the convergence rate of the HDC? if the HDC converges, what is the limiting state of the network?

Learning: (Synthesis problem) Given an N-node sensor network with a communication graph, $\mathcal{G}$, and an $M \times K$ weight matrix, $\mathbf{W}$, learn the matrices $\mathbf{P}$ and $\mathbf{B}$

in (3.7) such that the HDC converges to[4]

$$\lim_{t\to\infty} \mathbf{X}(t+1) = \mathbf{W}\mathbf{U}(0), \tag{3.8}$$

for every $\mathbf{U}(0) \in \mathbb{R}^{K\times m}$, where $\mathbf{W}$ is an arbitrary $M \times K$ matrix; if multiple solutions exist, we are interested in finding a solution that leads to fastest convergence. Furthermore, our solution leads to interesting performance-speed trade-offs as a solution with faster convergence may be desirable if a certain error can be tolerated in the limiting state.

3.3 Analysis problem: High dimensional consensus

As discussed in Section 3.2, the HDC algorithm is implemented as (3.1)–(3.2), and its matrix representation is given by (3.6). We divide the analysis of the HDC in the following two cases: (A) no anchors; and (B) with anchors. We analyze these two cases separately and provide, briefly, their practical applications.

3.3.1 No anchors: $\mathbf{B} = \mathbf{0}$

In this case, the HDC reduces to

$$\begin{aligned} \mathbf{X}(t+1) &= \mathbf{P}\mathbf{X}(t), \\ &= \mathbf{P}^{t+1}\mathbf{X}(0). \end{aligned} \tag{3.9}$$

An important problem covered by this case is average-consensus. As well known, when

$$\rho(\mathbf{P}) = 1, \tag{3.10}$$

[4]Note that the learning problem is the standard coefficient-design problem in average-consensus [17], where $\mathbf{W} = \mathbf{1}\mathbf{1}^T/N$ and $\mathbf{U}(0) = \mathbf{X}(0)$.

and under some minimal assumptions on $\mathbf{P}$ and on the network connectivity, (3.9) converges to the average of the initial sensors' states. For more precise and general statements in this regard, see for instance, [17, 20]. Average-consensus, thus, is a special case of the HDC, when $\mathbf{B} = \mathbf{0}$ and $\rho(\mathbf{P}) = 1$. This problem has been studied in great detail, a detailed set of references is provided in [45].

The rest of this paper deals entirely with the case $\rho(\mathbf{P}) < 1$ and the term HDC subsumes the $\rho(\mathbf{P}) < 1$ case, unless explicitly noted. Note that, when $\mathbf{B} = \mathbf{0}$, the HDC (with $\rho(\mathbf{P}) < 1$) leads to $\mathbf{X}_\infty = \mathbf{0}$, which is not interesting.

3.3.2 With anchors: $\mathbf{B} \neq \mathbf{0}$

This extends the average-consensus to "higher dimensions" (as will be explained in Section 3.3.3.) The following lemma 8 establishes: (i) the conditions under which the HDC converges; (ii) the limiting state of the network; and (iii) the rate of convergence of the HDC.

Lemma 8: Let $\mathbf{B} \neq 0$ and $\mathbf{U}(0) \notin \mathcal{N}(\mathbf{B})$, where $\mathcal{N}(\mathbf{B})$ is the null space of $\mathbf{B}$. If

$$\rho(\mathbf{P}) < 1, \tag{3.11}$$

then the limiting state of the sensors, $\mathbf{X}_\infty$, is given by

$$\mathbf{X}_\infty \triangleq \lim_{t\to\infty} \mathbf{X}(t+1) = (\mathbf{I} - \mathbf{P})^{-1}\mathbf{B}\mathbf{U}(0), \tag{3.12}$$

and the error, $\mathbf{E}(t) = \mathbf{X}(t) - \mathbf{X}_\infty$, decays exponentially to $\mathbf{0}$ with exponent $\ln(\rho(\mathbf{P}))$, i.e.,

$$\limsup_{t\to\infty} \frac{1}{t}\ln\|\mathbf{E}(t)\| \;\leq\; \ln(\rho(\mathbf{P})). \tag{3.13}$$

Proof: From (3.7), we note that

$$\mathbf{X}(t+1) \;=\; \mathbf{P}^{t+1}\mathbf{X}(0) + \sum_{k=0}^{t} \mathbf{P}^k\mathbf{B}\mathbf{U}(0), \tag{3.14}$$

$$\Rightarrow \mathbf{X}_\infty \;=\; \lim_{t\to\infty} \mathbf{P}^{t+1}\mathbf{X}(0) + \lim_{t\to\infty} \sum_{k=0}^{t} \mathbf{P}^k\mathbf{B}\mathbf{U}(0), \tag{3.15}$$

and (3.12) follows from (3.11) and Lemma 25 in Appendix A.1. The error, $\mathbf{E}(t)$, is given by

$$\begin{aligned}\mathbf{E}(t) &= \mathbf{X}(t) - (\mathbf{I}-\mathbf{P})^{-1}\mathbf{B}\mathbf{U}(0), \\ &= \mathbf{P}^t\mathbf{X}(0) + \sum_{k=0}^{t-1}\mathbf{P}^k\mathbf{B}\mathbf{U}(0) - \sum_{k=0}^{\infty}\mathbf{P}^k\mathbf{B}\mathbf{U}(0), \\ &= \mathbf{P}^t\left[\mathbf{X}(0) - \sum_{k=0}^{\infty}\mathbf{P}^k\mathbf{B}\mathbf{U}(0)\right].\end{aligned}$$

To go from the first equation to the second, we recall (3.11) and use (A.2) from Lemma 25 in Appendix A.1. Let $\mathbf{R} = \mathbf{X}(0) - \sum_{k=0}^{\infty}\mathbf{P}^k\mathbf{B}\mathbf{U}(0)$. To establish the convergence rate of $\|\mathbf{E}(t)\|$, we have

$$\begin{aligned}\frac{1}{t}\ln\|\mathbf{E}(t)\| &= \frac{1}{t}\ln\|\mathbf{P}^t\mathbf{R}\|, \\ &\leq \frac{1}{t}\ln\left(\|\mathbf{P}^t\|\|\mathbf{R}\|\right), \\ &\leq \ln\|\mathbf{P}^t\|^{1/t} + \frac{1}{t}\ln\|\mathbf{R}\|. \end{aligned} \tag{3.16}$$

Now, letting $t \to \infty$ on both sides, we get

$$\begin{aligned}\limsup_{t\to\infty}\frac{1}{t}\ln\|\mathbf{E}(t)\| &\leq \limsup_{t\to\infty}\left(\ln\|\mathbf{P}^t\|^{1/t} + \frac{1}{t}\ln\|\mathbf{R}\|\right), &(3.17)\\ &= \ln\lim_{t\to\infty}\|\mathbf{P}^t\|^{1/t}, &(3.18)\\ &= \ln\left(\rho(\mathbf{P})\right). &(3.19)\end{aligned}$$

and (3.13) follows. The interchange of lim and ln is permissible because of the continuity of ln and the last step follows from (1.10). ■

The above lemma shows that the limiting state of the sensors, $\mathbf{X}_\infty$, is independent of the sensors' initial conditions and is given by (3.12), for any $\mathbf{X}(0) \in \mathbb{R}^{M\times m}$. It is also straightforward to show that if $\rho(\mathbf{P}) \geq 1$, then the HDC algorithm (3.7) diverges for all $\mathbf{U}(0) \notin \mathcal{N}(\mathbf{B})$, where $\mathcal{N}(\mathbf{B})$ is the null space of $\mathbf{B}$. Clearly, the case $\mathbf{U}(0) \in \mathcal{N}(\mathbf{B})$ is not interesting as it leads to $\mathbf{X}_\infty = \mathbf{0}$.

Remarks: When we restrict the HDC linear state update (3.1) at each sensor to be convex, the resulting iteration matrix, $\mathbf{\Upsilon}$, in (3.6) is stochastic. Since we assumed that $\mathbf{B} \neq \mathbf{0}$ (recall that the matrix $\mathbf{B}$ governs the anchor to sensor communication graph), at least one anchor is connected to at least one sensor. Hence, we can show that the matrix $\mathbf{P}$ (that governs the sensor to sensor communication) is strictly-substochastic (at least one row sum is strictly less than 1). Under the additional condition that $\mathbf{P}$ is irreducible, which translates to a strongly-connected sensor to sensor communication graph, the matrix $\mathbf{P}$ becomes uniformly substochastic (an irreducible and strictly substochastic matrix). For uniformly substochastic matrices, we have $\rho(\mathbf{P}) < 1$ [70], as assumed in (3.11). Hence, the limiting state (3.12) and the convergence rate (3.13) derived in Lemma 8 can be also derived under the assumptions of a strongly-connected sensor to sensor communication graph and convex updates at the sensors, since they imply (3.11), i.e., $\rho(\mathbf{P}) < 1$.

Furthermore, when we restrict HDC to convex updates, this setup has an absorbing Markov chain [71] interpretation. In this interpretation, the anchors represent the absorbing states, the sensors represent the transient states, and the coefficient b_{lk}s and p_{lj}s represent the state-transition probabilities. Clearly, this interpretation holds only under convex update assumptions, since the coefficients should be non-negative and should sum to 1 at each sensor. Under the assumption that each transient state has a path to any absorbing state, the Markov chain almost surely resides in the set of absorbing states, asymptotically. This is obvious from (3.12), where each sensor state converges to a linear combination of the anchor states. This parallel has been studied in detail in our work on distributed localization in Chapter 4, see also [41]. In this localization problem, the coefficients b_{lk}s and p_{lj}s are the barycentric coordinates that are non-negative convex at each sensor, hence, the Markov chain theory applies.

Lemma 8 is more general as it does not assume non-negativity and convexity on the state updating coefficients. Hence, the cases when the state updates are not necessarily convex are also included in the lemma (for instance, in the case of the Jacobi algorithm that we discuss in Section 3.4) and the lemma statement does not lose generality. Clearly, additional conditions on the state updates may imply (3.11), as we discussed above. Another interesting comment here is that Lemma 8 does not impose

any restriction on the communication graph. Although, certain restrictions may be required to ensure (3.11). Further parallels, where random walks and Markov chain based arguments are used, can also be established with discrete Dirichlet problems [72], discrete Green's function [73], and discretized partial difference equations with boundary conditions.

3.3.3 Consensus subspace

We now define the consensus subspace as follows.

Definition 1 (Consensus subspace): Given the matrices, $\mathbf{B} \in \mathbb{R}^{M\times K}$ and $\mathbf{P} \in \mathbb{R}^{M\times M}$, such that $\rho(\mathbf{P}) < 1$, the consensus subspace, Ξ, is defined as

$$\Xi = \{\mathbf{X}_\infty \mid \mathbf{X}_\infty = (\mathbf{I}-\mathbf{P})^{-1}\mathbf{B}\mathbf{U}(0)\}, \tag{3.20}$$

for all $\mathbf{U}(0) \in \mathbb{R}^{K\times m}$.

The dimension of the consensus subspace, Ξ, is established in the following theorem.

Theorem 3: If $K < M$ and $\rho(\mathbf{P}) < 1$, then the dimension of the consensus subspace, Ξ, is

$$\dim(\Xi) = m\ \mathrm{rank}(\mathbf{B}) \leq mK. \tag{3.21}$$

Proof: The proof follows from Lemma 8, and Lemma 26 in Appendix A.1. ■

Now, we formally define the dimension of the HDC.

Definition 2 (Dimension): The dimension of the HDC algorithm is the dimension of the consensus subspace, Ξ, normalized by m, i.e.,

$$\dim(\mathrm{HDC}) = \frac{\dim(\Xi)}{m} = \mathrm{rank}(\mathbf{B}). \tag{3.22}$$

This definition is natural because the HDC is a decoupled algorithm, i.e., HDC corresponds to m parallel algorithms, one for each column of $\mathbf{X}(t)$. So, the number of columns, m, in $\mathbf{X}(t)$ is factored out in the definition of dim(HDC). Each column of $\mathbf{X}(t)$ lies in a subspace that is spanned by exactly rank($\mathbf{B}$) basis vectors that can be at most the number of anchors K.

3.3.4 Practical applications of the HDC

Several interesting applications can be framed in the context of HDC, as for example the sensor localization problem, banded matrix inversion problem, or the distributed Jacobi algorithm.

- *Distributed Jacobi algorithm*: Distributed Jacobi algorithm solves a linear system of equations in a distributed fashion. It turns out that the distributed Jacobi algorithm falls into the category of HDC algorithms by choosing the HDC parameters $\mathbf{P}, \mathbf{B}$, appropriately. We describe this in the next section.

- *Sensor localization in m-dimensional Euclidean spaces,* $\mathbb{R}^m$: In Chapter 4, we present an HDC algorithm specialized to sensor localization in $\mathbb{R}^m$, where an arbitrary number of sensors (with unknown locations) iteratively learn their exact locations given that they lie in the convex hull of a minimal number, $m+1$ of anchors (that know their precise locations). Each sensor updates its m-dimensional state, $\mathbf{x}_l(t) \in \mathbb{R}^m$ (location estimate at time t), as a convex linear combination of the states of its $m+1$ carefully chosen neighbors. The weights of this linear combination (p_{lj}'s and b_{lk}'s) are the barycentric coordinates (*computed locally by local distances* and Cayley-Menger determinants).

- *Distributed banded matrix inversion*: In Chapter 5, we specialize HDC to solve a matrix inversion problem when the sub-matrices in its band are distributed among several nodes. This distributed inversion algorithm leads to distributed Kalman filters in sensor networks using Gauss-Markov approximations by noting that the inverse of a Gauss-Markov covariance matrix is banded. The distributed Kalman filter is the subject of Chapter 6.

In all of the above problems, note that although the HDC parameters, $\mathbf{P}, \mathbf{B}$, are known (i.e., readily determined from local information), the weight matrix, $\mathbf{W}$, is not given. The applications where the weight matrix, $\mathbf{W}$, is known and the HDC parameters, $\mathbf{P}, \mathbf{B}$ are to be designed fall into the learning problem, as discussed in Section 3.6.1.

3.4 Distributed Jacobi algorithm

In this section, we show that the well-known Jacobi algorithm [58] is a special case of HDC. The Jacobi algorithm solves a linear system of equations in a distributed and iterative fashion. In particular, we are interested in solving

$$\mathbf{DX} = \mathbf{U}, \tag{3.23}$$

where $\mathbf{X} \in \mathbb{R}^{M\times m}$ denotes the unknown state of a sensor network that follows (3.23). The system matrix, $\mathbf{D} \in \mathbb{R}^{M\times M}$, is strict diagonally dominant and sparse, and we further have $\mathcal{G}^{\mathbf{D}} \subseteq \mathcal{G}$. The anchors have the fixed state, $\mathbf{U} \in \mathbb{R}^{M\times m}$. In this case, we note that the number of anchors, K, is equal to the number of sensors, M, i.e., $K = M$. Hence the total number of nodes in the network is $N = 2M$. Linear systems of equations appear naturally in sensor networks, for example, power flow equations in power systems monitored by sensors, [74], or time synchronization algorithms in sensor networks, [75].

3.4.1 Design of the iteration matrix, $\boldsymbol{\Upsilon}$

Let $\mathbf{M} = \text{diag}(\mathbf{D})$. We make the following design choice:

$$\mathbf{B} = \mathbf{M}^{-1}, \tag{3.24}$$

$$\mathbf{P} = \mathbf{M}^{-1}(\mathbf{M} - \mathbf{D}). \tag{3.25}$$

With the above design choice the iteration matrix, $\boldsymbol{\Upsilon}$, is given by

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \mathbf{I}_M & \mathbf{0} \\ \mathbf{M}^{-1} & \mathbf{M}^{-1}(\mathbf{M} - \mathbf{D}) \end{bmatrix}. \tag{3.26}$$

At each sensor, l, we note that the lth row, $\mathbf{p}_l$, of the matrix $\mathbf{P}$ in (3.25) is a function of the lth row, $\mathbf{d}_l$, of the system matrix, $\mathbf{D}$ and the lth diagonal element, m_{ll}, of the diagonal matrix, $\mathbf{M}^{-1}$. With (3.25), the sparsity pattern of $\mathbf{P}$ is the same as the sparsity pattern of the system matrix, $\mathbf{D}$, since $\mathbf{M}$ is a diagonal matrix. Hence,

the underlying sensor communication graph, $\mathcal{G}$, comes directly form the graph, $\mathcal{G}^{\mathbf{D}}$ associated to the system matrix, $\mathbf{D}$. The non-zero elements of the system matrix, $\mathbf{D}$, thus establish the inter-connections among the sensors. The reason for assuming a sparse system matrix, $\mathbf{D}$, is apparent here since a full system matrix, $\mathbf{D}$, will result into an all-to-all communication graph among the sensors. Each sensor is, further, directly connected to exactly one anchor. The anchors in this case can be considered as dummy anchors with their states being available at each sensor they are connected to in the graph, $\mathcal{G}^{\boldsymbol{\Upsilon}}$, associated to $\boldsymbol{\Upsilon}$.

3.4.2 Convergence

To establish the convergence, we give the following lemma.

Lemma 9: Let $\mathbf{D}$ be a strict diagonally dominant matrix. Let the matrix $\mathbf{P}$ be given by (3.25). Then

$$\rho(\mathbf{P}) < 1. \tag{3.27}$$

Proof: The proof is straightforward and relies on Gershgorin's circle theorem [76]. ■

With Lemma 9, we note that a distributed iterative algorithm with $\mathbf{B}$ and $\mathbf{P}$ as given in (3.24) and (3.25), respectively, converges to

$$\begin{aligned} \lim_{t\to\infty} \mathbf{X}(t+1) &= (\mathbf{I}_M - \mathbf{P})^{-1}\mathbf{B}\mathbf{U}, \\ &= (\mathbf{I}_M - \mathbf{M}^{-1}(\mathbf{M}-\mathbf{D}))^{-1}\mathbf{M}^{-1}\mathbf{U}, \\ &= \mathbf{D}^{-1}\mathbf{U}. \end{aligned} \tag{3.28}$$

Hence, the sensors asymptotically reach the solution of (3.23) in a distributed fashion.

3.4.3 Remarks

HDC with the matrices $\mathbf{B}$ and $\mathbf{P}$ given in (3.24)–(3.25) is the matrix extension of the well-known Jacobi algorithm [58]. HDC (3.1)–(3.2), when implemented with the given matrices $\mathbf{B}$ and $\mathbf{P}$, thus, gives a sensor network framework for solving the linear system of equations (3.23). The Jacobi algorithm can be further specialized to sparse

symmetric, positive definite matrices, $\mathbf{D}$, see [58]. When these matrices have the special structure of being banded, we present a distributed inversion algorithm by specializing HDC with a non-linear collapse operator in 5. The collapse operator adds a collapse step to the general model in (3.7) that exploits structural properties of banded matrices to make the algorithm computationally more efficient.

If $\mathbf{D}$ is not diagonally dominant (or positive-definite), we cannot guarantee (3.27). Furthermore, when the sparsity of $\mathbf{D}$ does not correspond to the underlying communication graph, $\mathcal{G}$, the convergence to the exact solution cannot be established. In such cases, the learning problem can be employed as we show in Section 3.6.1.

3.5 Robustness of the HDC

Robustness is key in the context of HDC, when the information exchange is subject to data packet drops, communication noise, and imprecise knowledge of system parameters. These random phenomena can be modeled as follows:

- **Data packet drops:** We assume that each data packet sent over the communication link $(l \leftarrow j)$ is received at node l with a non-zero probability, q_{lj}, at each iteration, where $0 < q_{lj} \leq 1$. We model this by a binary random variable, $e_{lj}(t)$, such that

$$e_{lj}(t) = \begin{cases} 1, & \text{w.p. } q_{lj}, \\ 0, & \text{w.p. } 1 - q_{lj}. \end{cases} \tag{3.29}$$

  where $e_{lj}(t) = 1$ indicates that the data packet has successfully arrived $(l \leftarrow j)$ at time t, and $e_{lj}(t) = 0$ indicates a data packet drop.

- **Communication noise:** We model the communication noise as additive channel noise, i.e., at the t-th iteration, sensor l receives only a corrupt version, $\mathbf{y}_{lj}(t)$, of node j's state, $\mathbf{x}_j(t)$, given by

$$\mathbf{y}_{lj}(t) = \mathbf{x}_j(t) + \mathbf{v}_{lj}(t), \tag{3.30}$$

  where each component of the noise, $\mathbf{v}_{lj}(t)$, belongs to a family of independent

zero-mean random variables with finite second moments.

- **Small perturbation of system matrices:** We may also assume that because of imprecise measurements the matrices $\mathbf{P}$ and $\mathbf{B}$ are known up to a certain error computed at each iteration to be $\widehat{\mathbf{P}}(t) = \{\widehat{p}_{lj}\}$ and $\widehat{\mathbf{B}}(t) = \{\widehat{p}_{lj}\}$ given by

$$\widehat{\mathbf{P}}(t) = \mathbf{P} + \mathbf{S_P} + \widetilde{\mathbf{S}}_{\mathbf{P}}(t), \qquad \widehat{\mathbf{B}}(t) = \mathbf{B} + \mathbf{S_B} + \widetilde{\mathbf{S}}_{\mathbf{B}}(t), \tag{3.31}$$

where $\mathbf{S_P}$ and $\mathbf{S_B}$ are mean measurement errors, and $\{\widetilde{\mathbf{S}}_{\mathbf{P}}(t)\}_{t\geq 0}$ and $\{\widetilde{\mathbf{S}}_{\mathbf{B}}(t)\}_{t\geq 0}$ are independent sequence of random matrices with zero-mean and finite second moments. Here, we assume a small signal perturbation such that

$$\rho(\mathbf{P} + \mathbf{S_P}) < 1. \tag{3.32}$$

3.5.1 HDC in random environments

To account for the above random phenomena, we present the following modified HDC algorithm that writes the iteration for the kth component of the m-dimensional sensor state, $\mathbf{x}_l(t)$ at sensor l and time t.

$$\begin{aligned} x_{l,k}(t+1) &= (1-\alpha(t))\, x_{l,k}(t) + \alpha(t) \left[\sum_{j\in\mathcal{K}_\kappa(l)} \frac{e_{lj}(t)\widehat{b}_{lj}(t)}{q_{lj}} \left(u_j^k + v_{lj}^k(t)\right) \right] \\ &+ \alpha(t) \left[\sum_{j\in\mathcal{K}_\Omega(l)} \frac{e_{lj}(t)\widehat{p}_{lj}(t)}{q_{lj}} \left(x_{j,k}(t) + v_{lj}^k(t)\right) \right], \end{aligned} \tag{3.33}$$

for $l \in \Omega,\ 1 \leq k \leq m$, where $\alpha(t)$ is a weight sequence (elaborated in the next theorem). The following theorem establishes the convergence of the above algorithm.

Theorem 4: Under the random phenomena described in Section 3.5, the distributed algorithm given by (3.33) along with (3.1) converges to

$$\lim_{t\to\infty} \mathbf{x}^k(t+1) = (\mathbf{I} - \mathbf{P} - \mathbf{S_P})^{-1}\mathbf{B} + \mathbf{S_B}\mathbf{u}^k, \qquad 1 \leq k \leq m, \tag{3.34}$$

where the superscript k denotes the kth column of the appropriate matrix, under the following persistence conditions on the weight sequence, $\alpha(t)$:

$$\alpha(t) > 0, \tag{3.35}$$

$$\sum_{t\geq 0} \alpha(t) = \infty, \tag{3.36}$$

$$\sum_{t\geq 0} \alpha^2(t) < \infty. \tag{3.37}$$

Proof: The proof relies on the theory of stochastic recursive algorithms and is provided in [41]. ■

Clearly, when $\mathbf{S_P} = \mathbf{S_B} = \mathbf{0}$, the HDC converges to the exact solution of the deterministic case, as established in Lemma 8, otherwise a steady state error expression can be formulated [41]. Here, we note that as $t \to \infty$, $1 - \alpha(t) \to 1$ and the algorithm in (3.33) relies heavily on the past state estimate at each sensor as opposed to the information arriving from the neighbors. Hence, the algorithm, as time goes on, suppresses the inclusion of noise that comes due to the random phenomena. The persistence conditions on $\alpha(t)$, commonly assumed in the adaptive control and adaptive signal processing literature, assumes that the weights decay to zero, but not too fast, allowing optimal mixing time for the information before suppressing the noise affects.

3.6 Synthesis problem: Learning in large-scale networks

As we briefly mentioned before, the synthesis problem learns the parameter matrices ($\mathbf{B}$ and $\mathbf{P}$) of the HDC when the weight matrix, $\mathbf{W}$ in (3.8), is given. Before we proceed with the discussion on the synthesis problem, we briefly sketch some relevant practical applications.

3.6.1 Practical applications of the synthesis problem

We present the following practical applications of the synthesis problem.

- *Generalization of weight design in average-consensus*: A standard problem in average-consensus is to design the parameters (the matrices $\mathbf{B}$ and $\mathbf{P}$ in our notation) to optimize the convergence rate of consensus, subject to the graph constraints [17, 68]. In particular, average-consensus requires $\mathbf{X}_\infty = \mathbf{WU}(0)$, where $\mathbf{W} = \mathbf{1}\mathbf{1}^T/M$ and $\mathbf{U}(0) = \mathbf{X}(0)$. The synthesis problem in HDC broadly generalizes the coefficient design in consensus–it designs the parameters ($\mathbf{B}$ and $\mathbf{P}$) when the weight matrix, $\mathbf{W}$, is arbitrary.

- *Solving arbitrary linear system of equations*: As provided in Section 3.4, the HDC algorithm can be specialized to solve linear systems of equations of the form $\mathbf{DX} = \mathbf{U}$, when $\mathbf{D}$ is a positive-definite matrix and $\mathcal{G}^{\mathbf{D}} \subseteq \mathcal{G}$. If $\mathbf{D}$ is not diagonally dominant (or positive-definite), we cannot guarantee (3.27). Furthermore, when the sparsity of $\mathbf{D}$ does not correspond to the underlying communication graph, $\mathcal{G}$, the convergence to the exact solution cannot be established. In such cases, one may be interested in an approximate distributed solution that minimizes the norm, $\|\mathbf{X}_\infty - \mathbf{WU}\|$, where the weight matrix, $\mathbf{W}$, is $\mathbf{D}^{-1}$. The HDC synthesis problem, in this case, provides the matrices $\mathbf{B}$ and $\mathbf{P}$ that give the optimal distributed solution minimizing $\|\mathbf{X}_\infty - \mathbf{WU}\|$ under the network sparsity, $\mathcal{G}$, and the desired convergence criteria.

  In many practical settings, the network parameters, $\mathbf{D}$, remain constant but the anchors' state, $\mathbf{U}$, change with time, i.e., we have $\mathbf{DX}_k = \mathbf{U}_k$, where k denotes the time-scale of the underlying phenomenon. This is the case, for instance, in solving power-flow equations in electrical grids [74] where the network parameters, $\mathbf{D}$, are a function of the line impedances (that remain constant) and the system inputs (power injections), $\mathbf{U}_k$, change because of the system loading. Hence, once the HDC parameters, $\mathbf{B}$ and $\mathbf{P}$, are computed off-line, the HDC algorithm may be implemented in a distributed way using the same $\mathbf{B}$ and $\mathbf{P}$ when the system loading changes.

- *Coordination in multi-agent systems (Leader-follower algorithms)* [53]: In the leader-follower algorithm, the state of each sensor converges to a particular anchor state (or a linear combination). For example, consider a scalar case

$(m = 1)$, with one anchor, i.e., $K = 1$, and let $u_1 \in \mathbb{R}$ be the (one-dimensional[5]) anchor state. For the sensors to converge to the state of the anchor, we require

$$\mathbf{x}_\infty \triangleq \lim_{t\to\infty} \mathbf{x}(t) = \mathbf{1}u_1, \tag{3.38}$$

where $\mathbf{x}(t) = [x_1(t), \ldots, x_M(t)]^T$ collects the sensor states at time t and $\mathbf{1}$ is the M-dimensional column-vector of 1s. Clearly, in this case, the weight matrix, $\mathbf{W}$, is given to be $\mathbf{1}$. With the following arguments, we can design the HDC parameters. Comparing (3.38) with (3.12), we have

$$(\mathbf{I} - \mathbf{P})^{-1}\mathbf{B} = \mathbf{1} \qquad \Rightarrow \qquad \mathbf{B} + \mathbf{P}\mathbf{1} = \mathbf{1}, \tag{3.39}$$

which results in the following design requirements for the elements of the matrices $\mathbf{P}$ and $\mathbf{B}$:

$$b_l + \sum_{j=1}^{M} p_{lj} = 1, \qquad \forall\ l, \tag{3.40}$$

along with (3.11). Specific choices for these elements are considered in [45].

In multi-agent systems, the leader-follower problem mentioned above is generalized to more than one leader, $K > 1$, see for instance [77] and references therein. In such cases, an important problem is to design an optimal control strategy, where the state of the leaders remains constant and the state of the followers is updated using a local control law [78, 40]. Clearly, the synthesis problem fits in this scenario, where the followers intend to learn the parameters of distinct leaders using local updates and under specified network constraints.

Further applications include networks of heterogenous robots, for example, consider a network of humans, ground robots, and aerial robots. In many disaster/military applications, humans are unable to access the region of interest where the robots are deployed [79, 80, 81]. The humans guide the robots to carry out several tasks and act as anchors whereas the robots act as followers (sensors). The network has communication constraints as each robot may not

[5]Similar arguments hold for multi-dimensional anchor state.

be able to directly access the human who is controlling it. We will revisit this architecture (with multiple leaders) in Section 3.9.

We now address the learning problem. Recall that for convergence of the HDC, we require the spectral radius constraint (3.11), and the matrices, $\mathbf{B}$ and $\mathbf{P}$, to follow the underlying communication network, $\mathcal{G}$. In general, due to the spectral norm constraint and the sparseness (network) constraints, equation (3.8) may not be met with equality. So, it is natural to relax the learning problem. Using Lemma 8 and (3.8), we restate the learning problem as follows.

Consider $\varepsilon \in [0, 1)$. Given an N-node sensor network with a communication graph, $\mathcal{G}$, and an $M \times K$ weight matrix, $\mathbf{W}$, solve the optimization problem:

$$\inf_{\mathbf{B},\mathbf{P}} \| \left(\mathbf{I} - \mathbf{P}\right)^{-1} \mathbf{B} \;-\; \mathbf{W} \|, \tag{3.41}$$

$$\text{subject to:} \quad \text{Spectral radius constraint,} \quad \rho\left(\mathbf{P}\right) \;\leq\; \varepsilon, \tag{3.42}$$

$$\text{Sparsity constraint,} \quad \mathcal{G}^{\boldsymbol{\Upsilon}} \;\subseteq\; \mathcal{G}, \tag{3.43}$$

for some induced matrix norm $\| \cdot \|$. By Lemma 8, if $\rho\left(\mathbf{P}\right) \leq \varepsilon$, the convergence is exponential with exponent less than or equal to $\ln(\varepsilon)$. Thus, we may ask, given a pre-specified convergence rate, ε, what is the minimum error between the limiting state, $\lim_{t\to\infty} \mathbf{X}(t)$, and the desired state, $\mathbf{WU}(0)$. Formulating the problem in this way naturally lends itself to a trade-off between the performance and the convergence rate.

In some cases, it may happen that the learning problem has an exact solution in the sense that there exist $\mathbf{B}, \mathbf{P}$, satisfying (3.42) and (3.43) such that the objective in (3.41) is zero. In case of multiple such solutions, we seek the one which corresponds to the fastest convergence. We may still formulate a performance versus convergence rate trade-off, if faster convergence is desired.

The learning problem stated in (3.41)–(3.43) is, in general, practically infeasible to solve because both (3.41) and (3.42) are non-convex in $\mathbf{P}$. We reformulate it into a more tractable framework next.

3.6.2 Revisiting the spectral radius constraint

We work with a convex relaxation of the spectral radius constraint. Recall that the spectral radius can be expressed as (1.10). However, direct use of (1.10) as a constraint is, in general, not computationally feasible. Hence, instead of using the spectral radius constraint (3.42) we use a matrix induced norm constraint by realizing that

$$\rho(\mathbf{P}) \leq \|\mathbf{P}\|, \tag{3.44}$$

for any matrix induced norm. The induced norm constraint, thus, becomes

$$\|\mathbf{P}\| \leq \varepsilon. \tag{3.45}$$

Clearly, (3.44) implies that any upper bound on $\|\mathbf{P}\|$ is also an upper bound on $\rho(\mathbf{P})$.

3.6.3 Revisiting the sparsity constraint

In this subsection, we rewrite the sparsity constraint (3.43) as a linear constraint in the design parameters, $\mathbf{B}$ and $\mathbf{P}$. The sparsity constraint ensures that the structure of the underlying communication network, $\mathcal{G}$, is not violated. To this aim, we introduce an auxiliary variable, $\mathbf{F}$, defined as

$$\mathbf{F} \triangleq [\mathbf{B} \mid \mathbf{P}] \in \mathbb{R}^{M \times N}. \tag{3.46}$$

This auxiliary variable, $\mathbf{F}$, combines the matrices $\mathbf{B}$ and $\mathbf{P}$ as they correspond to the adjacency matrix, $\mathbf{A}^{\mathcal{G}}$, of the given communication graph, $\mathcal{G}$, see the comments after (3.6).

To translate the sparsity constraint into linear constraints on $\mathbf{F}$ (and, thus, on $\mathbf{B}$ and $\mathbf{P}$), we employ a two-step procedure: (i) First, we identify the elements in the adjacency matrix, $\mathbf{A}^{\mathcal{G}}$, that are zero; these elements correspond to the pairs of nodes in the network where we do not have a communication link. (ii) We then force the elements of $\mathbf{F} = [\mathbf{B} \mid \mathbf{P}]$ corresponding to zeros in the adjacency matrix, $\mathbf{A}^{\mathcal{G}}$, to be zero. Mathematically, (i) and (ii) can be described as follows.

(i) Let the lower $M \times N$ submatrix of the $N \times N$ adjacency matrix, $\mathbf{A} = \{a_{lj}\}$ (this lower part corresponds to $\mathbf{F} = [\mathbf{P} \mid \mathbf{B}]$ as can be noted from (3.6)), be denoted by $\underline{\mathbf{A}}$, i.e,

$$\underline{\mathbf{A}} = \{\underline{a}_{ij}\} = \{a_{lj}\}, \tag{3.47}$$

for $l = K+1, \ldots, N$, $j = 1, \ldots N$, and $i = 1, \ldots, M$. Let χ contain all pairs (i, j) for which $\underline{a}_{ij} = 0$.

(ii) Let $\{\mathbf{e}_i\}_{i=1,\ldots,M}$ be a family of $1 \times M$ row-vectors such that $\mathbf{e}_i$ has a 1 as the ith element and zeros everywhere else. Similarly, let $\{\mathbf{e}^j\}_{j=1,\ldots,N}$ be a family of $N \times 1$, column-vectors such that $\mathbf{e}^j$ has a 1 as the jth element and zeros everywhere else. With this notation, the ij-th element, f_{ij}, of $\mathbf{F}$ can be written as $f_{ij} = \mathbf{e}_i \mathbf{F} \mathbf{e}^j$. The sparsity constraint (3.43) is explicitly given by

$$\mathbf{e}_i \mathbf{F} \mathbf{e}^j = 0, \qquad \forall\ (i, j) \in \chi. \tag{3.48}$$

3.6.4 Feasible solutions

Consider $\varepsilon \in [0, 1)$. We now define a set of matrices, $\mathcal{F}_{\leq \varepsilon} \subseteq \mathbb{R}^{M \times N}$, that follow both the induced norm constraint (3.45) and the sparsity constraint (3.48) of the learning problem. The set of feasible solutions is given by

$$\mathcal{F}_{\leq \varepsilon} = \left\{ \mathbf{F} = [\mathbf{B} \mid \mathbf{P}] \,\middle|\, \mathbf{e}_i \mathbf{F} \mathbf{e}^j = 0,\ \|\mathbf{F}\mathbf{T}\| \leq \varepsilon \right\}, \tag{3.49}$$

$\forall\ (i, j) \in \chi$, where

$$\mathbf{T} \ \triangleq \ \begin{bmatrix} \mathbf{0}_{K \times M} \\ \mathbf{I}_M \end{bmatrix} \in \mathbb{R}^{N \times M}. \tag{3.50}$$

With the matrix $\mathbf{T}$ defined as above, we note that $\mathbf{P} = \mathbf{F}\mathbf{T}$.

Lemma 10: The set of feasible solutions, $\mathcal{F}_{\leq \varepsilon}$, is convex.

Proof: Let $\mathbf{F}_1, \mathbf{F}_2 \in \mathcal{F}_{\leq \varepsilon}$, then

$$\mathbf{e}_i \mathbf{F}_1 \mathbf{e}^j = 0,\ \forall\ (i, j) \in \chi, \qquad \mathbf{e}_i \mathbf{F}_2 \mathbf{e}^j = 0,\ \forall\ (i, j) \in \chi. \tag{3.51}$$

For any $0 \leq \mu \leq 1$, and $\forall\ (i,j) \in \chi$,

$$\mathbf{e}_i \left(\mu \mathbf{F}_1 + (1-\mu)\mathbf{F}_2\right) \mathbf{e}^j = \mu \mathbf{e}_i \mathbf{F}_1 \mathbf{e}^j + (1-\mu)\mathbf{e}_i \mathbf{F}_2 \mathbf{e}^j = 0.$$

Similarly,

$$\| \left(\mu \mathbf{F}_1 + (1-\mu)\mathbf{F}_2\right) \mathbf{T}\| \leq \mu \|\mathbf{F}_1 \mathbf{T}\| + (1-\mu)\|\mathbf{F}_2 \mathbf{T}\| \leq \mu\varepsilon + (1-\mu)\varepsilon = \varepsilon.$$

The first inequality uses the triangle inequality for matrix induced norms and the second uses the fact that, for $i \in \{1,2\}$, $\mathbf{F}_i \in \mathcal{F}$ and $\|\mathbf{F}_i \mathbf{T}\| \leq \varepsilon$.

Thus, $\mathbf{F}_1, \mathbf{F}_2 \in \mathcal{F}_{\leq \varepsilon} \Rightarrow \mu \mathbf{F}_1 + (1-\mu)\mathbf{F}_2 \in \mathcal{F}_{\leq \varepsilon}$. Hence, $\mathcal{F}_{\leq \varepsilon}$ is convex. ■

Similarly, we note that the sets, $\mathcal{F}_{<\varepsilon}$ and $\mathcal{F}_{<1}$, are also convex.

3.6.5 Learning Problem: An upper bound on the objective

In this section, we simplify the objective function (3.41) and give a tractable upper bound. We have the following proposition.

Proposition 1: Let $\|\mathbf{P}\| < 1$, then

$$\left\|(\mathbf{I}-\mathbf{P})^{-1}\mathbf{B} - \mathbf{W}\right\| \leq \frac{1}{1-\|\mathbf{P}\|} \|\mathbf{B} + \mathbf{P}\mathbf{W} - \mathbf{W}\|. \tag{3.52}$$

Proof: We manipulate (3.41) to obtain successively.

$$\begin{aligned}
\left\|(\mathbf{I}-\mathbf{P})^{-1}\mathbf{B} - \mathbf{W}\right\| &\leq \left\|(\mathbf{I}-\mathbf{P})^{-1}\right\| \|(\mathbf{B} - (\mathbf{I}-\mathbf{P})\mathbf{W})\|, \\
&= \left\|\sum_k \mathbf{P}^k\right\| \|(\mathbf{B} - (\mathbf{I}-\mathbf{P})\mathbf{W})\|, \\
&\leq \sum_k \|\mathbf{P}\|^k \|(\mathbf{B} - (\mathbf{I}-\mathbf{P})\mathbf{W})\|, \\
&= \frac{1}{1-\|\mathbf{P}\|} \|\mathbf{B} + \mathbf{P}\mathbf{W} - \mathbf{W}\|.
\end{aligned} \tag{3.53}$$

To go from the first equation to the second, we use (A.2) from Lemma 25 in Appendix A.1. Lemma 25 is applicable here since (3.44) and given the norm constraint $\|\mathbf{P}\| < 1$ imply $\rho(\mathbf{P}) < 1$. The last step is the sum of a geometric series

which converges given $\|\mathbf{P}\| < 1$. ■

We now define the utility function, $u(\mathbf{B}, \mathbf{P})$, that we minimize instead of minimizing $\| (\mathbf{I} - \mathbf{P})^{-1} \mathbf{B} - \mathbf{W}\|$. This is valid because $u(\mathbf{B}, \mathbf{P})$ is an upper bound on (3.41) and hence minimizing the upper bound leads to a performance guarantee. The utility function is defined as:

$$u(\mathbf{B}, \mathbf{P}) \triangleq \frac{1}{1 - \|\mathbf{P}\|} \|\mathbf{B} + \mathbf{PW} - \mathbf{W}\| . \tag{3.54}$$

With the help of the previous development, we now formally present the *Learning Problem.*

Learning Problem: Given $\varepsilon \in [0, 1)$, an N-node sensor network with a sparse communication graph, $\mathcal{G}$, and a possibly full $M \times K$ weight matrix, $\mathbf{W}$, design the matrices $\mathbf{B}$ and $\mathbf{P}$ (in (3.7)) that minimize (3.54), i.e., solve the optimization problem

$$\inf_{[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq \varepsilon}} \quad u(\mathbf{B}, \mathbf{P}). \tag{3.55}$$

Note that the induced norm constraint (3.45) and the sparsity constraint (3.48) are implicit in (3.55), as they appear in the set of feasible solutions, $\mathcal{F}_{\leq \varepsilon}$. Furthermore, the optimization problem in (3.55) is equivalent to the following problem.

$$\inf_{[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq \varepsilon} \cap \{\|\mathbf{B}\| \leq b\}} \quad u(\mathbf{B}, \mathbf{P}), \tag{3.56}$$

where $b > 0$ is a sufficiently large number. Since (3.56) involves the infimum of a continuous function, $u(\mathbf{B}, \mathbf{P})$, over a compact set, $\mathcal{F}_{\varepsilon} \cap \{\|\mathbf{B}\| \leq b\}$, the infimum is attainable and, hence, in the subsequent development, we replace the infimum in (3.55) by a minimum.

We view the $\min u(\mathbf{B}, \mathbf{P})$ as the minimization of its two factors, $1/(1 - \|\mathbf{P}\|)$ and $\|\mathbf{B} + \mathbf{PW} - \mathbf{W}\|$. In general, we need $\|\mathbf{P}\| \to 0$ to minimize the first factor, $1/(1 - \|\mathbf{P}\|)$, and $\|\mathbf{P}\| \to 1$ to minimize the second factor, $\|\mathbf{B} + \mathbf{PW} - \mathbf{W}\|$ (we explicitly prove this statement later.) Hence, these two objectives are conflicting. Since, the minimization of the *non-convex* utility function, $u(\mathbf{B}, \mathbf{P})$, contains minimizing two coupled *convex* objective functions, $\|\mathbf{P}\|$ and $\|\mathbf{B} + \mathbf{PW} - \mathbf{W}\|$, we formulate

this minimization as a multi-objective optimization problem (MOP). In the MOP, we consider separately minimizing these two convex functions. We then couple the MOP solutions using the utility function.

3.6.6 Solution to the Learning Problem: MOP formulation

To solve the *Learning Problem* for every $\varepsilon \in [0, 1)$, we cast it in the context of a multi-objective optimization problem (MOP). In the MOP formulation, we treat $\|\mathbf{B} + \mathbf{PW} - \mathbf{W}\|$ as the first objective function, f_1, and $\|\mathbf{P}\|$ as the second objective function, f_2. The objective vector, $\mathbf{f}(\mathbf{B}, \mathbf{P})$, is

$$\mathbf{f}(\mathbf{B}, \mathbf{P}) \triangleq \begin{bmatrix} f_1(\mathbf{B}, \mathbf{P}) \\ f_2(\mathbf{B}, \mathbf{P}) \end{bmatrix} = \begin{bmatrix} \|\mathbf{B} + \mathbf{PW} - \mathbf{W}\| \\ \|\mathbf{P}\| \end{bmatrix}. \tag{3.57}$$

The multi-objective optimization problem (MOP) is given by

$$\min_{[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq 1}} \mathbf{f}(\mathbf{B}, \mathbf{P}), \tag{3.58}$$

where[6]

$$\mathcal{F}_{\leq 1} = \{\mathbf{F} = [\mathbf{B} \mid \mathbf{P}] \ : \ \mathbf{e}_i \mathbf{F} \mathbf{e}^j = 0, \|\mathbf{FT}\| \leq 1\}, \tag{3.59}$$

$\forall\ (i, j) \in \chi$.

We now define Pareto-optimal solutions of an MOP.

Definition 3: [Pareto optimal solutions] A solution, $[\mathbf{B}^* \mid \mathbf{P}^*]$, is said to be a Pareto optimal (or non-inferior) solution of a MOP, if there exists no other feasible $[\mathbf{B} \mid \mathbf{P}]$ (i.e., there is no $[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq 1}$) such that $\mathbf{f}(\mathbf{B}, \mathbf{P}) \leq \mathbf{f}(\mathbf{B}^*, \mathbf{P}^*)$, meaning that $f_k(\mathbf{B}, \mathbf{P}) \leq f_k(\mathbf{B}^*, \mathbf{P}^*),\ \forall\ k$, with strict inequality for at least one k.

Before providing one of the main results of this chapter on the equivalence of MOP and the *Learning Problem*, we set the following notation. We define

$$\varepsilon_{\text{exact}} = \min\{\|\mathbf{P}\| \mid (\mathbf{I} - \mathbf{P})^{-1}\mathbf{B} = \mathbf{W}, [\mathbf{B} \ |\mathbf{P}] \in \mathcal{F}_{<1}\}, \tag{3.60}$$

[6]Although the *Learning Problem* is valid only when $\|\mathbf{P}\| < 1$, the MOP is defined at $\|\mathbf{P}\| = 1$. Hence, we consider $\|\mathbf{P}\| \leq 1$ when we seek the MOP solutions.

where the minimum of an empty set is taken to be $+\infty$. In other words, $\varepsilon_{\text{exact}}$ is the minimum value of $f_2 = \|\mathbf{P}\|$ at which we may achieve an exact solution[7] of the *Learning Problem*. A necessary condition for the existence of an exact solution is studied in Appendix A.2. If the exact solution is infeasible ($\notin \mathcal{F}_{<1}$), then $\varepsilon_{\text{exact}} = \min\{\varnothing\}$, which we defined to be $+\infty$. We let

$$\mathcal{E} = [0, 1) \cap [0, \varepsilon_{\text{exact}}]. \tag{3.61}$$

The *Learning Problem* is interesting if $\varepsilon \in \mathcal{E}$. We now study the relationship between the MOP and the *Learning Problem* (3.55). Recall Definition 3 of Pareto-optimal solutions of an MOP. We have the following theorem.

Theorem 5: Let $\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon$, be an optimal solution of the *Learning Problem*, where $\varepsilon \in \mathcal{E}$. Then, $\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon$ is a Pareto-optimal solution of the MOP (3.58).

The proof relies on analytical properties of the MOP (discussed in Section 3.7) and is deferred until Section 3.7.3. We discuss here the consequences of Theorem 5. Theorem 5 says that the optimal solutions to the *Learning Problem* can be obtained from the Pareto-optimal solutions of the MOP. In particular, it suffices to generate the Pareto front (collection of Pareto-optimal solutions of the MOP) for the MOP and seek the solutions to the *Learning Problem* from the Pareto front. The subsequent Section is devoted to constructing the Pareto front for the MOP and studying the properties of the Pareto front.

3.7 Multi-objective optimization: Pareto front

We consider the MOP (3.58) as an ε-constraint problem, denoted by $P_k(\varepsilon)$ [69]. For a two-objective optimization, $n = 2$, we denote the ε-constraint problem as $P_1(\varepsilon_2)$

[7] An exact solution is given by $[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}$ such that $(\mathbf{I} - \mathbf{P})^{-1}\mathbf{B} = \mathbf{W}$ or when the infimum in (3.41) is attainable and is 0.

or $P_2(\varepsilon_1)$, where $P_1(\varepsilon_2)$ is given by[8]

$$\min_{[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq 1}} f_1(\mathbf{B}, \mathbf{P}) \qquad \text{s.t.} \qquad f_2(\mathbf{B}, \mathbf{P}) \leq \varepsilon_2. \tag{3.62}$$

and $P_2(\varepsilon_1)$ is given by

$$\min_{[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq 1}} f_2(\mathbf{B}, \mathbf{P}) \qquad \text{s.t.} \qquad f_1(\mathbf{B}, \mathbf{P}) \leq \varepsilon_1. \tag{3.63}$$

In both $P_1(\varepsilon_2)$ and $P_2(\varepsilon_1)$, we are minimizing a real-valued convex function, subject to a constraint on the real-valued convex function over a convex feasible set. Hence, either optimization can be solved using a convex program [82]. We can now write $\varepsilon_{\text{exact}}$ in terms of $P_2(\varepsilon_1)$ as

$$\varepsilon_{\text{exact}} = \begin{cases} P_2(0), & \text{if there exists a solution to } P_2(0), \\ +\infty, & \text{otherwise.} \end{cases} \tag{3.64}$$

Using $P_1(\varepsilon_2)$, we find the Pareto-optimal set of solutions of the MOP. We explore this in Section 3.7.1. The collection of the values of the functions, f_1 and f_2, at the Pareto-optimal solutions forms the Pareto front (formally defined in Section 3.7.2). We explore properties of the Pareto front, in the context of our learning problem, in Section 3.7.2. These properties will be useful in addressing the minimization in (3.55) for solving the *Learning Problem*.

3.7.1 Pareto-optimal solutions

In general, obtaining Pareto-optimal solutions requires iteratively solving ε-constraint problems [69], but we will show that the optimization problem, $P_1(\varepsilon_2)$, results directly into a Pareto-optimal solution. To do this, we provide Lemma 11 and its Corollary 1 in the following. Based on these, we then state the Pareto-optimality of the solutions of $P_1(\varepsilon_2)$ in Theorem 6.

[8] All the infima can be replaced by minima in a similar way as justified in Section 3.6.5. Further note that, for technical convenience, we use $\mathcal{F}_{\leq 1}$ and not $\mathcal{F}_{<1}$, which is permissible because the MOP objectives are defined for all values of $\|\mathbf{P}\|$.

Lemma 11: Let

$$[\mathbf{B}_0 \mid \mathbf{P}_0] \;=\; \text{argmin}_{[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq 1}} P_1(\varepsilon_0). \tag{3.65}$$

If $\varepsilon_0 \in \mathcal{E}$, then the minimum of the optimization, $P_1(\varepsilon_0)$, is attained at ε_0, i.e.,

$$f_2(\mathbf{B}_0, \mathbf{P}_0) = \varepsilon_0. \tag{3.66}$$

Proof: Let the minimum value of the objective, f_1, be denoted by δ_0, i.e.,

$$\delta_0 = f_1(\mathbf{B}_0, \mathbf{P}_0). \tag{3.67}$$

We prove this by contradiction. Assume, on the contrary, that $\|\mathbf{P}_0\| = \varepsilon' < \varepsilon_0$. Define

$$\alpha_0 \triangleq \frac{1-\varepsilon_0}{1-\varepsilon'}. \tag{3.68}$$

Since, $\varepsilon' < \varepsilon_0 < 1$, we have $0 < \alpha_0 < 1$. For $\alpha_0 \leq \alpha < 1$, we define another pair, $\mathbf{B}_1, \mathbf{P}_1$, as

$$\mathbf{B}_1 \triangleq \alpha \mathbf{B}_0, \qquad\qquad \mathbf{P}_1 \triangleq (1-\alpha)\mathbf{I} + \alpha \mathbf{P}_0. \tag{3.69}$$

Clearly, this choice is feasible, as it does not violate the sparsity constraints of the problem and further lies in the constraint of the optimization in (3.65), since

$$\|\mathbf{P}_1\| \;\leq\; (1-\alpha) + \alpha\varepsilon' \leq 1 - \alpha(1-\varepsilon') \leq 1 - \alpha_0(1-\varepsilon') = \varepsilon_0. \tag{3.70}$$

With the matrices $\mathbf{B}_1, \mathbf{P}_1$ in (3.69), we have the following value, δ_1, of the objective function, f_1,

$$\begin{aligned} \delta_1 &= \|\mathbf{B}_1 + \mathbf{P}_1\mathbf{W} - \mathbf{W}\|, \\ &= \|\alpha\mathbf{B}_0 + ((1-\alpha)\,\mathbf{I} + \alpha\mathbf{P}_0)\,\mathbf{W} - \mathbf{W}\|\,, \\ &= \|\alpha\mathbf{B}_0 + \alpha\mathbf{P}_0\mathbf{W} - \alpha\mathbf{W}\|\,, \\ &= \alpha f_1(\mathbf{B}_0, \mathbf{P}_0) = \alpha\delta_0. \end{aligned} \tag{3.71}$$

Since, $\alpha < 1$ and non-negative, we have $\delta_1 < \delta_0$. This shows that the new pair, $\mathbf{B}_1, \mathbf{P}_1$, constructed from the pair, $\mathbf{B}_0, \mathbf{P}_0$, results in a lower value of the objective function. Hence, the pair, $\mathbf{B}_0, \mathbf{P}_0$, with $\|\mathbf{P}_0\| = \varepsilon' < \varepsilon_0$ is not optimal, which is a contradiction. Hence, $f_2(\mathbf{B}_0, \mathbf{P}_0) = \varepsilon_0$,. ■

Lemma 11 shows that if a pair of matrices, $\mathbf{B}_0, \mathbf{P}_0$, solves the optimization problem $P_1(\varepsilon_0)$ with $\varepsilon_0 \in \mathcal{E}$, then the pair of matrices, $\mathbf{B}_0, \mathbf{P}_0$, meets the constraint on f_2 with equality, i.e., $f_2(\mathbf{B}_0, \mathbf{P}_0) = \varepsilon_0$. The following corollary follows from Lemma 11.

Corollary 1: Let $\varepsilon_0 \in \mathcal{E}$, and

$$[\mathbf{B}_0 \mid \mathbf{P}_0] = \text{argmin}_{[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq 1}} P_1(\varepsilon_0), \tag{3.72}$$

$$\delta_0 = f_1(\mathbf{B}_0, \mathbf{P}_0). \tag{3.73}$$

Then,

$$\delta_0 < \delta_\varepsilon, \tag{3.74}$$

for any $\varepsilon < \varepsilon_0$, where

$$[\mathbf{B}_\varepsilon \mid \mathbf{P}_\varepsilon] = \text{argmin}_{[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq 1}} P_1(\varepsilon), \tag{3.75}$$

$$\delta_\varepsilon = f_1(\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon). \tag{3.76}$$

Proof: Clearly, from Lemma 11 there does not exist any $\varepsilon < \varepsilon_0$ that results in a lower value of the objective function, f_1. ■

The above lemma shows that the optimal value of f_1 obtained by solving $P_1(\varepsilon)$ is strictly greater than the optimal value of f_1 obtained by solving $P_1(\varepsilon_0)$ for any $\varepsilon < \varepsilon_0$.

The following theorem now establishes the Pareto-optimality of the solutions of $P_1(\varepsilon)$.

Theorem 6: If $\varepsilon_0 \in \mathcal{E}$, then the solution $\mathbf{B}_0, \mathbf{P}_0$, of the optimization problem, $P_1(\varepsilon_0)$, is Pareto optimal.

Proof: Since, $\mathbf{B}_0, \mathbf{P}_0$ solves the optimization problem, $P_1(\varepsilon_0)$, we have $\|\mathbf{P}_0\| = \varepsilon_0$, from Lemma 11. Assume, on the contrary that $\mathbf{B}_0, \mathbf{P}_0$, are not Pareto-optimal. Then,

by definition of Pareto-optimality, there exists a feasible $\mathbf{B}, \mathbf{P}$, with

$$f_1(\mathbf{B}, \mathbf{P}) \leq f_1(\mathbf{B}_0, \mathbf{P}_0), \tag{3.77}$$

$$f_2(\mathbf{B}, \mathbf{P}) \leq f_2(\mathbf{B}_0, \mathbf{P}_0), \tag{3.78}$$

with strict inequality in at least one of the above equations. Clearly, if $f_2(\mathbf{B}, \mathbf{P}) < f_2(\mathbf{B}_0, \mathbf{P}_0)$, then $\|\mathbf{P}\| < \varepsilon_0$ and $\mathbf{B}, \mathbf{P}$, are feasible for $P_1(\varepsilon_0)$. By Corollary 1, we have $f_1(\mathbf{B}, \mathbf{P}) > f_1(\mathbf{B}_0, \mathbf{P}_0)$. Hence, $f_2(\mathbf{B}, \mathbf{P}) < f_2(\mathbf{B}_0, \mathbf{P}_0)$ is not possible.

On the other hand, if $f_1(\mathbf{B}, \mathbf{P}) < f_1(\mathbf{B}_0, \mathbf{P}_0)$ then we contradict the fact that $\mathbf{B}_0, \mathbf{P}_0$, are optimal for $P_1(\varepsilon_0)$, since by (3.78), $\mathbf{B}, \mathbf{P}$, is also feasible for $P_1(\varepsilon_0)$.

Thus, in either way, we have a contradiction and $\mathbf{B}_0, \mathbf{P}_0$ are Pareto-optimal. ■

3.7.2 Properties of the Pareto front

In this section, we formally introduce the Pareto front and explore some of its properties in the context of the *Learning Problem*. The Pareto front and their properties are essential for the minimization of the utility function, $u(\mathbf{B}, \mathbf{P})$ over $\mathcal{F}_{\leq \varepsilon}$ (3.55), as introduced in Section 3.6.5.

Let $\overline{\mathcal{E}}$ denote the closure of $\mathcal{E}$. The Pareto front is defined as follows.

Definition 4: [Pareto front] Consider $\varepsilon \in \overline{\mathcal{E}}$. Let $\mathbf{B}_\varepsilon$, $\mathbf{P}_\varepsilon$, be a solution of $P_1(\varepsilon)$ then[9] $\varepsilon = f_2(\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon)$. Let $\delta = f_1(\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon)$. The collection of all such (ε, δ) is defined as the Pareto front.

For a given $\varepsilon \in \overline{\mathcal{E}}$, define $\delta(\varepsilon)$ to be the minimum of f_1 in $P_1(\varepsilon)$. By Theorem 6, $(\varepsilon, \delta(\varepsilon))$ is a point on the Pareto front. We now view the Pareto front as a function, $\delta : \overline{\mathcal{E}} \longmapsto \mathbb{R}_+$, which maps every $\varepsilon \in \overline{\mathcal{E}}$ to the corresponding $\delta(\varepsilon)$. In the following development, we use the Pareto front, as defined in Definition 4, and the function, δ, interchangeably. The following lemmas establish properties of the Pareto front.

Lemma 12: The Pareto front is strictly decreasing.

Proof: The proof follows from Corollary 1. ■

[9]This follows from Lemma 11. Also, note that since $\mathbf{B}_\varepsilon$, $\mathbf{P}_\varepsilon$, is a solution of $P_1(\varepsilon)$, $\mathbf{B}_\varepsilon$, $\mathbf{P}_\varepsilon$, is Pareto optimal from Theorem 6.

Lemma 13: The Pareto front is convex, continuous, and, its left and right derivatives[10] exist at each point on the Pareto front. Also, when $\varepsilon_{\text{exact}} = +\infty$, we have

$$\delta(1) = \lim_{\varepsilon \to 1} \delta(\varepsilon) = 0. \tag{3.79}$$

Proof: Let $\varepsilon = f_2(\cdot)$ be the horizontal axis of the Pareto front, and let $\delta(\varepsilon) = f_1(\cdot)$ be the vertical axis. By definition of the Pareto front, for each pair $(\varepsilon, \delta(\varepsilon))$ on the Pareto front, there exists matrices $\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon$ such that

$$\|\mathbf{P}_\varepsilon\| = \varepsilon, \qquad \text{and} \qquad \|\mathbf{B}_\varepsilon + \mathbf{P}_\varepsilon \mathbf{W} - \mathbf{W}\| = \delta(\varepsilon). \tag{3.80}$$

Let $(\varepsilon_1, \delta(\varepsilon_1))$ and $(\varepsilon_2, \delta(\varepsilon_2))$ be two points on the Pareto front, such that $\varepsilon_1 < \varepsilon_2$. Then, there exists $\mathbf{B}_1, \mathbf{P}_1$, and $\mathbf{B}_2, \mathbf{P}_2$, such that

$$\|\mathbf{P}_1\| = \varepsilon_1, \qquad \text{and} \qquad \|\mathbf{B}_1 + \mathbf{P}_1 \mathbf{W} - \mathbf{W}\| = \delta(\varepsilon_1), \tag{3.81}$$

$$\|\mathbf{P}_2\| = \varepsilon_2, \qquad \text{and} \qquad \|\mathbf{B}_2 + \mathbf{P}_2 \mathbf{W} - \mathbf{W}\| = \delta(\varepsilon_2). \tag{3.82}$$

For some $0 \leq \mu \leq 1$, define

$$\mathbf{B}_3 = \mu \mathbf{B}_1 + (1-\mu)\mathbf{B}_2, \tag{3.83}$$

$$\mathbf{P}_3 = \mu \mathbf{P}_1 + (1-\mu)\mathbf{P}_2. \tag{3.84}$$

Clearly, $[\mathbf{B}_3 \mid \mathbf{P}_3] \in \mathcal{F}_{\leq 1}$ as the sparsity constraint is not violated and

$$\|\mathbf{P}_3\| \leq \mu \|\mathbf{P}_1\| + (1-\mu)\|\mathbf{P}_2\| < 1, \tag{3.85}$$

since $\|\mathbf{P}_1\| < 1$ and $\|\mathbf{P}_2\| < 1$. Let

$$\varepsilon_3 = \|\mathbf{P}_3\|, \tag{3.86}$$

and let

$$z(\varepsilon_3) = \|\mathbf{B}_3 + \mathbf{P}_3 \mathbf{W} - \mathbf{W}\|. \tag{3.87}$$

[10] At $\varepsilon = 0$, only the right derivative is defined and at $\varepsilon = \sup \overline{\mathcal{E}}$, only the left derivative is defined.

We have

$$\begin{aligned} z(\varepsilon_3) &= \|\mu\mathbf{B}_1 + (1-\mu)\mathbf{B}_2 + (\mu\mathbf{P}_1 + (1-\mu)\mathbf{P}_2)\mathbf{W} - \mathbf{W}\|, \\ &= \|\mu\mathbf{B}_1 + \mu\mathbf{P}_1\mathbf{W} - \mu\mathbf{W} + (1-\mu)\mathbf{B}_2 + (1-\mu)\mathbf{P}_2\mathbf{W} - (1-\mu)\mathbf{W}\|, \\ &\leq \mu\|\mathbf{B}_1 + \mathbf{P}_1\mathbf{W} - \mathbf{W}\| + (1-\mu)\|\mathbf{B}_2 + \mathbf{P}_2\mathbf{W} - \mathbf{W}\|, \\ &= \mu\delta(\varepsilon_1) + (1-\mu)\delta(\varepsilon_2). \end{aligned} \tag{3.88}$$

Since $(\varepsilon_3, z(\varepsilon_3))$ may not be Pareto-optimal, there exists a Pareto optimal point, $(\varepsilon_3, \delta(\varepsilon_3))$, at ε_3 (from Lemma 11) and we have

$$\delta(\varepsilon_3) \leq z(\varepsilon_3) \leq \mu\delta(\varepsilon_1) + (1-\mu)\delta(\varepsilon_2). \tag{3.89}$$

From (3.85), we have $\varepsilon_3 \leq \mu\varepsilon_1 + (1-\mu)\varepsilon_2$, and since the Pareto front is strictly decreasing (from Lemma 12), we have

$$\delta(\mu\varepsilon_1 + (1-\mu)\varepsilon_2) \leq \delta(\varepsilon_3). \tag{3.90}$$

From (3.90) and (3.89), we have

$$\delta(\mu\varepsilon_1 + (1-\mu)\varepsilon_2) \leq \mu\delta(\varepsilon_1) + (1-\mu)\delta(\varepsilon_2), \tag{3.91}$$

which establishes convexity of the Pareto front. Since, the Pareto front is convex, it is continuous, and it has left and right derivatives [83].

Clearly, $\delta(1) = \lim_{\varepsilon\to 1}\delta(\varepsilon)$ by continuity of the Pareto front. By choosing $\mathbf{P} = \mathbf{I}$ and $\mathbf{B} = \mathbf{0}$, we have $\delta(1) = 0$. Note that $(1, 0)$ lies on the Pareto front when $\varepsilon_{\text{exact}} = +\infty$. Indeed, for any $\mathbf{B}, \mathbf{P}$ satisfying the sparsity constraints, we simultaneously cannot have

$$\|\mathbf{P}\| \leq 1, \tag{3.92}$$

$$\text{or} \qquad \|\mathbf{B} + \mathbf{P}\mathbf{W} - \mathbf{W}\| \leq 0, \tag{3.93}$$

with strict inequality in at least one of the above equations. Thus, the pair $\mathbf{B} = \mathbf{0}, \mathbf{P} = \mathbf{I}$

is Pareto-optimal leading to $\delta(1) = 0$. ■

3.7.3 Proof of Theorem 5

With the Pareto-optimal solutions of MOP established in Section 3.7.1 and the properties of the Pareto front in Section 3.7.2, we now prove Theorem 5.

Proof: We prove the theorem by contradiction. Let $\|\mathbf{P}_\varepsilon\| = \varepsilon' \leq \varepsilon$, and $\delta' = \|\mathbf{B}_\varepsilon + \mathbf{P}_\varepsilon\mathbf{W} - \mathbf{W}\|$. Assume, on the contrary, that $\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon$ is not Pareto-optimal. From Lemma 11, there exists a Pareto-optimal solution $\mathbf{B}^*, \mathbf{P}^*$, at ε', such that

$$\|\mathbf{P}^*\| = \varepsilon', \text{ and } \delta(\varepsilon') = \|\mathbf{B}^* + \mathbf{P}^*\mathbf{W} - \mathbf{W}\|, \tag{3.94}$$

with $\delta(\varepsilon') < \delta'$, since $\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon$, is not Pareto-optimal. Since, $\|\mathbf{P}_\varepsilon\| = \varepsilon' \leq \varepsilon$, the Pareto-optimal solution, $\mathbf{B}^*, \mathbf{P}^*$, is feasible for the *Learning Problem*. In this case, the utility function for the Pareto-optimal solution, $\mathbf{B}^*, \mathbf{P}^*$, is

$$u(\mathbf{B}^*, \mathbf{P}^*) = \frac{(\varepsilon')}{1-\varepsilon'} < \frac{\delta'}{1-\varepsilon'} = u(\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon). \tag{3.95}$$

Hence, $\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon$ is not an optimal solution of the *Learning Problem*, which is a contradiction. Hence, $\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon$ is Pareto-optimal. ■

The above theorem suggests that it suffices to find the optimal solutions of the *Learning Problem* from the set of Pareto-optimal solutions, i.e., the Pareto front. The next section addresses the minimization of the utility function, $u(\mathbf{B}, \mathbf{P})$, and formulates the performance-convergence rate trade-offs.

3.8 Minimization of the utility function

In this section, we develop the solution of the *Learning Problem* from the Pareto front. The solution of the *Learning Problem* (3.55) lies on the Pareto front as already established in Theorem 5. Hence, it suffices to choose a Pareto-optimal solution from the Pareto front that minimizes (3.55) under the given constraints. In the following, we study properties of the utility function.

3.8.1 Properties of the utility function

With the help of Theorem 5, we now restrict the utility function to the Pareto-optimal solutions[11]. By Lemma 11, for every $\varepsilon \in \mathcal{E}$, there exists a Pareto-optimal solution, $\mathbf{B}_\varepsilon, \mathbf{P}_\varepsilon$, with

$$\|\mathbf{P}_\varepsilon\| = \varepsilon, \qquad \text{and} \qquad \|\mathbf{B}_\varepsilon + \mathbf{P}_\varepsilon \mathbf{W} - \mathbf{W}\| = \delta(\varepsilon). \tag{3.96}$$

Also, we note that, for any Pareto-optimal solution, $\mathbf{B}, \mathbf{P}$, the corresponding utility function is

$$u(\mathbf{B}, \mathbf{P}) = \frac{\|\mathbf{B} + \mathbf{P}\mathbf{W} - \mathbf{W}\|}{1 - \|\mathbf{P}\|} = \frac{\delta(\|\mathbf{P}\|)}{1 - \|\mathbf{P}\|}. \tag{3.97}$$

This permits us to redefine the utility function as, $u^* : \mathcal{E} \longmapsto \mathbb{R}_+$, such that, for any Pareto-optimal solution, $\mathbf{B}, \mathbf{P}$,

$$u(\mathbf{B}, \mathbf{P}) = u^*(\|\mathbf{P}\|) \tag{3.98}$$

We establish properties of u^*, which enable determining the solutions of the *Learning Problem.*

Lemma 14: The function $u^*(\varepsilon)$, for $\varepsilon \in \mathcal{E}$, is non-increasing, i.e., for $\varepsilon_1, \varepsilon_2 \in \mathcal{E}$ with $\varepsilon_1 < \varepsilon_2$, we have

$$u^*(\varepsilon_2) \leq u^*(\varepsilon_1). \tag{3.99}$$

Hence,

$$\min_{[\mathbf{B} \mid \mathbf{P}] \in \mathcal{F}_{\leq \varepsilon}} u(\mathbf{B}, \mathbf{P}) = u^*(\varepsilon). \tag{3.100}$$

Proof: Consider $\varepsilon_1, \varepsilon_2 \in \mathcal{E}$ such that $\varepsilon_1 < \varepsilon_2$, then ε_2 is a convex combination of ε_1 and 1, i.e., there exists a $0 < \mu < 1$ such that

$$\varepsilon_2 = \mu\varepsilon_1 + (1 - \mu). \tag{3.101}$$

[11]Note that when $\varepsilon_{\text{exact}} = +\infty$, the solution $\mathbf{B} = \mathbf{0}, \mathbf{P} = \mathbf{I}$ is Pareto-optimal, but the utility function is undefined here, although the MOP is well-defined. Hence, for the utility function, we consider only the Pareto-optimal solutions with $\|\mathbf{P}\|$ in $\mathcal{E}$.

From Lemma 11, there exist $\delta(\varepsilon_1)$ and $\delta(\varepsilon_2)$ on the Pareto front corresponding to ε_1 and ε_2, respectively. Since the Pareto front is convex (from Lemma 13), we have

$$\delta(\varepsilon_2) \leq \mu\delta(\varepsilon_1) + (1-\mu)\delta(1). \tag{3.102}$$

Recall that $\delta(1) = 0$; we have

$$u^*(\varepsilon_2) = \frac{\delta(\varepsilon_2)}{1-\mu\varepsilon_1-(1-\mu)} = \frac{\delta(\varepsilon_2)}{\mu(1-\varepsilon_1)} \leq \frac{\mu\delta(\varepsilon_1)}{\mu(1-\varepsilon_1)}, \tag{3.103}$$

and (3.99) follows.

We now have

$$\begin{aligned} \min_{[\mathbf{B} \mid \mathbf{P}]\in\mathcal{F}_\varepsilon} u(\mathbf{B},\mathbf{P}) &= \min_{[\mathbf{B} \mid \mathbf{P}]\in\mathcal{F}_\varepsilon,\ (\mathbf{B},\mathbf{P}) \text{ is P.O.}} u(\mathbf{B},\mathbf{P}), \\ &= \min_{\|\mathbf{P}\|\leq\varepsilon,\ (\mathbf{B},\mathbf{P}) \text{ is P.O.}} u(\mathbf{B},\mathbf{P}), \\ &= \min_{0\leq\varepsilon'\leq\varepsilon} u^*(\varepsilon'), \\ &= u^*(\varepsilon), \end{aligned} \tag{3.104}$$

where P.O. stands for Pareto-optimal. The first step follows from Theorem 5. The second step is just a restatement since the sparsity constraints are included in the MOP. The third step follows from the definition of u^* and finally, we use the non-increasing property of u^* to get the last equation. ■

We now study the cost of the utility function. From Lemma 14, we note that this cost is non-increasing as ε increases. When $\varepsilon_{\text{exact}} < 1$, this cost is 0. When $\varepsilon_{\text{exact}} = +\infty$, we may be able to decrease the cost as $\varepsilon \to 1$. We now define the limiting cost.

Definition 5: [Infimum cost] The infimum cost, c_{inf}, of the utility function is defined as

$$c_{\text{inf}} \triangleq \begin{cases} \lim_{\varepsilon\to 1} u^*(\varepsilon), & \text{if } \varepsilon_{\text{exact}} = +\infty, \\ 0, & \text{otherwise.} \end{cases} \tag{3.105}$$

Clearly, the cost does not increase as $\varepsilon \to 1$ from Lemma 14. If $\varepsilon_{\text{exact}} = +\infty$, it is not possible for the utility function, $u^*(\varepsilon)$, to attain c_{inf}, since $u^*(\varepsilon)$ is undefined at $\|\mathbf{P}\| = 1$, instead the utility function can have a value as close as desired to c_{inf},

but it cannot attain c_{inf}. The following lemma establishes the cost of the utility function, $u^*(\varepsilon)$, as $\varepsilon \to 1$.

Lemma 15: If $\varepsilon_{\text{exact}} = +\infty$, then the infimum cost, c_{inf}, is the negative of the left derivative, $D^-(\delta(\varepsilon))$, of the Pareto front evaluated at $\varepsilon = 1$.

Proof: Recall that $\delta(1) = 0$. Then c_{inf} is given by

$$\begin{aligned} c_{\text{inf}} &= \lim_{\varepsilon \to 1} u^*(\varepsilon), \\ &= \lim_{\varepsilon \to 1} \frac{\delta(\varepsilon)}{1-\varepsilon}, \\ &= \lim_{\varepsilon \to 1} \frac{\delta(\varepsilon) - \delta(1)}{1-\varepsilon}, \\ &= -D^-(\delta(\varepsilon))|_{\varepsilon=1}. \end{aligned} \tag{3.106}$$

■

3.8.2 Graphical representation of the analytical results

In this section, we graphically view the analytical results developed earlier. To this aim, we establish a graphical procedure using the following lemma.

Lemma 16: Let $(\varepsilon, \delta(\varepsilon))$ be a point on the Pareto front and $g(\varepsilon)$ a straight line that passes through $(\varepsilon, \delta(\varepsilon))$ and $(1, 0)$. The cost associated to the Pareto-optimal solution(s) corresponding to $(\varepsilon, \delta(\varepsilon))$ is both the (negative) slope and the intercept (on the vertical axis) of $g(\varepsilon)$.

Proof: We define the straight line, $g(\varepsilon)$, as

$$g(\varepsilon) = c_1 \varepsilon + c_2, \tag{3.107}$$

where c_1 is its slope and c_2 is its intercept on the vertical axis. Since $g(\varepsilon)$ passes through $(\varepsilon, \delta(\varepsilon))$ and $(1, 0)$, its slope, c_1, is given by

$$c_1 = \frac{\delta(\varepsilon) - 0}{\varepsilon - 1} = -u^*(\varepsilon). \tag{3.108}$$

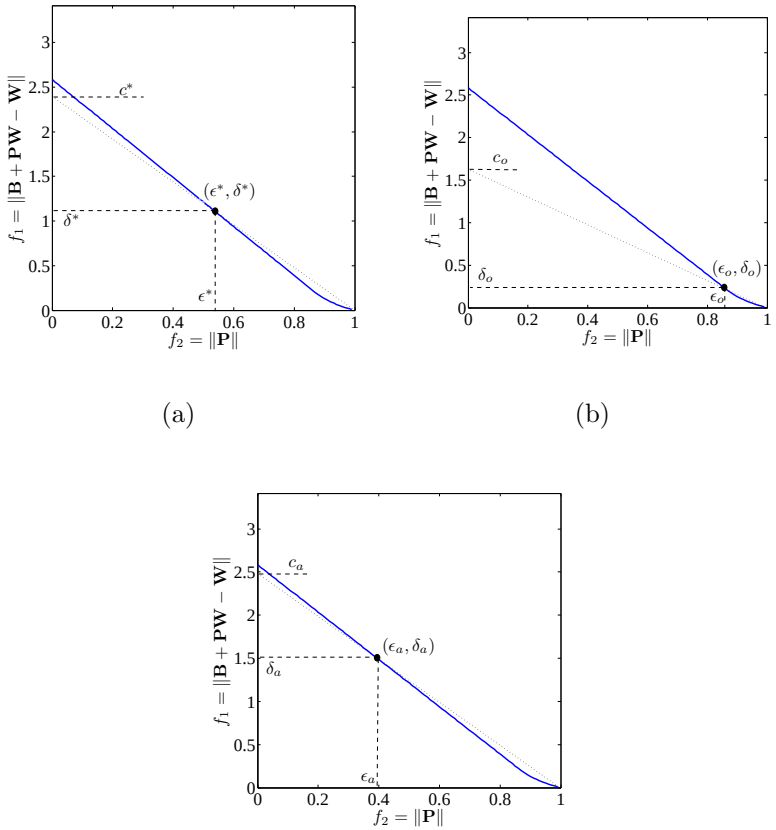


Figure 3.1: (a) Graphical illustration of Lemma 16. (b) Illustration of case (i) in performance-speed tradeoff. (c) Illustration of case (ii) in performance-speed tradeoff.

Since $g(\varepsilon)$ passes through $(1, 0)$, at $\varepsilon = 1$ we have

$$c_2 = [g(\varepsilon) - c_1\varepsilon]_{\varepsilon=1} = g(1) - c_1 = u^*(\varepsilon). \tag{3.109}$$

■

Figure 3.1(a) illustrates Lemma 16, graphically. Let $(\varepsilon^*, \delta^*)$ be a point on the Pareto front. The cost, c^*, of the utility function, $u^*(\varepsilon^*)$, is the intercept of the straight line passing through $(\varepsilon^*, \delta^*)$ and $(1, 0)$.

3.8.3 Performance-speed tradeoff: $\varepsilon_{\mathbf{exact}} = +\infty$

In this case, no matter how large we choose $\|\mathbf{P}\|$, the HDC does not converge to the exact solution. By Lemma 8, the convergence rate of the HDC depends on $\rho(\mathbf{P})$ and thus upper bounding $\|\mathbf{P}\|$ leads to a guarantee on the convergence rate. Also, from Lemma 14, the utility function is non-increasing as we increase $\|\mathbf{P}\|$. We formulate the *Learning Problem* as a performance-speed tradeoff. From the Pareto front and the constant cost straight lines, we can address the following two questions.

(i) Given a pre-specified performance, c_o (the cost of the utility function), choose a Pareto-optimal solution that results into the fastest convergence of the HDC to achieve c_o. We carry out this procedure by drawing a straight line that passes the points $(0, c_o)$ and $(1, 0)$ in the Pareto plane. Then, we pick the Pareto-optimal solution from the Pareto front that lies on this straight line and also has the smallest value of $\|\mathbf{P}\|$. See Figure 3.1(b).

(ii) Given a pre-specified convergence speed, ε_a, of the HDC algorithm, choose a Pareto-optimal solution that results into the smallest cost of the utility function, $u(\mathbf{B}, \mathbf{P})$. We carry out this procedure by choosing the Pareto-optimal solution, $(\varepsilon_a, \delta_a)$, from the Pareto front. The cost of the utility function for this solution is then the intercept (on the vertical axis) of the constant cost line that passes through both $(\varepsilon_a, \delta_a)$ and $(1, 0)$. See Figure 3.1(c).

We now characterize the steady state error. Let $\mathbf{B}_o, \mathbf{P}_o$, be the operating point of the HDC obtained from either of the two tradeoff scenarios described above. Then, the steady state error in the limiting state, $\mathbf{X}_\infty$, of the network when the HDC with $\mathbf{B}_o, \mathbf{P}_o$ is implemented, is given by

$$e_{ss} = \|(\mathbf{I} - \mathbf{P}_o)^{-1}\mathbf{B}_o - \mathbf{W}\|, \tag{3.110}$$

which is clearly bounded above by (3.54).

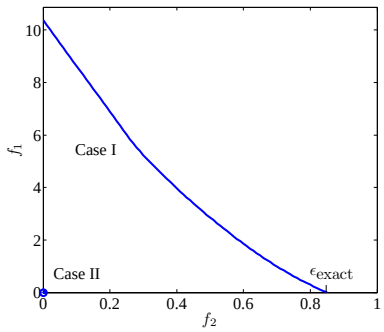


Figure 3.2: Typical Pareto front.

3.8.4 Exact solution: $\varepsilon_{\text{exact}} < 1$

In this case, the optimal operating point of the HDC algorithm is the Pareto-optimal solution corresponding to $(\varepsilon_{\text{exact}}, 0)$ on the Pareto front. A typical Pareto front in this case is shown in Figure 3.2, labeled as Case I. A special case is when the sparsity pattern of $\mathbf{B}$ is the same as the sparsity of the weight matrix, $\mathbf{W}$. We can then choose

$$\mathbf{B} = \mathbf{W}, \qquad \mathbf{P} = \mathbf{0}, \tag{3.111}$$

as the solution to the *Learning Problem* and the Pareto front is a single point $(0, 0)$ shown as Case II in Figure 3.2.

If it is desirable to operate the HDC algorithm at a faster speed than corresponding to $\varepsilon_{\text{exact}}$, we can consider the performance-speed tradeoff in Section 3.8.3 to get the appropriate operating point.

3.9 Simulations

In this section, we revisit the leader-follower setup introduced in Section 3.6.1. We consider a multi-agent system with $N = 24$ agents inter-connected using the nearest-neighbor rule, with $M_1 = 10$ ground robots, $M_2 = 10$ aerial robots and $K = 4$ humans. We choose the robots as the followers (sensors), i.e., we have $M = M_1 + M_2 = 20$, and the humans as the leaders (anchors). This setup is shown in Fig. 3.3(a) where '$\circ$' represents a ground robot, '$\times$' represents an aerial robot, and '∇' represents a human

anchor. We may consider the four anchors that are located at four corners on the ground as base-stations that generate the operating instructions to the robots. Each robot intends to receive operating instructions from one of the four humans but the underlying communication network is restricted. We implement the HDC algorithm to convey this information, i.e., if x_i is the state of the ith robot, the HDC algorithm requires

$$\lim_{t\to\infty} x_i(t+1) = \mathbf{w}_i\mathbf{u}, \tag{3.112}$$

where $\mathbf{w}_i$ is a 1×4 row-vector. Without loss of generality, we choose $\mathbf{w}_i = [1,\ 0,\ 0,\ 0]$ for $i = 1,\ldots,5$, $\mathbf{w}_i = [0,\ 1,\ 0,\ 0]$ for $i = 6,\ldots,10$, $\mathbf{w}_i = [0,\ 0,\ 1,\ 0]$ for $i = 11,\ldots,15$, and $\mathbf{w}_i = [0,\ 0,\ 0,\ 1]$ for $i = 16,\ldots,20$. For simulation purposes, we choose $\mathbf{u} = [15,\ 5,\ -5,\ -15]^T$.

For the underlying communication graph shown in Fig. 3.3(a), the Pareto-front is shown in Fig. 3.3(b). It turns out that in this case, there is indeed an exact solution as shown on the Pareto-front. To study the performance-speed tradeoff, we choose three operating points on the Paerto-front corresponding to $f_2 = \|\mathbf{P}\| \in \{0.3,\ 0.7, \varepsilon_{\text{exact}} = 0.9\}$. We implement the HDC algorithm with these choices of f_2. The performance of the HDC algorithm for these different operating points is shown in Fig. 3.3(c) where we plot the normalized error summed over the entire network. Clearly, we can trade convergence speed with the accuracy of the algorithm. For instance, notice that as we get closer to $\varepsilon_{\text{exact}}$, the steady state error is reduced on the expense of the convergence speed.

3.10 Conclusions

In this chapter, we describe a framework for the analysis and synthesis of linear distributed algorithms that we term High Dimensional Consensus (HDC). HDC contains average-consensus as a special case. We present the conditions under which the HDC converges, the limiting state, and its convergence rate of the HDC.

We then focus on the design of the HDC parameters so that the network states converge to a pre-specified state. We term this as the learning problem. We show that

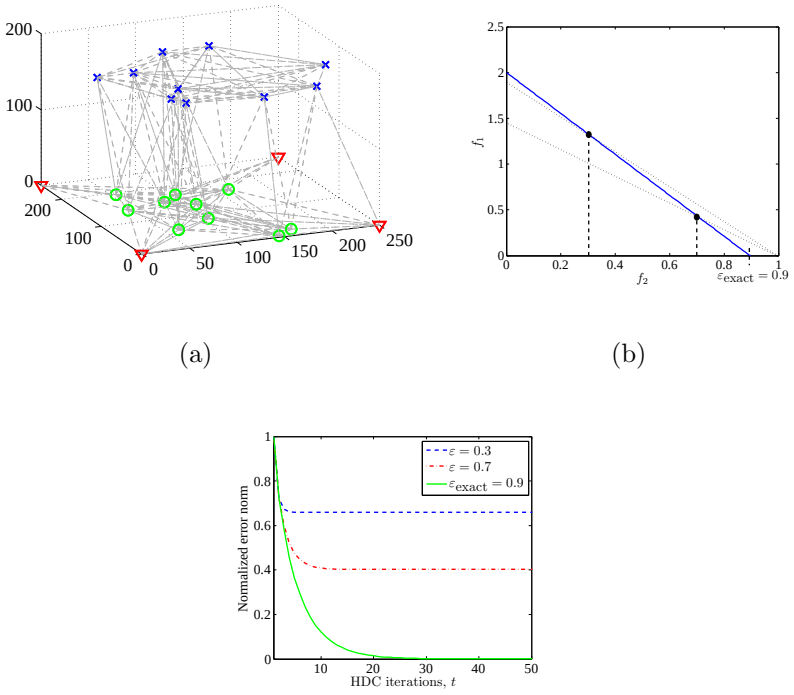


Figure 3.3: (a) Multi-agent system: Network of ground robots '$\circ$', aerial robots '$\times$', and humans '∇'. (b) The Pareto-front for the given $\mathbf{W}$ and the underlying communication graph. (c) HDC algorithm implemented for three different scenarios reflecting the performance-speed tradeoff.

the solution of this learning problem is a Pareto-optimal solution to a multi-objective optimization problem (MOP). We explicitly prove the Pareto-optimality of the MOP solutions. We then prove that the Pareto front (collections of the Pareto-optimal solutions) is convex and strictly decreasing. Using these properties, we solve the learning problem and formulate performance-speed tradeoffs. We include experimental results for the leader-follower architecture in multi-agent systems that learns the parameters when we have multiple leaders. We explore different performance-speed tradeoffs for this example.

Chapter 4

Localization in sensor networks

This chapter develops DILOC, a special case of the HDC algorithm (discussed in Chapter 3). DILOC is a *distributed*, *iterative* algorithm to locate M sensors (with unknown locations) in $\mathbb{R}^m, m \geq 1$ (for example, $m = 2$ corresponds to nodes lying on a plane, while $m = 3$ corresponds to nodes in 3d space), with respect to a minimal number, $K = m + 1$, of anchors with known locations. DILOC uses the barycentric coordinates of a node with respect to its neighbors; these coordinates are computed using the Cayley-Menger determinants, i.e., the determinants of matrices of inter-node distances. We show convergence of DILOC by associating with it an absorbing Markov chain whose absorbing states are the states of the anchors. In particular, DILOC requires each sensor to communicate to exactly $m+1$ carefully chosen neighbors. We relax this and consider certain enhancements to DILOC, namely, when we have (i) dynamic network topology; (ii) more than $m + 1$ anchors; and (iii) more than $m + 1$ neighbors. We then provide an algorithm, MDL, which performs distributed localization and tracking in networks of mobile agents. Finally, we study localization in random environments (communication failures, communication noise, and noisy distance measurements).

Parts of this chapter have been presented in [41, 44, 50, 51, 54, 57].

4.1 Introduction

Localization is a fundamental problem in sensor networks. Information about the location of the sensors is key to process the sensors' measurements accurately. In applications where sensors are deployed randomly, they have no knowledge of their exact locations, but equipping each of them with a localization device like a GPS is expensive, not robust to jamming in military applications, and is usually of limited use in indoor environments. We develop a distributed (decentralized) localization algorithm where the sensors find their locations under a limited set of assumptions and conditions.

For DILOC, we assume that there are exactly $m+1$ *anchors*, i.e., $K = m+1$. In the context of sensor localization, anchors are the nodes in the network that know their exact locations in $\mathbb{R}^m$. The goal is then to locate the remaining M nodes in the network, which we call *sensors* and which do not know their locations, with respect to the anchors[1]. The problem is practically interesting when $M \gg m+1$. Two important characteristics in this work are: (i) we assume that the sensors lie in the convex hull of the anchors; and, from this, it follows that (ii) each sensor can find a triangulation set, i.e., a set of $m+1$ nodes (a possible combination of anchors and sensors) such that the sensor in question lies in the convex hull of this triangulation set. The chapter will discuss the practical significance of these assumptions.

In applications with large-scale sensor networks, the distance of most of the M sensors to the $m+1$ anchors is large, so that it is impractical for the sensors to communicate *directly* with the anchors. Further, because M is assumed very large, to compute the locations of the sensors at a central station is not feasible, as it would require a large communication effort, expensive large-scale computation, and add latency and bottlenecks to the network operation. These networks call for efficient *distributed* algorithms where each node communicates directly only with a few neighboring nodes (either sensors or anchors) and a low order computation is performed locally at the node and at each iteration of the algorithm, for example. We present

[1]In the sequel, the term *node* refers to either *anchors* (known location) or *sensors* (locations to be determined). In a few exceptions, easily resolved from the context, we will still write *sensors*, when we actually mean nodes.

here the Distributed Iterative LOCalization algorithm (DILOC, pronounced *die-lock*) that overcomes the above challenges in large-scale randomly deployed networks.

In DILOC, the sensors start with an initial estimate of their locations, that can be a random guess. This random guess is arbitrary and does not need to place the sensors in the convex hull of the anchors. The sensors then update their locations, which we call the state of the network, by exchanging their state information *only* with a carefully chosen subset of $m + 1$ of their neighbors (see (ii) above.) This state updating is a convex combination of the states of the neighboring nodes. The coefficients of the convex combination are the barycentric coordinates of the sensors with respect to their neighbors, [84, 85], which are determined from the Cayley-Menger determinants [86], see Appendix B.2.

We further extend DILOC to dynamic network topologies and more than $m + 1$ anchors and neighbors. When the sensor network consists of mobile agents[2] e.g., robots, vehicles or cell-phones, whose positions change as a function of time, simple DILOC updates are not applicable since the network configuration is not static. Hence, the state update (the optimal combining coefficients) changes with time as the neighborhood at each sensor changes. In the mobile case, the localization problem is not only to estimate the starting position of the agents, but also to track their motion. To this end, we provide an algorithm MDL for localization in networks of mobile agents.

In the following, we contrast our work with the existing literature on sensor localization.

Brief review of the literature: The literature on localization algorithms may be broadly characterized into centralized and distributed algorithms. Illustrative centralized localization algorithms include: maximum likelihood estimators that are formulated when the data is known to be described by a statistical model, [87, 88]; multi-dimensional scaling (MDS) algorithms that formulate the localization problem as a least squares problem at a centralized location, [89, 90]; work that exploits the geometry of the Euclidean space, like when locating a single robot using trilateration

[2]The term *agents* and *sensors* mean the same and are used interchangeably to denote the network elements with unknown locations.

in $m = 3$-dimensional space, see [91] where a geometric interpretation is given to the traditional algebraic distance constraint equations; localization algorithms with imprecise distance information, see [92] where the authors exploit the geometric relations among the distances in the optimization procedure; for additional work, see, e.g., [93, 94]. Centralized algorithms are fine in small or tethered network environments; but in large untethered networks, they incur high communication cost and may not be scalable; they depend on the availability and robustness of a central processor and have a single point of failure.

Distributed localization algorithms can be characterized into two classes: multilateration and successive refinements. In multilateration algorithms, [95, 96, 97, 98], each sensor estimates its range from the anchors and then calculates its location via multilateration, [99]. The multilateration scheme requires a high density of anchors, which is a practical limitation in large sensor networks. Further, the location estimates obtained from multilateration schemes are subject to large errors because the estimated sensor-anchor distance in large networks, where the anchors are far apart, is noisy. To overcome this problem, a high density of anchors is required. We, on the other hand, do not estimate distances to far-away nodes. Only local distances to nearby nodes are estimated; these should have better accuracy. This allows us to employ the minimal number $m + 1$ of anchors (for localization in $\mathbb{R}^m$.)

A distributed multidimensional scaling algorithm is presented in [100]. Successive refinement algorithms that perform an iterative minimization of a cost function are presented in [101, 102, 103]. Reference [101] discusses an iterative scheme where they assume 5% of the nodes as anchors. Reference [103] discusses a Self-Positioning Algorithm (SPA) that provides a GPS-free positioning and builds a relative coordinate system.

Another formulation to solve localization problems in a distributed fashion is the probabilistic approach. Nonparametric belief propagation on graphical models is used in [104]. Sequential Monte Carlo methods for mobile localization are considered in [105]. Particle filtering methods have been addressed in [106] where each sensor stores representative particles for its location that are weighted according to their likelihood. Reference [107] tracks and locates mobile robots using such probabilistic

methods.

In the context of localization for mobile networks, reference [108] discusses a tracking algorithm that tracks objects using wireless sensing devices sensing the objects. The wireless devices already know their locations and tracking is achieved by aggregating the sensed information. Another interesting tracking algorithm is provided in [109] that tracks humans on a tiled floor by using pressure sensors on the tiles and pressure and gait patterns of the humans. Our algorithm falls into token localization and tracking where the objects to be tracked possess a sensor (or an RFID tag) and helps the tracker. Reference [110] uses trilaterations to solve the localization/tracking problem that requires a large number of close by anchors to have a reasonable location estimate. Some other relevant references in this direction include [111, 112, 113, 114].

In comparison with these references, DILOC is equivalent to solving by a *distributed* and *iterative* algorithm a large system of linear algebraic equations where the system matrix is highly sparse. Our method exploits the structure of this matrix, which results from the topology of the communication graph of the network.

We now describe the rest of the chapter. Section 4.2 presents preliminaries and then DILOC, the distributed iterative localization algorithm, that is based on barycentric coordinates, generalized volumes, and Cayley-Menger determinants. Section 4.3 proves DILOC's convergence. Section 4.4 presents the DILOC-REL, DILOC with relaxation, and proves that it asymptotically reduces to the deterministic case without relaxation. We then consider enhancements to DILOC in Section 4.5, i.e., Section 4.5.1 provides DILOC for dynamic network topologies, Section 4.5.2 provides DILOC for more than $m+1$ anchors, and Section 4.5.3 provides DILOC where each sensor utilizes the information from all of its neighbors. We then consider localization for networks of mobile agents in Section 4.6. Specifically, Section 4.6.1 presents our motion model, Section 4.6.2 presents the MDL algorithm, whereas Section 4.6.3 discusses the convergence proof of the MDL algorithm. We then briefly outline distributed localization in random environments in Section 4.7. Finally, Section 4.8 presents the simulations for all of the above scenarios and Section 4.9 concludes the chapter. Appendices B.1–B.2 provide the convex hull inclusion test, and the Cayley-Menger determinant.

4.2 Distributed sensor localization: DILOC

In this section, we formally state DILOC (Distributed Iterative LOCalization algorithm) in m-dimension Euclidean space, $\mathbb{R}^m$ ($m \geq 1$), and introduce relevant notation. Of course, for sensor localization, $m = 1$ (sensors in a straight line), $m = 2$ (plane), or $m = 3$ (3d-space.) The generic case of $m > 3$ is of interest, for example, when the graph nodes represent m-dimensional feature vectors in classification problems, and the goal is still to find in a distributed fashion their global coordinates (with respect to a reference frame.) Since our results are general, we deal with m-dimensional 'localization,' but, for easier accessibility, the reader may consider $m = 2$ or $m = 3$.

To provide a quantitative assessment on some of the assumptions underlying DILOC, we will, when needed, assume that the deployment of the sensors in a given region follows a Poisson distribution. This random deployment is often assumed and is realistic; we use it to derive probabilistic bounds on the deployment density of the sensors and on the communication radius at each sensor; these can be straight forwardly related to the values of network field parameters (like transmitting power or signal-to-noise ratio) in order to implement DILOC. We discuss the computation/communication complexity of the algorithm and provide a simplistic, yet insightful, example that illustrates DILOC.

4.2.1 Notation

Recall that the sensors and anchors are in $\mathbb{R}^m$. Let Θ be the set of nodes in the network decomposed as

$$\Theta = \kappa \cup \Omega, \tag{4.1}$$

where κ is the set of anchors, i.e., the nodes whose locations are known, and Ω is the set of sensors whose locations are to be determined. By $|\cdot|$ we mean the cardinality of the set, and we let $|\Theta| = N$, $|\kappa| = m + 1$, and $|\Omega| = M$. For a set Ψ of nodes, we denote its convex hull by $\mathcal{C}(\Psi)$[3]. For example, if Ψ is a set of three non-collinear nodes in a plane, then $\mathcal{C}(\Psi)$ is a triangle. Let A_Ψ be the generalized volume (area in $m = 2$,

[3]The convex hull, $\mathcal{C}(\Psi)$, of a set of points in Ψ is the minimal convex set containing Ψ.

volume in $m = 3$, and their generalization in higher dimensions) of $\mathcal{C}(\Psi)$. Let d_{lk} be the Euclidean distance between two nodes $l, k \in \Theta$, their inter-node distance; the neighborhood of node l in a given radius, r_l, is

$$\mathcal{K}(l, r_l) = \{k \in \Theta : \ d_{lk} < r_l\}. \tag{4.2}$$

Let $\mathbf{c}_l$ be the m-dimensional coordinate vector for node, $l \in \Theta$, with respect to a global coordinate system, written as the m-dimensional row vector,

$$\mathbf{c}_l \ = \ [c_{l,1}, c_{l,2}, \ldots, c_{l,m}]. \tag{4.3}$$

The true (possibly unknown) location of node l is represented by $\mathbf{c}_l^*$. Because the distributed localization algorithm DILOC is iterative, $\mathbf{c}_l(t)$ will represent the location estimates, or state, for node l at iteration t.

4.2.2 Distributed iterative localization algorithm.

We state explicitly the assumptions that we make when developing DILOC.

(B0) Convexity. All the sensors lie inside the convex hull of the anchors

$$\mathcal{C}(\Omega) \subseteq \mathcal{C}(\kappa). \tag{4.4}$$

(B1) Anchor nodes. The anchors' locations are known, i.e., their state remains constant

$$\mathbf{c}_q(t) = \mathbf{c}_q^*, \qquad q \in \kappa, \ t \geq 0. \tag{4.5}$$

(B2) Nondegeneracy. The generalized volume[4] for κ, $A_\kappa \neq 0$.

From **(B0)**, the next Lemma follows easily.

Lemma 17 (Triangulation): For every sensor $l \in \Omega$, there *exists* some $r_l > 0$ such

[4]Nondegeneracy simply states that the anchors do not lie on a hyperplane. If this was the case, then the localization problem reduces to a lower dimensional problem, i.e., $\mathbb{R}^{m-1}$ instead of $\mathbb{R}^m$. For instance, if all the anchors in the network lie on a plane in $\mathbb{R}^3$, by **(B0)**, the sensors also lie in $\mathbb{R}^3$, and the localization problem can be thought of as localization in $\mathbb{R}^2$.

that a *triangulation set*, $\Theta_l(r_l)$, satisfying the following conditions

$$\Theta_l\left(r_l\right) \subseteq \mathcal{K}\left(l, r_l\right),\ l \notin \Theta_l\left(r_l\right),\ l \in \mathcal{C}\left(\Theta_l\left(r_l\right)\right),\ \left|\Theta_l\left(r_l\right)\right| = m+1,\ A_{\Theta_l(r_l)} \neq 0, \quad (4.6)$$

exists[5].

Proof: Clearly, by (B0), κ satisfies (4.6) and by taking $r_l = \max_{l,k} d_{lk}, (l \in \Omega, k \in \kappa)$ the Lemma follows. ■

Lemma 17 provides an existence proof, but in localization in wireless sensor networks, it is important to triangulate a sensor not with the network diameter but with a small r_l. In fact, Section 4.2.4 discusses the probability of finding one such Θ_l with $r_l \ll \max_{l,k} d_{lk}, (l \in \Omega, k \in \Theta)$. In addition, Appendix B.1 provides a procedure to test the convex hull inclusion of a sensor, i.e., for any sensor, l, to determine if it lies in the convex hull of $m+1$ nodes arbitrarily chosen from the set, $\mathcal{K}\left(l, r_l\right)$, of its neighbors. Finding Θ_l is an important step in DILOC and we refer to it as *triangulation*.

To state the next assumption, define a communication link between nodes l and j, if l and j can exchange information. If l and j have a communication link, l and j can both estimate the inter-node distance, d_{lj}, between them. This distance can be found by Received Signal Strength (RSS), Time of Arrival (ToA), or Angle of Arrival (AoA), see [115] for details.

(B3) Inter-node communication. There is a communication link between all of the nodes in the set $\{l\} \cup \mathcal{K}(l, r_l),\ \forall\ l \in \Omega$.

With the above assumptions and notations, we present barycentric coordinates that serve as the updating coefficients in DILOC.

Barycentric coordinates. DILOC is expressed in terms of the barycentric coordinates, a_{lk}, of the node, $l \in \Omega$, with respect to the nodes, $k \in \Theta_l$. The barycentric coordinates, a_{lk}, are unique and are given by, see [84, 85],

$$a_{lk} = \frac{A_{\{l\} \cup \Theta_l \setminus \{k\}}}{A_{\Theta_l}}, \quad (4.7)$$

[5]Recall that the set $\mathcal{K}\left(l, r_l\right)$ groups the neighboring nodes of l within a radius r_l. By (4.6), $\Theta_l\left(r_l\right)$ is a subset of $m+1$ nodes such that sensor l lies in its convex hull but is not one of its elements.

with $A_{\Theta_l} \neq 0$, where '\' denotes the set difference, $A_{\{l\}\cup\Theta_l\setminus\{k\}}$ is the generalized volume of the set $\{l\}\cup\Theta_l\setminus\{k\}$, i.e., the set Θ_l with node l added and node k removed. The barycentric coordinates can be computed from the inter-node distances d_{lk} using the Cayley-Menger determinants as shown in Appendix B.2. From (4.7), and the facts that the generalized volumes are non-negative and

$$\sum_{k\in\Theta_l} A_{\Theta_l\cup\{l\}\setminus\{k\}} = A_{\Theta_l}, \qquad l \in \mathcal{C}(\Theta_l), \tag{4.8}$$

it follows that, for each $l \in \Omega$, $k \in \Theta_l$,

$$a_{lk} \in [0,\ 1], \qquad \sum_{k\in\Theta_l} a_{lk} = 1. \tag{4.9}$$

We now present DILOC in two steps: set-up and DILOC proper. We then provide its matrix form useful for analysis purposes.

DILOC set-up: Triangulation. In the set-up step, each sensor l triangulates itself, so that by the end of this step we have paired every $l \in \Omega$ with its corresponding $m+1$ neighbors in Θ_l. Since triangulation should be with a small r_l, the following is a practical protocol for the set-up step.

Sensor l starts with a communication radius, r_l, that guarantees triangulation with high probability with the given density of deployment, γ_0. This choice is explained in detail in Section 4.2.4. Sensor l then chooses arbitrarily $m+1$ nodes within r_l, and tests if it lies in the convex hull of these nodes using the procedure described in Appendix B.1. Sensor l attempts this with all collections of $m+1$ nodes within r_l. If all attempts fail, the sensor adaptively increases, in small increments, its communication radius, r_l, and repeats the process[6]. By **(B0)** and (4.4), success is eventually achieved and each sensor is triangulated by finding Θ_l with properties (4.6) and **(B3)**.

If a sensor has directionality a much simpler algorithm, based on Lemma 18 below (see also the discussion following the Lemma), triangulates the sensor with high

[6]The step size of this increment is also dependent on the density of the deployment, γ_0, such that a reasonable number of sensors are added in the larger neighborhood obtained by increasing r_l. This will be clear from the discussion in Section 4.2.4.

probability of success in one shot. To assess the practical implications required by DILOC's set-up phase, Subsection 4.2.4 considers the realistic scenario where the sensors are deployed using a random Poisson distribution and computes in terms of deployment parameters the probability of finding at least one such Θ_l in a given radius, r_l.

DILOC iterations: state updating. Once the set-up phase is complete, at time $t+1$, each sensor $l \in \Omega$, iteratively updates its state, i.e., its current location estimate, by a convex combination of the states at time t of the nodes in Θ_l. The anchors do not update their state, since they know their locations. The updating is explicitly given by

$$\mathbf{c}_l(t+1) = \begin{cases} \mathbf{c}_l(t), & l \in \kappa, \\ \sum_{k \in \Theta_l} a_{lk} \mathbf{c}_k(t), & l \in \Omega, \end{cases} \tag{4.10}$$

where a_{lk} are the barycentric coordinates of l with respect to $k \in \Theta_l$. DILOC in (4.10) is distributed since (i) the update is implemented at each sensor independently; (ii) at sensor $l \in \Omega$, the update of the state, $\mathbf{c}_l(t+1)$, is obtained from the states of its $m+1$ neighboring nodes in Θ_l; and (iii) there is no central location and only local information is available.

DILOC: Matrix format. For compactness of notation and analysis purposes, we write DILOC (4.10) in matrix form. Without loss of generality, we index the anchors in κ as $1, 2, \ldots, m+1$ and the sensors in Ω as $m+2, m+3, \ldots, m+1+M = N$. We now stack the (row vectors) states, $\mathbf{c}_l$, given in (4.3) for all the N nodes in the network in the $N \times m$-dimensional coordinate matrix

$$\mathbf{C} = \left[\begin{array}{c} \mathbf{c}_1^T, \ldots, \mathbf{c}_N^T \end{array} \right]^T. \tag{4.11}$$

DILOC equations in (4.10) now become in compact matrix form

$$\mathbf{C}(t+1) = \boldsymbol{\Upsilon} \mathbf{C}(t). \tag{4.12}$$

The structure of the $N \times N$ iteration matrix $\boldsymbol{\Upsilon}$ is more apparent if we partition it as

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ \mathbf{B} & \mathbf{P} \end{bmatrix}, \tag{4.13}$$

The first $m+1$ rows correspond to the update equations for the anchors in κ. Since the states of the anchors are constant, see **(B1)** and (4.5), the first $m+1$ rows of $\boldsymbol{\Upsilon}$ are zero except for a 1 at their diagonal entry $(q,q), q \in \kappa = \{1, \ldots, m+1\}$. In other words, these first $m+1$ rows are the $(m+1) \times N$ block matrix $[\mathbf{I}_{m+1}|\mathbf{0}]$, i.e., the $(m+1) \times (m+1)$ identity matrix $\mathbf{I}_{m+1}$ concatenated with the $(m+1) \times M$ zero matrix, $\mathbf{0}$.

Each of the M remaining rows in $\boldsymbol{\Upsilon}$, indexed by $l \in \Omega = \{m+2, m+3, \ldots, N\}$, have only $m+1$ non-zero elements corresponding to the nodes in the triangulation set, Θ_l, of l, and these non-zero elements are the barycentric coordinates, a_{lk}, of sensor l with respect to the nodes in Θ_l. The $M \times (m+1)$ block $\mathbf{B} = \{b_{lj}\}$ is a zero matrix, except in those entries b_{lj} corresponding to sensors $l \in \Omega$ that have a direct link to anchors $j \in \kappa$. The $M \times M$ block $\mathbf{P} = \{p_{lj}\}$ is also a sparse matrix where the non-zero entries in row l correspond to the sensors in Θ_l. The matrices $\boldsymbol{\Upsilon}$, $\mathbf{P}$, and $\mathbf{B}$ have important properties that will be used to prove the convergence of the distributed iterative algorithm DILOC in Sections 4.3 and 4.4.

Remarks: Equation (4.12) writes DILOC in matrix format for compactness; it should not be confused with a centralized algorithm–it still is a *distributed* iterative algorithm. It is iterative, because each iteration through (4.12) simply updates the (matrix of) state(s) from $\mathbf{C}(t)$ to $\mathbf{C}(t+1)$. It is distributed because each row equation updates the state of sensor l from a linear combination of the states of the $m+1$ nodes in Θ_l. In all, the iteration matrix, $\boldsymbol{\Upsilon}$, is highly sparse having exactly $(m+1)+M(m+1)$ non-zeros out of possible $(m+1+M)^2$ elements.

4.2.3 Example

To illustrate DILOC, we consider an $m = 2$-dimensional Euclidean plane with $m+1 = 3$ anchors and $M = 4$ sensors, see Fig. 4.1. The nodes are indexed such that the anchor

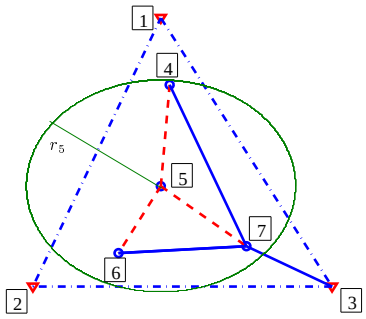


Figure 4.1: Deployment corresponding to the example in Section 4.2.3.

set is $\kappa = \{1, 2, 3\}$, $|\kappa| = m + 1 = 3$, and the sensor set is $\Omega = \{4, 5, 6, 7\}$, $|\Omega| = M = 4$. The set of all the nodes in the network is, thus, $\Theta = \kappa \cup \Omega = \{1, \ldots, 7\}$, $|\Theta| = N = 7$. The triangulation sets, $\Theta_l, l \in \Omega$, identified by using the convex hull inclusion test are $\Theta_4 = \{1, 5, 7\}$, $\Theta_5 = \{4, 6, 7\}$, $\Theta_6 = \{2, 5, 7\}$, $\Theta_7 = \{3, 5, 6\}$. These triangulation sets satisfy the properties in (4.6). Sensor 5 does not have any anchor node in its triangulation set, Θ_5, while the other sensors have only one anchor in their respective triangulation sets. Since no sensor communicates with the $m+1 = 3$ anchors directly, no sensor can localize itself in a single step.

At each sensor, $l \in \Omega$, the barycentric coordinates, a_{lk}, $k \in \Theta_l$, are computed using the inter-node distances (among the nodes in the set $\{l\} \cup \Theta_l$) in the Cayley-Menger determinant. It is noteworthy that the inter-node distances that need to be known at each sensor l to compute a_{lk} are only the inter-node distances among the $m+2$ nodes in the set $\{l\} \cup \Theta_l$. For instance, the distances in the Cayley-Menger determinant needed by sensor 5 to compute a_{54}, a_{56}, a_{57} are the inter-node distances among the nodes in the set, $\{5\} \cup \Theta_5$, i.e., $d_{54}, d_{56}, d_{57}, d_{46}, d_{47}, d_{67}$. These inter-node distances are known at sensor 5 due to **(B3)**.

Once the barycentric coordinates, a_{lk}, are computed, DILOC for the sensors in Ω is

$$\mathbf{c}_l(t+1) \;=\; \sum_{k \in \Theta_l} a_{lk} \mathbf{c}_k(t), \qquad l \in \Omega = \{4, 5, 6, 7\}. \tag{4.14}$$

We write DILOC for this example in the matrix format (4.12).

$$\begin{bmatrix} \mathbf{c}_1(t+1) \\ \mathbf{c}_2(t+1) \\ \mathbf{c}_3(t+1) \\ \mathbf{c}_4(t+1) \\ \mathbf{c}_5(t+1) \\ \mathbf{c}_6(t+1) \\ \mathbf{c}_7(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ a_{41} & 0 & 0 & 0 & a_{45} & 0 & a_{47} \\ 0 & 0 & 0 & a_{54} & 0 & a_{56} & a_{57} \\ 0 & a_{62} & 0 & 0 & a_{65} & 0 & a_{67} \\ 0 & 0 & a_{73} & a_{74} & 0 & a_{76} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c}_1(t) \\ \mathbf{c}_2(t) \\ \mathbf{c}_3(t) \\ \mathbf{c}_4(t) \\ \mathbf{c}_5(t) \\ \mathbf{c}_6(t) \\ \mathbf{c}_7(t) \end{bmatrix}, \qquad (4.15)$$

where the initial conditions are $\mathbf{C}(0) = [\mathbf{c}_1^*, \mathbf{c}_2^*, \mathbf{c}_3^*, \mathbf{c}_4^0, \mathbf{c}_5^0, \mathbf{c}_6^0, \mathbf{c}_7^0]^T$, with $\mathbf{c}_l^0,\ l \in \Omega$, being randomly chosen row vectors of appropriate dimensions. The sparseness in the matrix in (4.15) illustrates the locality of the communication among the nodes.

4.2.4 Random Poisson deployment

For sensors to determine their locations, they need to triangulate. We first consider a sufficient condition for a sensor to triangulate. We illustrate it on the plane, $m = 2$; the discussion can be extended to arbitrary dimensions. Consider Fig. 4.2(a), which shows the triangulation region, a circle of radius, r_l, centered at sensor l. Let Q_1, Q_2, Q_3, Q_4 be four disjoint sectors partitioning this circle with equal areas, i.e., $A_{Q_i} = \dfrac{\pi r_l^2}{4},\ i = 1, \ldots, 4$.

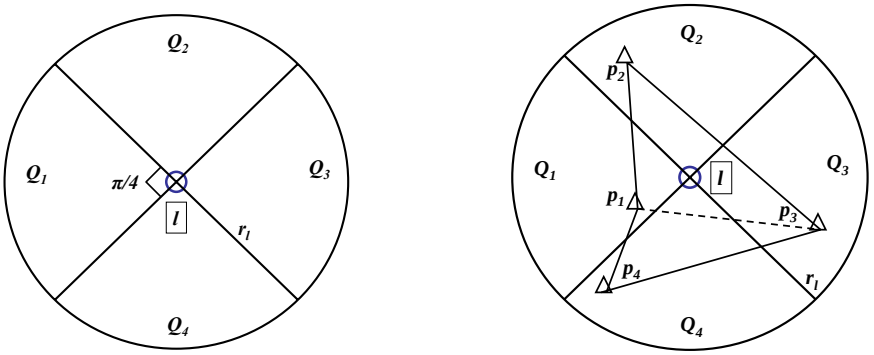
Lemma 18: A sufficient condition to triangulate sensor $l \in \mathbb{R}^2$ is to have at least one node in each of the four disjoint equal area sectors, $Q_i,\ i = 1, \ldots, 4$, which partition the circle of radius, r_l, centered at l.

Proof: In Fig. 4.2(b) consider the triangulation of sensor l located at the center of the circle; we choose arbitrarily four nodes p_1, p_2, p_3, p_4 in each of the four sectors Q_1, Q_2, Q_3, Q_4. Denote the polygon with vertices p_1, p_2, p_3, p_4 by $\text{Pol}\,(p_1p_2p_3p_4)$. Consider the diagonal[7] $p1$—$p3$ that partitions this polygon into two triangles $\triangle p_1p_2p_3$

[7]If $\text{Pol}\,(p_1p_2p_3p_4)$ is concave, we choose the diagonal that lies inside $\text{Pol}\,(p_1p_2p_3p_4)$, i.e., $p1$—$p3$ in Fig. 4.2(b). If $\text{Pol}\,(p_1p_2p_3p_4)$ is convex, we can choose any of the two diagonals and the proof follows.

and $\triangle p_1p_3p_4$. Since $l \in \text{Pol}\,(p_1p_2p_3p_4)$ and $\triangle p_1p_2p_3 \cup \triangle p_1p_3p_4 = \text{Pol}\,(p_1p_2p_3p_4)$ with $\triangle p_1p_2p_3 \cap \triangle p_1p_3p_4 = \text{LineSegment}(p1, p3)$, then either $l \in \triangle p_1p_2p_3$ or $l \in \triangle p_1p_3p_4$ or l belongs to both (when it falls on the $\text{LineSegment}(p1, p3)$). The triangle in which l lies becomes the triangulating set, Θ_l, of l.

This completes the proof. The generalization to higher dimensions is straightforward; for instance, in $\mathbb{R}^3$, we have eight sectors and an arbitrary sensor l is triangulated with at least one node in each of these eight sectors (with equal volume) of a sphere of radius, r_l, centered around l. ■



(a) (b)

Figure 4.2: (a) Sensor l identifies its triangulation set, Θ_l, in the circle of radius, r_l, centered at sensor l. The circle is divided into four disjoint sectors with equal areas, $Q_1, \ldots, Q_4$. A sufficient condition for triangulation is that there exists at least one node in each of these four sectors. (b) Illustration of Lemma 18.

In the following subsections, we study various probabilities associated with the triangulation, assuming a Poisson sensor deployment. For simplicity, we restrict the discussion to $m = 2$; it can be extended to arbitrary dimensions.

Probability that a sensor can triangulate

Here, we provide a local result concerned with the triangulation of an arbitrary sensor. We characterize the probability that a sensor l can triangulate successfully in a region of radius, r_l, centered at l. A common deployment model in wireless sensor

networks is the Poisson deployment [116, 117]. For a Poisson distribution with rate parameter or deployment density, $\gamma_0 > 0$, the mean number of nodes in a sector Q with area A_Q is $\gamma_0 A_Q$. The numbers of nodes in any two disjoint sectors, Q_1 and Q_2, are independent random variables, and the locations of the nodes in a sector Q are uniformly distributed. Let $\overline{Q}_i$ be the set of nodes in the sector Q_i. It follows from the Poisson deployment that the probability of finding at least one node in each of the four sets, $\overline{Q}_1, \ldots, \overline{Q}_4$, is the product

$$\mathbb{P}\left(|\overline{Q}_i| > 0,\, 1 \leq i \leq 4\right) = \left(1 - e^{-\gamma_0 \pi r_l^2/4}\right)^4, \tag{4.16}$$

since the distribution of the number of nodes in disjoint sectors is independent. Thus, we have

$$\begin{aligned} \mathbb{P}_{\Theta_l}(\gamma_0) &\triangleq \mathbb{P}(\{\text{sensor } l \text{ can triangulate}\}) \\ &= \mathbb{P}\left(\Theta_l \text{ exists satisfying (4.6) given } \gamma_0\right) \\ &\geq \mathbb{P}\left(|\overline{Q}_i| > 0,\, 1 \leq i \leq 4\right). \end{aligned} \tag{4.17}$$

The probability that a sensor fails to triangulate is

$$\mathbb{P}_{F,l}(\gamma_0) \triangleq \mathbb{P}(\{\text{sensor } l \text{ cannot triangulate}\}) = 1 - \mathbb{P}_{\Theta_l}(\gamma_0). \tag{4.18}$$

Equations (4.17) and (4.18) provide a tradeoff between r_l and γ_0, Indeed, to guarantee triangulation of sensor l with probability ϵ, arbitrarily close to 1, either

$$r_l \geq \left(\frac{-4\ln\left(1 - \epsilon^{\frac{1}{4}}\right)}{\gamma_0 \pi}\right)^{\frac{1}{2}} \qquad \text{or} \qquad \gamma_0 \geq \frac{-4}{\pi r_l^2}\ln\left(1 - \epsilon^{\frac{1}{4}}\right). \tag{4.19}$$

Probability that all M sensors triangulate

Here, we provide a global result, i.e., we (lower) bound the probability that all sensors in the network triangulate. We have

$$\begin{aligned}\mathbb{P}\left(\{\text{sensor } l \text{ triangulates}\},\, 1 \le l \le M\right) &= 1 - \mathbb{P}\left(\bigcup_l \{\text{sensor } l \text{ cannot triangulate}\}\right), \\ &\ge 1 - \sum_l \mathbb{P}\left(\{\text{sensor } l \text{ cannot triangulate}\}\right), \\ &= 1 - M\mathbb{P}_{F,l}(\gamma_0), \end{aligned} \tag{4.20}$$

where we use the union bound to go from the first equation to the second. To get the third equation, we use (4.18) and assume a flat network, i.e., a network where all the sensors have the same characteristics (in particular, r_l is the same for all sensors). Clearly, the bound in (4.20) is only meaningful if $\mathbb{P}_{F,l}(\gamma_0)$ is very small.

Probability that the resulting sensor network triangulates given triangulation failures

Given that some sensors may fail to triangulate, we ask the question of what is the probability that the remaining sensors can all triangulate. An exact expression is beyond the scope of this chapter. Here, we give a plausible argument when the number of sensors is large so that the laws of large numbers are valid. Since the probability of failure of each sensor to triangulate is $\mathbb{P}_{F,l}$ in (4.18), $M\mathbb{P}_{F,l}$ sensors fail to triangulate. Hence, to compute the probability that the reduced network (the network of sensors that can triangulate once we exclude the sensors that could not) triangulates, we simply repeat the steps in Subsections 4.2.4–4.2.4, but, now with M_1 sensors and deployment density γ_1, given by

$$M_1 = M(1 - \mathbb{P}_{F,l}(\gamma_0)), \tag{4.21}$$

$$\gamma_1 = \gamma_0(1 - \mathbb{P}_{F,l}(\gamma_0)), \tag{4.22}$$

From (4.20), the probability that the reduced network triangulates is

$$\begin{aligned} \mathbb{P}\left(\text{sensor } l \text{ triangulates, } \forall\, l\right) &\geq 1 - M_1 \mathbb{P}_{F,l}(\gamma_1), && (4.23) \\ &= 1 - M(1 - \mathbb{P}_{F,l}(\gamma_0))\mathbb{P}_{F,l}(\gamma_1). && (4.24) \end{aligned}$$

Equation (4.24) shows that, although not all sensors can triangulate, the probability of triangulating the reduced network can be made arbitrarily high by choosing either γ_0, or r_l, or both appropriately, such that $\mathbb{P}_{F,l}(\gamma_1) \to 0$ (or alternatively $\mathbb{P}_{\Theta_l}(\gamma_1) \to 1$).

4.2.5 Complexity of DILOC

Once the barycentric coordinates are computed, each sensor performs the update in (4.10) that requires $m+1$ multiplications and m additions. Assuming the computation complexity of the multiplication and the addition operations to be the same, the computation complexity of DILOC is $2m+1$ operations, i.e., $O(1)$ per sensor, per iteration. Since each sensor exchanges information with $m+1$ nodes in its triangulation set, the communication complexity of DILOC is $m+1$ communications, i.e., $O(1)$ per sensor, per iteration. Hence, both the computation and communication complexity are $O(M)$ for a network of M sensors. Note that, since the triangulation set-up phase[8], which identifies $\Theta_l\,(r_l)$ and computes the barycentric coordinates, as explained in Subsection 4.2.2, are to be carried out only once at the start of DILOC, they require a constant computation/communication complexity, so we do not account for it explicitly.

4.3 Convergence of DILOC

In this section, we prove the convergence of DILOC to the exact locations of the sensors, $\mathbf{c}_l^*,\ l \in \Omega$. To formally state the convergence result, we provide briefly

[8]It follows from Lemma 18 that if the sensors have directional capability then each sensor has to find one neighbor in each of its 4 sectors, $Q_{l,1}, Q_{l,2}, Q_{l,3}, Q_{l,4}$ (in $m = 2$-dimensional space). Once a neighbor is found, triangulation requires choosing 3 out of these 4, in order to identify Θ_l. The computational complexity in $m = 2$-dimensional Euclidean space is 4 choose $3 = 4$. Without directionality the process of finding Θ_l has the (expected) computation complexity of $\gamma_0 \pi r_l^2$ choose 3.

background and additional results, based on assumptions **(B0)**–**(B3)**.

The entries of the rows of the iteration matrix $\boldsymbol{\Upsilon}$, in (4.12), are either zero or the barycentric coordinates, a_{lk}, which are non-negative, and, by (4.9), add to 1. This matrix can then be interpreted as the transition matrix of a Markov chain. We then describe the localization problem and DILOC in terms of a Markov chain. Let the assumptions **(B0)**–**(B3)** in Section 4.2.2 hold and the N nodes in the sensor network correspond to the states of a Markov chain where the (ij)-th element of the iteration matrix, $\boldsymbol{\Upsilon} = \{\upsilon_{ij}\}$, defines the probability that the ith state goes to the jth state. Because of the structure of $\boldsymbol{\Upsilon}$, this chain is a very special Markov chain.

Absorbing Markov chain. Let an $N \times N$ matrix, $\boldsymbol{\Upsilon} = \{\upsilon_{ij}\}$, denote the transition probability matrix of a Markov chain with N states, $s_{i,i=1,\ldots,N}$. A state s_i is called absorbing if the probability of leaving that state is 0 (i.e., $\upsilon_{ij} = 0, i \neq j$, in other words $\upsilon_{ii} = 1$). A Markov chain is said to be absorbing if it has at least one absorbing state, and if from every state it is possible to go with non-zero probability to an absorbing state (not necessarily in one step). In an absorbing Markov chain, a state which is not absorbing is called transient. For additional background, see, for example, [71].

Lemma 19: The underlying Markov chain with the transition probability matrix given by the iteration matrix, $\boldsymbol{\Upsilon}$, is absorbing.

Proof: We prove by contradiction. Since $\upsilon_{ii} = 1$, $i \in \kappa$, the anchors are the absorbing states of the Markov chain. We now show that the Markov chain is absorbing with the sensors as the transient states.

Assume that the underlying Markov chain is not absorbing. This can happen only if the transient states can be partitioned into two disjoint clusters C1 and C2 (with C2 non-empty), such that each non-absorbing state (sensor) in C1 can reach, with non-zero probability, at least one of the absorbing states (i.e., there is a directed path from each non-absorbing state to at least one of the anchors) and, with probability 1, the states in C2 cannot reach an absorbing state (i.e., there is no directed path from the transient state to any anchor). It follows that with probability 1 that the states in C2 cannot reach the states in C1 (in one or multiple steps); otherwise, they reach an absorbing state with a non-zero probability.

Now consider the non-absorbing states (or sensors) that lie on the boundary of the convex hull, $\mathcal{C}(\text{C2})$, i.e., the vertices of $\mathcal{C}(\text{C2})$. Because they are on the boundary, they cannot lie in the interior of the convex hull of any subset of sensors in $\mathcal{C}(\text{C2})$, and, thus, cannot triangulate themselves, which contradicts Lemma 17. In order to triangulate the boundary sensors in $\mathcal{C}(\text{C2})$, the boundary sensors in C2 must be able to reach the non-absorbing states (sensors) of C1 and/or the absorbing states (anchors); that is to say that the boundary sensors in $\mathcal{C}(\text{C2})$ have directed paths to the absorbing states (anchors). This clearly contradicts the assumption that the set C2 is non-empty and, hence, every non-absorbing state has a directed path to the absorbing states. Hence, the Markov chain is absorbing and the sensors correspond to the transient states of the absorbing Markov chain. ■

Consider the partitioning of the iteration matrix, $\boldsymbol{\Upsilon}$, in (4.13). With the Markov chain interpretation, the $M \times (m+1)$ block $\mathbf{B} = \{b_{lj}\}$ is a transition probability matrix for the transient states to reach the absorbing states in one-step, and the block $M \times M$ $\mathbf{P} = \{p_{lj}\}$ is a transition probability matrix for the transient states. With (4.13), $\boldsymbol{\Upsilon}^{t+1}$ can be written as

$$\boldsymbol{\Upsilon}^{t+1} = \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ \sum_{k=0}^{t} \mathbf{P}^k \mathbf{B} & \mathbf{P}^{t+1} \end{bmatrix}, \tag{4.25}$$

and, as t goes to infinity, we have

$$\lim_{t\to\infty} \boldsymbol{\Upsilon}^{t+1} = \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ (\mathbf{I}_M - \mathbf{P})^{-1}\mathbf{B} & \mathbf{0} \end{bmatrix}, \tag{4.26}$$

by Lemma 25, in Appendix A.1. Lemma 25 uses the fact that if $\mathbf{P}$ is the matrix associated to the transient states of an absorbing Markov chain, then $\rho(\mathbf{P}) < 1$, where $\rho(\cdot)$ is the spectral radius of a matrix. With (4.26), DILOC (4.10) converges to

$$\lim_{t\to\infty} \mathbf{C}(t+1) = \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ (\mathbf{I}_M - \mathbf{P})^{-1}\mathbf{B} & \mathbf{0} \end{bmatrix} \mathbf{C}(0). \tag{4.27}$$

From (4.27) we note that the coordinates of the M sensors in Ω (last M rows of $\mathbf{C}(t+1)$) converge as $t \to \infty$ to functions of the $m+1$ anchors in κ (whose coordinates are exactly known). The limiting values of the states of the M sensors in Ω are written in terms of the coordinates of the $m+1$ anchors in κ weighted by $(\mathbf{I}_M - \mathbf{P})^{-1}\mathbf{B}$. To show that the limiting values are indeed the exact solution, we give the following Lemma.

Lemma 20: Let $\mathbf{c}_l^*$ be the exact coordinates of a node, $l \in \Theta$. Let the $M \times (m+1)$ matrix, $\mathbf{D} = \{d_{lj}\}, l \in \Omega, j \in \kappa$, be the matrix of the barycentric coordinates of the M sensors (in Ω) in terms of the $m+1$ anchors in κ, relating the coordinates of the sensors to the coordinates of the anchors by

$$\mathbf{c}_l^* = \sum_{j \in \kappa} d_{lj} \mathbf{c}_j^*, \qquad l \in \Omega. \tag{4.28}$$

Then, we have

$$\mathbf{D} \;=\; (\mathbf{I}_M - \mathbf{P})^{-1}\,\mathbf{B}. \tag{4.29}$$

Proof: Clearly $(\mathbf{I}_M - \mathbf{P})$ is invertible, since, by Lemma 25 in Appendix A.1, $\rho(\mathbf{P}) < 1$; this follows from the fact that the eigenvalues of the matrix $\mathbf{I}_M - \mathbf{P}$ are $1 - \lambda_j$, where λ_j is the jth eigenvalue of the matrix $\mathbf{P}$ and $|\lambda_j| < 1,\ \forall j = 1, \ldots, M$. It suffices to show that,

$$\mathbf{D} \;=\; \mathbf{B} + \mathbf{P}\mathbf{D}, \tag{4.30}$$

since (4.29) follows from (4.30). In (4.30), $\mathbf{D}$ and $\mathbf{B}$ are both $M \times (m+1)$ matrices, whereas $\mathbf{P}$ is an $M \times M$ matrix whose non-zero elements are the barycentric coordinates for the sensors in Ω. Hence, for the lj-th element in (4.30), we need to show that

$$d_{lj} = b_{lj} + \sum_{k \in \Omega} p_{lk} d_{kj}. \tag{4.31}$$

For an arbitrary sensor, $l \in \Omega$, its triangulation set, Θ_l, may contain nodes from both κ and Ω. We denote κ_{Θ_l} as the elements of Θ_l that are anchors, and Ω_{Θ_l} as the elements of Θ_l that are sensors, i.e., non anchors. The exact coordinates, $\mathbf{c}_l^*$, of the sensor, l, can be expressed as a convex combination of the coordinates of its neighbors

in its triangulation set, $k \in \Theta_l$, using the barycentric coordinates, a_{lk}, i.e.,

$$\begin{aligned} \mathbf{c}_l^* &= \sum_{k\in\Theta_l} a_{lk}\mathbf{c}_k^*, \\ &= \sum_{j\in\kappa_{\Theta_l}} a_{lj}\mathbf{c}_j^* + \sum_{k\in\Omega_{\Theta_l}} a_{lk}\mathbf{c}_k^*, \\ &= \sum_{j\in\kappa} b_{lj}\mathbf{c}_j^* + \sum_{k\in\Omega} p_{lk}\mathbf{c}_k^*, \end{aligned} \tag{4.32}$$

since the scalars, a_{lj}, are given by

$$a_{lj} = \begin{cases} b_{lj}, & \text{if } j \in \kappa_{\Theta_l}, \\ p_{lj}, & \text{if } j \in \Omega_{\Theta_l}, \\ 0, & \text{if } j \notin \Theta_l. \end{cases} \tag{4.33}$$

Equation (4.32) becomes, after writing each $k \in \Omega$ in terms of the $m+1$ anchors in κ,

$$\begin{aligned} \mathbf{c}_l^* &= \sum_{j\in\kappa} b_{lj}\mathbf{c}_j^* + \sum_{k\in\Omega} p_{lk} \sum_{j\in\kappa} d_{kj}\mathbf{c}_j^*, \\ &= \sum_{j\in\kappa} b_{lj}\mathbf{c}_j^* + \sum_{j\in\kappa}\sum_{k\in\Omega} p_{lk}d_{kj}\mathbf{c}_j^*, \\ &= \sum_{j\in\kappa} \left(b_{lj} + \sum_{k\in\Omega} p_{lk}d_{kj} \right) \mathbf{c}_j^*. \end{aligned} \tag{4.34}$$

This is a representation of the coordinates of sensor, l, in terms of the coordinates of the anchors, $j \in \kappa$. Since for each $j \in \kappa$, the value inside the parentheses is non-negative with their sum over $j \in \kappa$ being 1, and the fact that the barycentric representation is unique, we must have

$$d_{lj} = b_{lj} + \sum_{k\in\Omega} p_{lk}d_{kj}, \tag{4.35}$$

which, comparing to (6.53), completes the proof. ■

We now recapitulate these results in the following theorem.

Theorem 7 (DILOC convergence): DILOC (4.10) converges to the *exact* sensor

coordinates, $\mathbf{c}_l^*,\ l \in \Omega$, i.e.,

$$\lim_{t\to\infty} \mathbf{c}_l(t+1) = \mathbf{c}_l^*, \qquad \forall\ l \in \Omega. \tag{4.36}$$

Proof: The proof is a consequence of Lemmas 19 and 20. ■

Convergence rate. The convergence rate of the localization algorithm depends on the spectral radius of the matrix $\mathbf{P}$, which by Lemma 25 in Appendix A.1 is strictly less than one. In fact, as shown in Lemma 8, DILOC is characterized by geometric convergence rate with exponent $\rho(\mathbf{P})$. This is a consequence of the fact that $\mathbf{P}$ is a uniformly substochastic matrix. The convergence is slow if the spectral radius, $\rho(\mathbf{P})$, is close to 1. This can happen if the matrix $\mathbf{B}$ is close to a zero matrix, $\mathbf{0}$. This is the case if the sensors cluster in a region of very small area inside the convex hull of the anchors, and the anchors themselves are very far apart. In fact, it can be seen that in this case the barycentric coordinates for the sensors with $\kappa_{\Theta_l} \neq \varnothing$ (see Lemma 20 for this notation) corresponding to the elements in κ_{Θ_l} are close to zero. When, as in practical wireless sensor applications, the nodes are assumed to be deployed in a geometric or a Poisson fashion (see details in Section 4.2.4), the above event is highly improbable.

4.4 DILOC with relaxation

In this Section, we modify DILOC to obtain a form that is more suitable to study distributed localization in random environments. We observe that in DILOC (4.10), at time $t+1$, the expression for $\mathbf{c}_l(t+1),\ l \in \Omega$, does not involve its own coordinates, $\mathbf{c}_l(t)$, at time t. We introduce a relaxation parameter, $\alpha \in (0, 1]$, in the iterations, such that the expression of $\mathbf{c}_l(t+1)$ is a convex combination of $\mathbf{c}_l(t)$ and (4.10). We refer to this version as DILOC *with relaxation*, DILOC-REL. It is given by

$$\mathbf{c}_l(t+1) \;=\; \begin{cases} (1-\alpha)\mathbf{c}_l(t) + \alpha\mathbf{c}_l(t) = \mathbf{c}_l(t), & l \in \kappa, \\ (1-\alpha)\mathbf{c}_l(t) + \alpha \sum_{k\in\Theta_l} a_{lk}\mathbf{c}_k(t), & l \in \Omega. \end{cases} \tag{4.37}$$

DILOC is the special case of DILOC-REL with $\alpha = 1$. Clearly, DILOC-REL is also distributed as the sensor updates now have additional terms corresponding to their

own past states. The matrix representation of DILOC-REL is

$$\mathbf{C}(t+1) = \mathbf{H}\mathbf{C}(t), \tag{4.38}$$

where $\mathbf{H} = (1-\alpha)\,\mathbf{I}_N + \alpha\boldsymbol{\Upsilon}$ and $\mathbf{I}_N$ is the $N \times N$ identity matrix. It is straightforward to show that the iteration matrix, $\mathbf{H}$, corresponds to a transition probability matrix of an absorbing Markov chain, where the anchors are the absorbing states and the sensors are the transient states. Let $\mathbf{J} = (1-\alpha)\,\mathbf{I}_M + \alpha\mathbf{P}$; partitioning $\mathbf{H}$ as

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ \alpha\mathbf{B} & \mathbf{J} \end{bmatrix}. \tag{4.39}$$

We note the following

$$\mathbf{H}^{t+1} = \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ \sum_{k=0}^{t} \mathbf{J}^k \alpha\mathbf{B} & \mathbf{J}^{t+1} \end{bmatrix}, \tag{4.40}$$

$$\lim_{t\to\infty} \mathbf{H}^{t+1} = \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ (\mathbf{I}_M - \mathbf{J})^{-1}\,\alpha\mathbf{B} & \mathbf{0} \end{bmatrix}, \tag{4.41}$$

from Lemma 25. Lemma 25 applies to $\mathbf{H}$, since $\mathbf{H}$ is non-negative and $\rho(\mathbf{J}) < 1$. To show $\rho(\mathbf{J}) < 1$, we recall that $\rho(\mathbf{P}) < 1$ and the eigenvalues of $\mathbf{J}$ are $(1-\alpha) + \alpha\lambda_j$, where λ_j are the eigenvalues of $\mathbf{P}$. Therefore, we have

$$\rho(\mathbf{J}) = \max_j |(1-\alpha) + \alpha\lambda_j| < 1. \tag{4.42}$$

The following Theorem establishes convergence of DILOC-REL.

Theorem 8: DILOC-REL (4.37) converges to the *exact* sensor coordinates, $\mathbf{c}_l^*,\ l \in \Omega$, i.e.,

$$\lim_{t\to\infty} \mathbf{c}_l(t+1) = \mathbf{c}_l^*, \qquad \forall\ l \in \Omega. \tag{4.43}$$

Proof: It suffices to show that

$$(\mathbf{I}_M - \mathbf{J})^{-1}\,\alpha\mathbf{B} = (\mathbf{I}_M - \mathbf{P})^{-1}\,\mathbf{B}. \tag{4.44}$$

To this end, we note that

$$(\mathbf{I}_M - \mathbf{J})^{-1}\,\alpha\mathbf{B} \;=\; (\mathbf{I}_M - ((1-\alpha)\,\mathbf{I}_M + \alpha\mathbf{P}))^{-1}\,\alpha\mathbf{B}, \tag{4.45}$$

which reduces to (4.44) and the convergence of DILOC-REL follows from Lemma 20. ■

4.5 Enhancements to DILOC

We now generalize DILOC to the following cases.

- We assumed earlier that the iteration matrix, $\boldsymbol{\Upsilon}$, is fixed in the algorithm. This assumption corresponds to static network topology. Here, we let the network topology to be dynamic, and hence a different iteration matrix can be chosen at each time step, t, of the algorithm.

- The number of anchors can be greater than $m+1$, i.e., $|\kappa| > m+1$. We also give a different proof (from Lemma 20) for the convergence of DILOC.

- The number of neighboring sensors that a particular sensor used to express its own coordinates can be more than $m+1$.

4.5.1 Dynamic network topology

In case of dynamic network topology, each sensor, l, chooses a different triangulation set, $\Theta_l(t)$, at each iteration t of the iterative algorithm, such that the conditions in Lemma 17 hold. In this case, the coordinates of the lth sensor can be written as

$$\mathbf{c}_l(t+1) = \begin{cases} \mathbf{c}_l(t), & l \in \kappa, \\ \sum_{j\in\Theta_l(t)} a_{pj}(t)\mathbf{c}_j(t), & l \in \Omega. \end{cases} \tag{4.46}$$

The following lemma establishes the convergence of the above algorithm.

Lemma 21: The localization algorithm (4.46) with different triangulation sets at each iteration, converges to the exact sensor locations, $\mathbf{c}_l^*\ \forall\ l \in \Omega$, i.e.,

$$\lim_{t\to\infty} \mathbf{c}_l(t+1) = \mathbf{c}_l^*, \qquad l \in \Omega. \tag{4.47}$$

Proof: The resulting localization algorithm (4.46) (note that the iteration matrix, $\boldsymbol{\Upsilon}$, is now a function of time, $\boldsymbol{\Upsilon}(t)$) can be written as

$$\begin{aligned} \mathbf{C}(t+1) &= \boldsymbol{\Upsilon}(t)\mathbf{C}(t), \\ &= \prod_{j=0}^{t} \boldsymbol{\Upsilon}(j)\mathbf{C}(0). \end{aligned} \tag{4.48}$$

Consider the matrices, $\mathbf{Y}_t$, given by the product in (4.48), i.e.,

$$\mathbf{Y}_t = \prod_{j=0}^{t} \boldsymbol{\Upsilon}(j) \tag{4.49}$$

where each $\boldsymbol{\Upsilon}(j)$ can be partitioned as shown in (4.13). It follows from the structure of the matrices, $\boldsymbol{\Upsilon}(j)$, that

$$\begin{aligned} \mathbf{Y}_t &= \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ \mathbf{B}_t & \mathbf{P}_t \end{bmatrix} \cdots \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ \mathbf{B}_0 & \mathbf{P}_0 \end{bmatrix}, \\ &= \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ \mathbf{J}_t & \prod_{j=0}^{t} \mathbf{P}_j \end{bmatrix}, \end{aligned} \tag{4.50}$$

where we denote the lower left block of $\mathbf{Y}_t$ as $\mathbf{J}_t$. Let the exact coordinate matrix, $\mathbf{C}^*$, be partitioned into the exact coordinates of the anchors, $\mathbf{C}_\kappa^*$, and the exact coordinates of the sensors, $\mathbf{C}_\Omega^*$, as

$$\mathbf{C}^* = \begin{bmatrix} \mathbf{C}_\kappa^* \\ \mathbf{C}_\Omega^* \end{bmatrix}. \tag{4.51}$$

Since, $\mathbf{C}^*$ is the fixed point of each $\boldsymbol{\Upsilon}(j)$ in the product matrix, $\mathbf{Y}_t$, i.e.,

$$\begin{bmatrix} \mathbf{C}_\kappa^* \\ \mathbf{C}_\Omega^* \end{bmatrix} = \boldsymbol{\Upsilon}(j) \begin{bmatrix} \mathbf{C}_\kappa^* \\ \mathbf{C}_\Omega^* \end{bmatrix}, \qquad \forall j, \tag{4.52}$$

it follows that $\mathbf{C}^*$ is the fixed point of the product matrix, $\mathbf{Y}_t$, in (4.49). In particular, since $\mathbf{C}^*$ is the fixed point of the product matrix, $\mathbf{Y}_t$, we have

$$\mathbf{J}_t\mathbf{C}^*_\kappa + \left(\prod_{j=0}^{t}\mathbf{P}_j\right)\mathbf{C}^*_\Omega = \mathbf{C}^*_\Omega, \qquad \forall t. \tag{4.53}$$

Since the matrix $\mathbf{P}_j$ is uniformly substochastic (i.e., $\rho(\mathbf{P}_j) < 1$, which follows from the discussion after (4.26)) for $j = 0, \ldots, t$, we have

$$\lim_{t\to\infty}\prod_{j=0}^{t}\mathbf{P}_j = 0. \tag{4.54}$$

Therefore, it follows from (4.53), that $\lim_{t\to\infty}\mathbf{J}_t\mathbf{C}^*_\kappa$ exists and, in particular,

$$\lim_{t\to\infty}\mathbf{J}_t\mathbf{C}^*_\kappa = \mathbf{C}^*_\Omega. \tag{4.55}$$

Now let

$$\mathbf{C}(0) = \begin{bmatrix} \mathbf{C}^*_\kappa \\ \mathbf{C}_\Omega \end{bmatrix}, \tag{4.56}$$

be the actual initial state of the algorithm where $\mathbf{C}_\Omega$ is any arbitrary initial guess of the sensor locations. Then,

$$\begin{aligned}
\lim_{t\to\infty}\mathbf{C}(t+1) &= \lim_{t\to\infty}\mathbf{Y}_t\begin{bmatrix} \mathbf{C}^*_\kappa \\ \mathbf{C}_\Omega \end{bmatrix}, \\
&= \lim_{t\to\infty}\begin{bmatrix} \mathbf{C}^*_\kappa \\ \mathbf{J}_t\mathbf{C}^*_\kappa + \left(\prod_{j=0}^{t}\mathbf{P}_j\right)\mathbf{C}_\Omega \end{bmatrix}, \\
&= \begin{bmatrix} \mathbf{C}^*_\kappa, \\ \mathbf{C}^*_\Omega \end{bmatrix}.
\end{aligned} \tag{4.57}$$

■

Clearly, the above lemma shows that the algorithm for dynamic network topologies in(4.46) converges to the exact sensor locations.

4.5.2 More than $m+1$ anchors

In this section, we study the case where the number of anchors is greater than $m+1$, i.e., $|\kappa| = K > m+1$. This happens when, for instance, M sensors do not lie in the convex hull of $m+1$ anchors, but, lie in the convex hull of $K > m+1$ anchors. The iterative procedure has the same form as (4.10), however, the total number of sensors plus anchors becomes $K+M$. The coordinate matrix, $\mathbf{C}$, has the dimension $(K+M)\times m$ and the iteration matrix, $\boldsymbol{\Upsilon}_K$, has the dimension $(K+M)\times(K+M)$ that can be partitioned as

$$\boldsymbol{\Upsilon}_K = \begin{bmatrix} \mathbf{I}_K & \mathbf{0} \\ \mathbf{B} & \mathbf{P} \end{bmatrix}, \tag{4.58}$$

where $\mathbf{I}_K$ is a $K\times K$ identity matrix, $\mathbf{B}$ is an $M\times K$ matrix and $\mathbf{P}$ is an $M\times M$ matrix. We have the following result.

Lemma 22: The iterative localization algorithm with $K > m+1$ anchors resulting into the iteration matrix, $\boldsymbol{\Upsilon}_K$, in (4.58) converges to the exact sensor locations, $\mathbf{c}_l^*\ \forall\ l\in\Omega$, i.e.,

$$\lim_{t\to\infty} \mathbf{c}_l(t+1) = \mathbf{c}_l^*, \qquad l\in\Omega. \tag{4.59}$$

Proof: It is straightforward to show that $\mathbf{C}^*$ is the fixed point of the matrix form of (4.10) with the iteration matrix, $\boldsymbol{\Upsilon}_K$, in (4.58), we have

$$\begin{bmatrix} \mathbf{C}_\kappa^* \\ \mathbf{C}_\Omega^* \end{bmatrix} = \begin{bmatrix} \mathbf{I}_K & \mathbf{0} \\ \mathbf{B} & \mathbf{P} \end{bmatrix} \begin{bmatrix} \mathbf{C}_\kappa^* \\ \mathbf{C}_\Omega^* \end{bmatrix}, \tag{4.60}$$

$$\Rightarrow \mathbf{C}_\Omega^* = \mathbf{B}\mathbf{C}_\kappa^* + \mathbf{P}\mathbf{C}_\Omega^*, \tag{4.61}$$

which gives

$$\mathbf{C}_\Omega^* = (\mathbf{I}_M - \mathbf{P})^{-1}\,\mathbf{B}\mathbf{C}_\kappa^*. \tag{4.62}$$

Since $\mathbf{P}$ is a uniformly substochastic matrix (i.e., $\rho(\mathbf{P}) < 1$, which follows from the discussion after (4.26)), the eigenvalues of $\mathbf{P}$ lie in $[0,1)$, the eigenvalues of $\mathbf{I}_M - \mathbf{P}$ lie in $(0,1]$ and hence $\mathbf{I}_M - \mathbf{P}$ is invertible. Using Lemma 25 again, it can be shown that

$$\lim_{t\to\infty} \boldsymbol{\Upsilon}_K^{t+1} = \begin{bmatrix} \mathbf{I}_K & \mathbf{0} \\ (\mathbf{I}_M - \mathbf{P})^{-1}\mathbf{B} & \mathbf{0} \end{bmatrix}. \tag{4.63}$$

Hence the iterative algorithm converges to

$$\lim_{t\to\infty} \mathbf{C}^{t+1} = (\mathbf{I}_M - \mathbf{P})^{-1}\mathbf{B}\mathbf{C}_\kappa^* = \mathbf{C}_\Omega^*. \tag{4.64}$$

This completes the proof for the case when we have more than $m+1$ anchors. ■

The proof for $K = m+1$ anchors can be formulated as a special case of the above arguments and, hence, the above argument provides an alternative proof for DILOC in Lemma 20.

4.5.3 More than $m+1$ neighbors

Motivated by wireless sensor networks, where each sensor broadcasts its data in a communication radius and every other sensor that lies in its communication radius can receive its data, we consider the case when a sensor can have more than $m+1$ neighboring nodes. Let $\widehat{\Theta}_l$ denote the set of sensors or anchors that lie in the communication radius, r_l, of sensor l, i.e.,

$$\widehat{\Theta}_l = \{j : d_{lj} < r_l\}, \tag{4.65}$$

where d_{lj} is the Euclidean distance between node l and node j. Let $\overline{\Theta}_l = \{\Theta_l^i\} \subseteq \widehat{\Theta}_l$ be the collection of subsets of $\widehat{\Theta}_l$ such that each element, $\Theta_l^i \in \overline{\Theta}_l$, is a triangulation set for sensor l (i.e., the conditions in Lemma 17 hold). If $\overline{\Theta}_l = \varnothing$, the lth sensor increases its communication radius, r_l, until $|\overline{\Theta}_l| \geq 1$ (note that $|\varnothing| = 0$). In this fashion, each sensor can adaptively choose its communication radius, r_l, large enough such that $|\overline{\Theta}_l| \geq 1$.

The coordinates of sensor l's location can now be expressed uniquely in terms of any element, $\Theta_l^i \in \overline{\Theta}_l$, by using the barycentric coordinates. Furthermore, each sensor l can express its coordinates in terms of all the elements in $\overline{\Theta}_l$ as a convex combination of each of them, i.e.,

$$\mathbf{c}_l = \sum_i w_l^i \sum_{j\in\Theta_l^i} a_{lj}^i \mathbf{c}_j, \qquad l \in \Omega, \tag{4.66}$$

where $w_l^i \geq 0 \ \forall i$ and $\sum_i w_l^i = 1$. An iterative procedure obtained on (4.66) is

$$\mathbf{c}_l(t+1) = \begin{cases} \mathbf{c}_l(t), & l \in \kappa, \\ \sum_i w_l^i \sum_{j \in \Theta_l^i} a_{lj}^i \mathbf{c}_j(t), & l \in \Omega. \end{cases} \tag{4.67}$$

We have the following result.

Lemma 23: The distributed localization algorithm in (4.67) converges to the exact sensor locations, $\mathbf{c}_l^* \ \forall \ l \in \Omega$, i.e.,

$$\lim_{t \to \infty} \mathbf{c}_l(t+1) = \mathbf{c}_l^*, \qquad l \in \Omega. \tag{4.68}$$

Proof: The distributed localization algorithm in (4.67) can be written in matrix form as

$$\mathbf{C}(t+1) = \widetilde{\boldsymbol{\Upsilon}} \mathbf{C}(t). \tag{4.69}$$

As shown in (4.66), we again note that the way we have derived the iteration matrix, $\widetilde{\boldsymbol{\Upsilon}}$, the matrix of exact coordinates, $\mathbf{C}^*$, still remains the fixed point of the algorithm, i.e.,

$$\begin{bmatrix} \mathbf{C}_\kappa^* \\ \mathbf{C}_\Omega^* \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ \widetilde{\mathbf{B}} & \widetilde{\mathbf{P}} \end{bmatrix} \begin{bmatrix} \mathbf{C}_\kappa^* \\ \mathbf{C}_\Omega^* \end{bmatrix}, \tag{4.70}$$

$$\Rightarrow \mathbf{C}_\Omega^* = \widetilde{\mathbf{B}} \mathbf{C}_\kappa^* + \widetilde{\mathbf{P}} \mathbf{C}_\Omega^*, \tag{4.71}$$

which gives

$$\mathbf{C}_\Omega^* = \left(\mathbf{I}_M - \widetilde{\mathbf{P}} \right)^{-1} \widetilde{\mathbf{B}} \mathbf{C}_\kappa^*. \tag{4.72}$$

It is straightforward to show that $\widetilde{\mathbf{P}}$ is a uniformly substochastic matrix (i.e., $\rho(\widetilde{\mathbf{P}}) < 1$) since it is a convex combination of uniformly substochastic matrices. Using Lemma 25 again, it can be shown that

$$\lim_{t \to \infty} \widetilde{\boldsymbol{\Upsilon}}^{t+1} = \begin{bmatrix} \mathbf{I}_{m+1} & \mathbf{0} \\ (\mathbf{I}_M - \widetilde{\mathbf{P}})^{-1} \widetilde{\mathbf{B}} & \mathbf{0} \end{bmatrix}. \tag{4.73}$$

Hence, the iterative algorithm converges to

$$\lim_{t\to\infty} \mathbf{C}^{t+1} = (\mathbf{I}_M - \widetilde{\mathbf{P}})^{-1}\widetilde{\mathbf{B}}\mathbf{C}_\kappa^* = \mathbf{C}_\Omega^*. \tag{4.74}$$

■

Clearly, the above lemma shows that the localization algorithm in (4.67) with more than $m+1$ neighbors at each sensor converges to the exact sensor locations.

4.5.4 Remarks

It is a straightforward generalization to combine all of the three scenarios presented in this section. The resulting algorithm gives a comprehensive distributed localization algorithm that deals with random network topologies, any number, $K \geq m+1$, of anchors and incorporates all the sensors in the neighborhood of each sensor to achieve sensor localization.

4.6 Distributed localization in mobile networks

In this section, we present a distributed localization algorithm in m-dimensional Euclidean space, $\mathbb{R}^m$, that can be used with mobile networks. In our setup, we assume that an arbitrary number of sensors with unknown locations lie in the convex hull of at leat $m+1$ anchors that precisely know their locations and motion (for instance, they may have a GPS unit). The proposed algorithm is distributed and requires only local distance information to compute the state updates at each time when the motion has taken place.

We present a broad motion model that captures several practical scenarios of coordinated and uncoordinated motion of mobile agents. As a motivation of coordinated motion, consider the anchors moving in a specified manner such that the underlying sensor network is guided in a desired direction. An example of uncoordinated motion is the motion of cell phones that move randomly in a given fixed region (or a cell). At each time step of the motion, the network configuration changes. Our algorithm is implemented on the same time scale as that of the motion. In this section, we

present our motion model, derive conditions under which our algorithm converges, and establish minimal assumptions required for this setup.

4.6.1 Motion dynamics of mobile agents

We consider the following model for the motion dynamics of the agents in our network. Let $\mathbf{C}^*(t)$ denote the exact locations of the nodes at time t, partitioned as

$$\mathbf{C}^*(t) = \begin{bmatrix} \mathbf{C}^*_\kappa(t) \\ \mathbf{C}^*_\Omega(t) \end{bmatrix}. \tag{4.75}$$

The motion model, we consider, is as follows.

$$\mathbf{C}^*(t) \;=\; \mathcal{A}\mathbf{C}^*(t) + \mathbf{z}(t) + \mathbf{y}(t), \tag{4.76}$$

which can be partitioned into anchors and sensors as

$$\begin{bmatrix} \mathbf{C}^*_\kappa(t+1) \\ \mathbf{C}^*_\Omega(t+1) \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathcal{A}_{ux} & \mathcal{A}_{xx} \end{bmatrix} \begin{bmatrix} \mathbf{C}^*_\kappa(t) \\ \mathbf{C}^*_\Omega(t) \end{bmatrix} + \begin{bmatrix} \mathbf{z}_u(t) \\ \mathbf{z}_x(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{y}_x(t) \end{bmatrix}. \tag{4.77}$$

The $N \times N$ matrix[9] $\mathcal{A}$ relates the motion of a sensor to its neighbors such that the network may move in a coordinated fashion. The matrix $\mathbf{z}(t)$ is the deterministic drift added to the coordinates, whereas the matrix $\mathbf{y}(t)$ is the random drift with bounded norm. We state this explicitly as an assumption.

Assumption M.2.

$$\|\mathbf{y}(t)\| \leq a \qquad \mathbb{P} \text{ a.s.} \tag{4.78}$$

i.e., the norm of the random drift vector, $\mathbf{y}(t)$, is bounded above almost surely for all t.

[9]The identity matrix on the upper left block of $\mathcal{A}$ emphasizes that the anchors move independently of each other and the sensors. The motion of the anchors is captured by the deterministic drift $\mathbf{z}_u(t)$. In some problems of interest, like formation control of mobile vehicles, the anchors may be coupled among each other and the identity block can be replaced by a matrix with a specific structure, like a stochastic matrix. In that case our analysis will hold with obvious modifications in the proofs.

Although, the motion of the anchors is independent of the sensors (because of the upper right zero block in $\mathcal{A}$), the sensors move in such a way that they remain in the convex hull of the anchors. To guarantee this, we may assume that $\mathcal{A}$ is stochastic and the drift matrices are such that the sensors do not leave the convex hull of the anchors. Since the anchors know their exact locations at all time, the random drift in the anchors motions is zero, i.e., $\mathbf{y}_u(t) = \mathbf{0},\ \forall t$. We further assume that each sensor l knows the lth row of the matrices $\mathcal{A}$ and $\mathbf{z}$. The above is the general form of our motion model that we consider in this chapter. Some special cases can be derived that we elaborate below.

Uncoordinated motion in a fixed region: Consider $\mathbf{z}_u(t) = 0,\ \forall t$ and $\mathcal{A}_{ux} = \mathbf{0}, \mathcal{A}_{xx} = \mathbf{I}$. In this scenario, the anchors remain fixed and the sensors move randomly inside their convex hull. This can be thought of as the motion of wireless objects that move randomly inside a given region (or a cell). The drift at each sensor l, i.e., the lth row of $\mathbf{z}_x(t) + \mathbf{y}_x(t)$, is such that each sensor l does not leave the convex hull of the anchors.

Coordinated motion driven by anchors: Consider another scenario where $\mathbf{z}_x(t) = 0,\ \forall t$, and $\mathcal{A}_{ux} \neq \mathbf{0}, \mathcal{A}_{xx} \neq \mathbf{0}$ are such that each column of $\mathcal{A}_{ux}$ contains at least one non-zero element and the resulting $\mathcal{A}$ is a stochastic matrix. This form of a motion model is driven by anchors and the conditions on $\mathcal{A}_{ux}, \mathcal{A}_{xx}$, guarantee that the sensors move in a coordinated manner driven by the anchors.

Given the motion dynamics in eq. (4.76), we would like to estimate and track the location of each sensor in a distributed manner.

4.6.2 Algorithm and assumptions

Consider the motion model for mobile agents presented in Section 4.6.1. We now present the following algorithm for distributed localization of mobile sensors:

Algorithm MDL:

$$\mathbf{C}(t+1) = \mathbf{\Upsilon}_{t+1}\left(\mathcal{A}\mathbf{C}(t) + \begin{bmatrix} \mathbf{z_u}(t) \\ \mathbf{z_x}(t) \end{bmatrix}\right) \tag{4.79}$$

Here: $\mathbf{C}(t) = [\mathbf{C}_\kappa^T(t)\ \mathbf{C}_\Omega^T(t)]^T$ corresponds to the location estimates of the agents at time t. Note that under the assumptions of the motion model, we have

$$\mathbf{C}_\kappa(t) = \mathbf{C}_\kappa^*(t), \quad \forall t. \tag{4.80}$$

Also, the time-varying matrix $\boldsymbol{\Upsilon}_{t+1}$ denotes the matrix of local barycentric coordinates computed at time $t+1$ based on distance measurements corresponding to the network configuration $\mathbf{C}^*(t+1)$.

Some Remarks on MDL: Before proceeding to the convergence analysis of MDL, we present some discussion on the above update rule. Recall the algorithm DILOC for distributed localization in a network of static agents (Section 4.2.2), where the update is of the form:

$$\mathbf{C}(t+1) = \boldsymbol{\Upsilon}\mathbf{C}(t). \tag{4.81}$$

In that case, the network configuration (i.e., the inter-sensor distances) remains constant over time, and the matrix $\boldsymbol{\Upsilon}$ of local barycentric coordinates does not change. The exact location $\mathbf{C}^*$ is a fixed point of the linear operator $\boldsymbol{\Upsilon}$, i.e.,

$$\mathbf{C}^* = \boldsymbol{\Upsilon}\mathbf{C}^*. \tag{4.82}$$

Under the assumptions of DILOC, the operator $\boldsymbol{\Upsilon}$ has the desired contraction properties and the update rule in (4.81) converges to the unique fixed point of $\boldsymbol{\Upsilon}$, i.e., we have

$$\lim_{t\to\infty} \mathbf{C}(t) = \mathbf{C}^* \tag{4.83}$$

In the mobile network case, if $\boldsymbol{\Upsilon}_{t+1}$ is the matrix of local barycentric coordinates computed on the basis of the network configuration at time $t+1$, we have the following fixed point condition:

$$\mathbf{C}^*(t+1) = \boldsymbol{\Upsilon}_{t+1}\left(\mathcal{A}\mathbf{C}^*(t) + \begin{bmatrix} \mathbf{z_u}(t) \\ \mathbf{z_x}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{y_x}(t) \end{bmatrix}\right). \tag{4.84}$$

For a moment, assume that the unknown random perturbation $\mathbf{y_x}(t)$ is absent. In this

case, under appropriate uniform contractive properties of the linear operators $\{\boldsymbol{\Upsilon}_t\}_{t\geq 0}$ (to be detailed in the assumptions provided later), it is reasonable to expect that the MDL update should converge to the exact coordinates as $t \to \infty$, i.e.,

$$\lim_{t\to\infty} \|\mathbf{C}(t) - \mathbf{C}^*(t)\| = 0 \tag{4.85}$$

This is the key intuition behind the MDL update rule. In the general case, when random perturbations are present and are unpredictable, the update takes the form of (4.79). In this case, we expect a steady state convergence error, i.e.,

$$\limsup_{t\to\infty} \|\mathbf{C}(t) - \mathbf{C}^*(t)\| \leq \mathbf{e}^* \tag{4.86}$$

where the steady-state error depends on the distribution of the random perturbation, as shown later.

We now present the key assumption on network connectivity and triangulation at time t, required for establishing desired convergence properties of the MDL algorithm:

Assumption M.1 For every t, define the matrix $\mathbf{H}_t$ by:

$$\mathbf{H}_t = \mathbf{P}_t \mathcal{A}_{xx}. \tag{4.87}$$

Here, $\mathbf{P}_t$ is the block coming from the natural decomposition of the matrix $\boldsymbol{\Upsilon}_t$ as:

$$\boldsymbol{\Upsilon}_t = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}_t & \mathbf{P}_t \end{bmatrix}. \tag{4.88}$$

We make the following assumption on network connectivity and triangulation:

There exists $0 < \varepsilon < 1$, such that,

$$\mathbb{P}\left(\|\mathbf{H}_t\| \leq \varepsilon, \quad \forall t\right) = 1 \tag{4.89}$$

(note that the matrices $\mathbf{H}_t$ are now random, because of the random perturbations $\mathbf{y}_x(t)$ that affect the network configuration and thus $\mathbf{H}_t$.)

Since, $\|\mathcal{A}_{xx}\| \leq 1$ (being a sub-block of a stochastic matrix, $\mathcal{A}$), a sufficient condition for **M.1** to hold is:

$$\mathbb{P}\left(\|\mathbf{P}_t\| \leq \varepsilon, \quad \forall t\right) = 1. \tag{4.90}$$

The fact, that **M.1** is reasonable, is demonstrated from the fact, that, in the static case, under minimal assumptions on network connectivity and triangulation, we have $\|\mathbf{P}\| \leq \varepsilon$ for some $\varepsilon < 1$. In the dynamic case, assuming the network is sufficiently dense or the motion is coordinated, the network structure does not change drastically over time and hence Assumption **M.1**, which is a uniformity condition on the relative network structure, is reasonable.

4.6.3 Convergence analysis of MDL

In this section, we present the convergence analysis of the algorithm MDL with the general motion model discussed in Section 4.6.1 and Assumption **M.1-M.2** (recall that **M.2** is provided in Section 4.6.1).

Theorem 9: Consider the dynamic sensor motion model with $\mathbf{C}^*(t)$ denoting the sensor configuration at time t as given in (4.76). Let $\mathbf{C}(t)$ be the sensor state estimates generated by the distributed localization algorithm MDL in (4.79). We then have

$$\mathbb{P}\left(\limsup_{t\to\infty} \|\mathbf{C}(t) - \mathbf{C}^*\| \leq \frac{a}{1-\varepsilon}\right) = 1 \tag{4.91}$$

Proof: By construction of the local barycentric coordinates, it follows that for all t (recall eqn. (4.84)),

$$\mathbf{C}^*(t+1) = \boldsymbol{\Upsilon}_{t+1} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathcal{A}_{ux} & \mathcal{A}_{xx} \end{bmatrix} \mathbf{C}(t)^* + \boldsymbol{\Upsilon}_{t+1} \begin{bmatrix} \mathbf{z_u}(t) \\ \mathbf{z_x}(t) \end{bmatrix} + \boldsymbol{\Upsilon}_{t+1} \begin{bmatrix} \mathbf{0} \\ \mathbf{y_x}(t) \end{bmatrix}. \tag{4.92}$$

Subtracting this from the MDL state update

$$\mathbf{C}(t+1) = \boldsymbol{\Upsilon}_{t+1} \left(\mathcal{A}\mathbf{C}(t) + \begin{bmatrix} \mathbf{z_u}(t) \\ \mathbf{z_x}(t) \end{bmatrix} \right) \tag{4.93}$$

we have

$$\mathbf{e}(t+1) = \boldsymbol{\Upsilon}_{t+1} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathcal{A}_{ux} & \mathcal{A}_{xx} \end{bmatrix} \mathbf{e}(t) - \boldsymbol{\Upsilon}_{t+1} \begin{bmatrix} \mathbf{0} \\ \mathbf{y}_{\mathbf{x}}(t) \end{bmatrix} \tag{4.94}$$

where:

$$\mathbf{e}(t) = \mathbf{C}(t) - \mathbf{C}^*(t) \tag{4.95}$$

is the matrix of location estimation errors at time t. Decomposing $\mathbf{e}(t)$ into sensors and anchors, we have

$$\mathbf{e}(t) = \begin{bmatrix} \mathbf{e}_\kappa(t) \\ \mathbf{e}_\Omega(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_\Omega(t) \end{bmatrix} \tag{4.96}$$

The fact, that $\mathbf{e}_\kappa(t) = 0$, follows from $\mathbf{C}_\kappa(t) = \mathbf{C}^*_\kappa(t)$, as the anchors know their locations exactly at all time t.

Multiplying out the various terms in eqn. (4.94) we then have the following update rule for $\mathbf{e}(t)$:

$$\mathbf{e}_\Omega(t+1) = \mathbf{H}_{t+1}\mathbf{e}_\Omega(t) - \mathbf{P}_{t+1}\mathbf{y}_{\mathbf{x}}(t), \quad \forall t \tag{4.97}$$

where $\mathbf{H}_t$ is defined in eqn. (4.87). Continuing the recursion for $\mathbf{e}_\Omega(t)$, we have

$$\mathbf{e}_\Omega(t) = \left(\prod_{k=1}^{t} \mathbf{H}_k \right) \mathbf{e}_\Omega(0) - \sum_{k=0}^{t-1} \left(\prod_{j=t-k+2}^{t} \mathbf{H}_j \right) \mathbf{P}_{t-k+1}\mathbf{y}_{\mathbf{x}}(t-k). \tag{4.98}$$

We thus have

$$\|\mathbf{e}_\Omega(t)\| \leq \|\mathbf{e}_\Omega(0)\| \prod_{k=1}^{t} \|\mathbf{H}_k\| + \sum_{k=0}^{t-1} \|\mathbf{P}_{t-k+1}\| \, \|\mathbf{y}_{\mathbf{x}}(t-k)\| \left(\prod_{j=t-k+2}^{t} \|\mathbf{H}_j\| \right). \tag{4.99}$$

Under Assumptions **M.1, M.2**, we have for all t

$$\|\mathbf{H}_t\| \leq \varepsilon, \quad \mathbb{P} \text{ a.s.} \tag{4.100}$$

$$\|\mathbf{y}_{\mathbf{x}}(t)\| \leq a, \quad \mathbb{P} \text{ a.s.} \tag{4.101}$$

Also, by construction, the matrix $\mathbf{P}_t$ is substochastic for all t and hence

$$\|\mathbf{P}_t\| \leq 1, \quad \mathbb{P} \text{ a.s.} \tag{4.102}$$

The following then holds $\mathbb{P}$ a.s. from (4.99)

$$\|\mathbf{e}_\Omega(t)\| \leq \varepsilon^t \|\mathbf{e}_\Omega(0)\| + a \sum_{k=0}^{t-1} \varepsilon^k \tag{4.103}$$

Taking the limit as $t \to \infty$ and noting that $0 < \varepsilon < 1$, we have $\mathbb{P}$ a.s.

$$\limsup_{t\to\infty} \|\mathbf{e}_\Omega(t)\| \leq \frac{a}{1-\varepsilon} \tag{4.104}$$

The result then follows from (4.104). ■

Remarks: The convergence results are established under minimal conditions on network connectivity and triangulation, embedded in the Assumption **M.1**, i.e.,

$$\mathbb{P}\left(\|\mathbf{H}_t\| \leq \varepsilon, \quad \forall t\right) = 1, \tag{4.105}$$

for some $\varepsilon < 1$. The problem of finding the right value of ε is also of interest and conveys significant information on the convergence rate of the algorithm. Such a characterization of ε depends on the specifics of the motion model, for example, the geometry of sensor deployment, the various model matrices and the distribution of the random drift. This is an interesting problem in its own right and we intend to investigate this in the future.

In the absence of the random drift (but with non-zero deterministic drift), the assumption **M.1** on network connectivity and local triangulation at all times can be relaxed, and we may work with the much weaker assumption of successful local triangulation infinitely often (i.o.), i.e.,

$$\mathbb{P}\left(\|\mathbf{H}_t\| \leq \varepsilon, \quad \text{i.o. } t\right) = 1 \tag{4.106}$$

for some $\varepsilon < 1$.

4.7 Localization in random environments

We discussed the random environments in the context of the HDC in Section 3.5. In this section, we translate the random phenomena to the sensor localization problem. Note that the assumptions on the *data packet drops* and the *communication noise* hold for the sensor localization problem. Below, we explain the significance of *small perturbation of system matrices* in the context of sensor localization.

Revisiting: Small perturbation of system matrices: Recall from **(B3)** in Section 4.2.2 that, at each sensor l, the distances required to compute the barycentric coordinates are the inter-node distances in the set

$$\mathcal{D}_l = \{l\} \cup \mathcal{K}(l, r_l). \tag{4.107}$$

In reality, the sensors do not know the precise distances, $d^*_{(\cdot)(\cdot)}$, but estimate the distances, $\widehat{d}_{(\cdot)(\cdot)}(t)$, from the Received-Signal-Strength (RSS) or Time-of-Arrival (TOA) measurements at time t. When we have noise on distance measurements, we cannot iterate with the system matrices $\mathbf{P}(\mathbf{d}^*)$ and $\mathbf{B}(\mathbf{d}^*)$, but iterate with $\mathbf{P}(\widehat{\mathbf{d}}_t)$ and $\mathbf{B}(\widehat{\mathbf{d}}_t)$, where the collection of all the required inter-node distance estimates is

$$\widehat{\mathbf{d}}_t = \{\widehat{d}_{kn}(t) \mid k, n \in \mathcal{D}_l, l \in \Omega\}. \tag{4.108}$$

Assuming the distance measurements are statistically independent over time, $\mathbf{P}(\widehat{\mathbf{d}}_t)$ and $\mathbf{B}(\widehat{\mathbf{d}}_t)$ can be written as

$$\mathbf{P}\left(\widehat{\mathbf{d}}_t\right) = \mathbf{P}(\mathbf{d}^*) + \mathbf{S}_\mathbf{P} + \widetilde{\mathbf{S}}_\mathbf{P}(t) \triangleq \{\widehat{p}_{ln}(t)\}, \quad \mathbf{B}\left(\widehat{\mathbf{d}}_t\right) = \mathbf{B}(\mathbf{d}^*) + \mathbf{S}_\mathbf{B} + \widetilde{\mathbf{S}}_\mathbf{B}(t) \triangleq \{\widehat{b}_{ln}(t)\}, \tag{4.109}$$

where $\mathbf{S}_\mathbf{P}$ and $\mathbf{S}_\mathbf{B}$ are mean measurement errors, and $\{\widetilde{\mathbf{S}}_\mathbf{P}(t)\}_{t\geq 0}$ and $\{\widetilde{\mathbf{S}}_\mathbf{B}(t)\}_{t\geq 0}$ are independent sequence of random matrices with zero-mean and finite second moments. In particular, even if the distance estimates are unbiased, the computed $\mathbf{P}(\widehat{\mathbf{d}}_t)$ and $\mathbf{B}(\widehat{\mathbf{d}}_t)$ may have non-zero biases, $\mathbf{S}_\mathbf{P}$ and $\mathbf{S}_\mathbf{B}$, respectively. Now, we employ the modified HDC algorithm, presented in Section 3.5, to establish the convergence to

the sensor locations described in Theorem 4. Clearly, if $\mathbf{S_P} = \mathbf{S_B} = \mathbf{0}$, then the sensors converge to the exact sensor locations. In the other case ($\mathbf{S_P} \neq \mathbf{0}, \mathbf{S_B} \neq \mathbf{0}$), there is a steady-state error.

We now present DILAND (distributed localization algorithm with noisy distances) that converges a.s. to the exact sensor locations, even if $\mathbf{S_P} \neq \mathbf{0}, \mathbf{S_B} \neq \mathbf{0}$. As mentioned above, since, at time t, we use only the current RSS or TOA measurements to compute distance estimates, the resulting system matrices have constant error bias, i.e., $\mathbf{S_P} \neq \mathbf{0}$ and $\mathbf{S_B} \neq \mathbf{0}$. Clearly, a more accurate scheme is to utilize the information from past distance measurements also, so that one computes the system matrices at time t as a function of the entire past measurements, $\{RSS_s\}_{s \leq t}$ or $\{TOA_s\}_{s \leq t}$. The DILAND algorithm efficiently utilizes the past information and, as will be shown, leads to a.s. convergence to the exact sensor locations under practical distance measurement schemes in sensor networks. To this aim, we explore two standard distance measurement methods in sensor networks, namely, the Received Signal Strength (RSS) and the Time-of-Arrival (TOA). For the remainder of this subsection, we borrow experimental and theoretical results from [115].

4.7.1 Received signal strength (RSS)

In wireless settings, the signal power decays with a path-loss exponent, n_p, which depends on the environment. If sensor a sends a packet to sensor b, then RSS0 is the power of the signal received by sensor b. Based on the RSS measurement, RSS_{ab}, the maximum likelihood estimator, $\widehat{d}_{ab}$, of the distance, d_{ab}, between sensors a and sensor b is [115]

$$\widehat{d}_{ab} = \Delta_0 10^{\frac{\Pi_0 - \mathrm{RSS}_{ab}}{10 n_p}}, \tag{4.110}$$

where Π_0 is the received power at a short reference distance Δ_0. For this estimate,

$$\mathbb{E}\left[\widehat{d}_{ab}\right] = C d_{ab}, \tag{4.111}$$

where C is a multiplicative bias factor. Based on calibration experiments and a priori knowledge of the environment, we can obtain precise estimates of C; for typical

channels, $C \approx 1.2$ [118] and hence scaling (4.110) by C gives us an unbiased estimate. If the estimate of C is not acceptable, we can employ the following.

DILAND is iterative and data is exchanged at each iteration t. We then have the measurements on RSS and C at each iteration. Hence, the distance estimate we employ is

$$\widetilde{d}_{ab}(t) = \frac{\widehat{d}_{ab}(t)}{\widehat{C}(t)}, \tag{4.112}$$

where $\widehat{C}(t)$ is the estimate of C (for details on this estimate, see [115] and references therein) at the t-th iteration of DILAND. Assuming that $\widehat{d}_{ab}(t)$ and $\widehat{C}(t)$ are statistically independent (which is a reasonable assumption if we use different measurements for both of these estimates and assuming that the measurement noise is independent over time), we have

$$\mathbb{E}\left[\widetilde{d}_{ab}(t)\right] = d_{ab}. \tag{4.113}$$

Since at time t, we have knowledge of $\{\widetilde{d}_{ab}(s)\}_{0\leq s\leq t}$, we can use the following sequence, $\{\overline{d}_{ab}(t)\}_{t\geq 0}$, of distance estimates to compute the barycentric coordinates at time t:

$$\overline{d}_{ab}(t) = \frac{1}{t}\sum_{s\leq t}\widetilde{d}_{ab}(s) = \frac{t-1}{t}\overline{d}_{ab}(t-1) + \frac{1}{t}\widetilde{d}_{ab}(t), \qquad \overline{d}_{ab}(0) = \widetilde{d}_{ab}(0). \tag{4.114}$$

Then from (4.112)-(4.113) and the strong law of large numbers, we have $\overline{d}_{ab}(t) \to d_{ab}$ a.s. as $t \to \infty$.

4.7.2 Time-of-arrival (TOA)

Time-of-arrival is also used in wireless settings to estimate distances. TOA is the time for a signal to propagate from sensor a to sensor b. To get the distance, TOA is multiplied by ν_p, the propagation velocity. Over short ranges, the measured time delay, T_{ab}, can be modeled as a Gaussian distribution[10] [115] with mean $d_{ab}/\nu_p + \mu_T$ and variance σ_T^2. The distance estimate is given by $\widehat{d}_{ab} = (T_{ab} - \mu_T)\nu_p$. Based on

[10] Although, as noted before, DILAND does not require any distributional assumption.

calibration experiments and a priori knowledge of the environment, we can obtain precise estimates of μ_T; Wideband DS-SS measurements [88] have shown $\mu_T = 10.9$ns.

Since DILAND is iterative; we can make the required measurements at each iteration t and also compute an estimate, $\widehat{\mu}_T(t)$, of the bias factor, μ_T (for details on this computation, see [88]). Then, using the same procedure described for RSS measurements, we can obtain a sequence, $\{\overline{d}_{ab}(t)\}_{t\geq 0}$, of distance estimates such that $\overline{d}_{ab}(t) \to d_{ab}$ a.s. as $t \to \infty$.

In both of the above cases, we note that, if $\{Z(t)\}_{t\geq 0}$ is a sequence of distance measurements, where $Z = RSS$ or $Z = TOA$, collected over time, then there exist estimates $\overline{\mathbf{d}}_t$ with the property: $\overline{\mathbf{d}}_t \to \mathbf{d}^*$ a.s. as $t \to \infty$. In other words, by efficiently (for example, simple averaging) using past distance information, we can estimate the required inter-sensor distances to arbitrary precision as $t \to \infty$. This leads to the following natural assumption:

$\overline{\textbf{(D1)}}$ **Noisy distance measurements:** Let $\{Z(t)\}_{t\geq 0}$ be any sequence of inter-node distance measurements collected over time. Then, there exists a sequence of estimates $\{\overline{\mathbf{d}}_t\}_{t\geq 0}$, such that, for all t, $\overline{\mathbf{d}}_t$ can be computed *efficiently* from $\{X(s)\}_{s\leq t}$ and we have

$$\mathbb{P}\left[\lim_{t\to\infty} \overline{\mathbf{d}}_t = \mathbf{d}^*\right] = 1 \tag{4.115}$$

We now present the algorithm DILAND under the modified assumption $\overline{\textbf{(D1)}}$.

4.7.3 Algorithm

For clarity of presentation, we analyze the DILAND algorithm only in the context of noisy distance measurements and assume inter-sensor communication is perfect (i.e., no link failures and communication noise[11].) Let $\mathbf{P}(\overline{\mathbf{d}}_t) \triangleq \{\overline{p}_{ln}(t)\}$ and $\mathbf{B}(\overline{\mathbf{d}}_t) \triangleq \{\overline{b}_{lk}(t)\}$ be the matrices of barycentric coordinates computed at time t from the distance estimate $\overline{\mathbf{d}}_t$.

[11] The modifications under these phenomena is straightforward as discussed in Section 3.5.

The DILAND algorithm updates the location estimates as follows:

$$\begin{aligned} x_l^j(t+1) &= (1-\alpha(t))x_l^j(t) \\ &+ \alpha(t)\left[\sum_{n\in\mathcal{K}_\Omega(l)} \overline{p}_{ln}(t)x_n^j(t) + \sum_{k\in\mathcal{K}_\kappa(l)} \overline{b}_{lk}(t)u_k^j\right], \end{aligned} \tag{4.116}$$

for $l \in \Omega,\ 1 \le j \le m$, where the weight sequence, $\alpha(t)$, satisfies

$$\alpha(t) \ge 0, \qquad \lim_{t\to\infty} \alpha(t) = 0, \qquad \text{and} \qquad \sum_t \alpha(t) = \infty. \tag{4.117}$$

In particular, here we consider the following choice: for $a > 0$ and $0 < \delta \le 1$,

$$\alpha(t) = \frac{a}{(t+1)^\delta}. \tag{4.118}$$

The update (4.116) is followed by a distance update of the form (4.114). The following result gives the convergence properties of DILAND.

Theorem 10: Let $\{\mathbf{x}(t)\}_{t\ge 0}$ be the state sequence generated by DILAND (4.116) under the standard DILOC assumptions **(B0)-(B3)** and $\overline{(\mathbf{D1})}$. Then, for every l, $\mathbf{x}_l(t)$ converges a.s. to the exact location of sensor l as $t \to \infty$, i.e.,

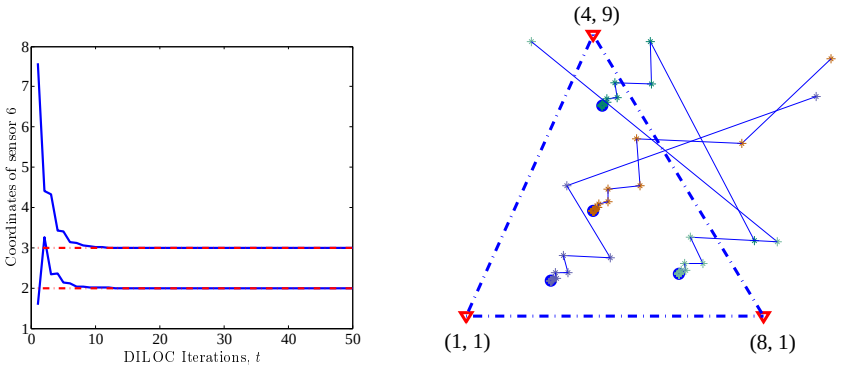
$$\mathbb{P}\left[\lim_{t\to\infty} \mathbf{x}^j(t) = \left(\mathbf{I} - \mathbf{P}(\mathbf{d}^*)\right)^{-1}\mathbf{B}(\mathbf{d}^*)\mathbf{u}^j, \ \ \forall j = 1,\dots,m\right] = 1, \tag{4.119}$$

which are the exact sensor locations.

Proof: A detailed proof is provided in [44]. ■

4.8 Simulations

In this section, we present numerical experiments for DILOC, its enhancements, and localization for the mobile agents.



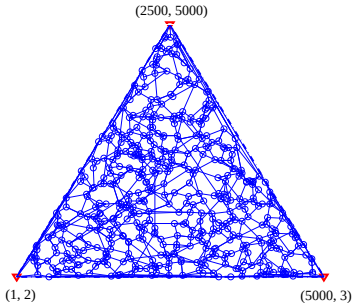
(a) (b)

Figure 4.3: Deterministic environment: (a) Estimated coordinates of sensor 6 in Section 4.2.3 as a function of DILOC iterations. (b)Trajectories of the sensors' estimates obtained by DILOC.

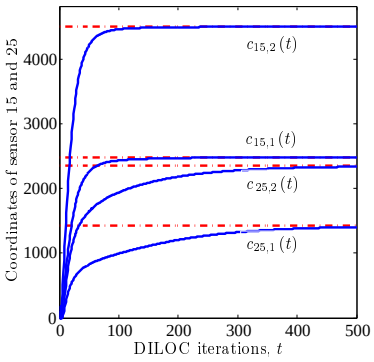
4.8.1 DILOC

We consider the example presented in Section 4.2.3, that has $m + 1 = 3$ anchors and $M = 4$ sensors with no noise in the system. DILOC, as given in (4.14), is implemented, where Fig. 4.3(a) shows the estimated coordinates of sensor 6, and Fig. 4.3(b) shows the trajectories of the estimated coordinates for all the sensors with random initial conditions. Both figures show fast convergence to the exact sensor locations, which should be the case because of the geometric convergence rate.

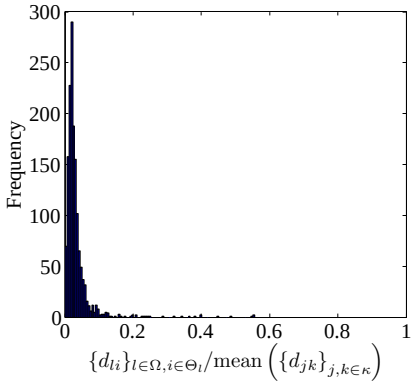
We further consider a network of $N = 500$ nodes shown in Fig. 4.4(a) after triangulation. The communication radius is increased until each sensor triangulates. DILOC is implemented with zero initial conditions and Fig. 4.4(b) shows the estimated coordinates of two arbitrary sensors; this illustrates that geometric convergence of DILOC estimates to the exact sensor locations. Fig. 4.4(c) shows a typical histogram of the inter-node distances (normalized by the mean of all anchor-anchor distances) over which the DILOC communications are implemented. It can be verified that the 95th percentile of the inter-node distances are within 10% of the mean anchor-anchor distance.



(a)



(b)



(c)

Figure 4.4: Deterministic environment: (a) An $N = 500$ node network and the respective triangulation sets. (b) Estimated coordinates of two arbitrarily chosen sensors as a function of DILOC iterations. (c) Histogram of normalized inter-node distances over which the DILOC communications are implemented.

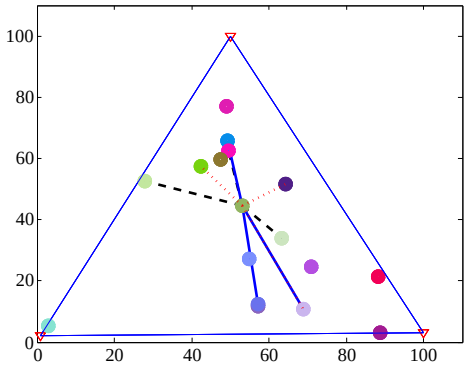


Figure 4.5: For a fixed sensor, its 3 different neighborhoods are shown.

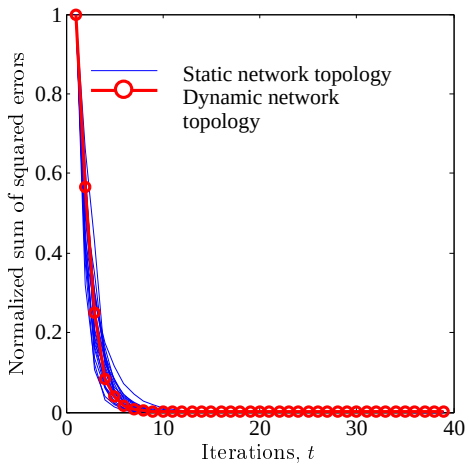


Figure 4.6: Performance comparison of the dynamic scheme with $T = 20$ static (fixed) topologies.

4.8.2 Dynamic network topology

We simulate an $N = 20$ node network in $m = 2-$Euclidean space, where we have $K = m + 1 = 3$ anchors (with known locations) and $M = 17$ sensors (with unknown locations). We formulate $T = 20$ different iteration matrices, $\boldsymbol{\Upsilon}(j)$, where $j = 1, \ldots, T$ and at each iteration, t, of the algorithm we randomly choose one out of the T iteration matrices. The dynamical network topology for a particular sensor is shown in Fig. 4.5 where 3 different neighborhoods are shown. Fig. 4.6 compares the performance of the dynamic scheme with $T = 20$ static networks.

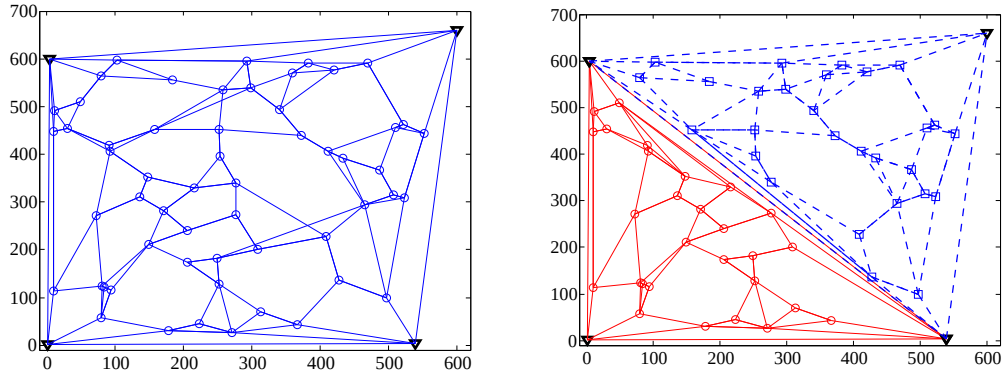


Figure 4.7: (a) The overall sensor network with $K = 4 > m + 1$ anchors such that $\mathcal{C}(\Omega) \subset \mathcal{C}(\kappa)$. (b) Dividing the overall network into two subproblems where we have $m + 1 = 3$ anchors for each of the subproblems.

4.8.3 More than $m + 1$ anchors

We simulate an $N = 60$ node network in $m = 2-$Euclidean space, where we have $K = 4 > m + 1$ anchors (with known locations) and $M = 56$ sensors (with unknown locations). Fig. 4.7(a) shows the overall network where the $M = 56$ sensors lie in the convex hull of $K = 4$ anchors. Fig. 4.7(b) divides the original problem into two subproblems, each of which is solving the unknown sensors with $m + 1 = 3$ anchors. Fig. 4.8 compares the combined performance of the two subproblems with the performance of the scheme where we used more anchors.

4.8.4 More than $m + 1$ neighbors

We simulate an $n = 20$ node network in $m = 2-$Euclidean space, where we have $K = m + 1 = 3$ anchors (with known locations) and $M = 17$ sensors (with unknown locations). The neighborhood, Θ_l^i, of the lth sensor is chosen adaptively by increasing the communication radius, R_l, of the lth sensor as discussed in Section 4.5.3. Adaptive choice of the communication radius is shown in Fig. 4.9(a) for three arbitrarily chosen sensors. Fig. 4.9(b) shows the resulting communication network where it can be

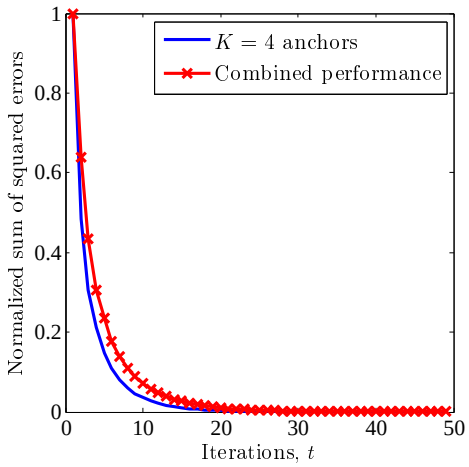


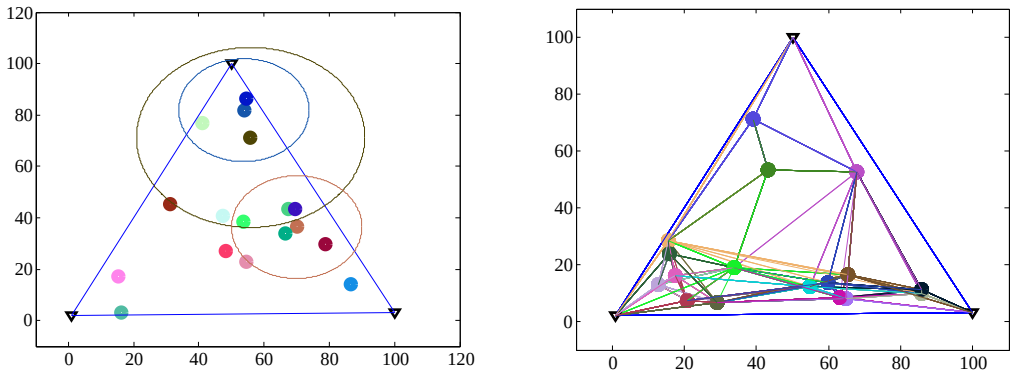
Figure 4.8: Performance comparison between the aggregated performance of the two subproblems and the scheme with $K = 4$ anchors.

verified that each sensor is now connected to more than $m+1$ neighbors. Performance comparison of fixed $m + 1$ neighbors for each $T = 20$ different neighborhoods with the scheme where all the neighborhoods are combined using a weighting sequence is shown in Fig. 4.10. The combining weights are chosen to be $1/N_l$, where $N_l = |\overline{\Theta_l}|$.

4.8.5 Localization of mobile agents

We now present numerical simulations for the MDL algorithm. Consider a network of $N = 50$ nodes in $\mathbb{R}^2$ (plane), where we have $m + 1 = 3$ anchors and $M = 47$ sensors. The sensors lie in the convex hull of the anchors. We assume a coordinated motion model on the sensors. In particular, we choose $\mathcal{A}_{ux}$ and $\mathcal{A}_{xx}$ such that $\mathcal{A}$ is a stochastic matrix and there is no drift on the sensors' motion, i.e., $\mathbf{z}_x = \mathbf{0}$. The anchors move with a known drift and the sensors move as a convex combination of their neighbors' movement. This guarantees that the sensors remain in the convex hull.

Fig. 4.12(a) shows the motion of the anchors and two randomly chosen sensors. Fig. 4.11(a) and fig. 4.11(b) show the horizontal and vertical coordinates of two randomly chosen sensors as solid lines. The estimated coordinates with the MDL algorithm are plotted as dashed lines. The initial condition of the algorithm are set to



(a) (b)

Figure 4.9: (a) Adaptively choosing the communication radius, R_l shown for three arbitrarily chosen sensors. (b) Resulting network where each sensor is connected to more than $m + 1 = 3$ neighbors.

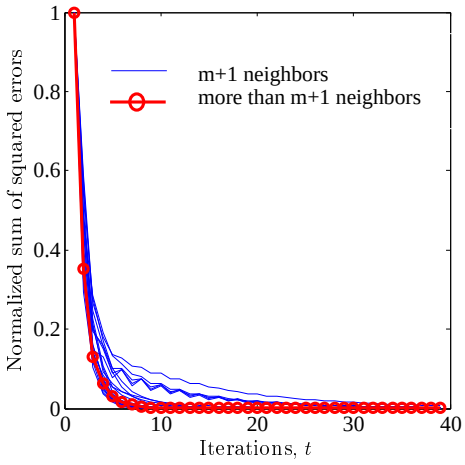
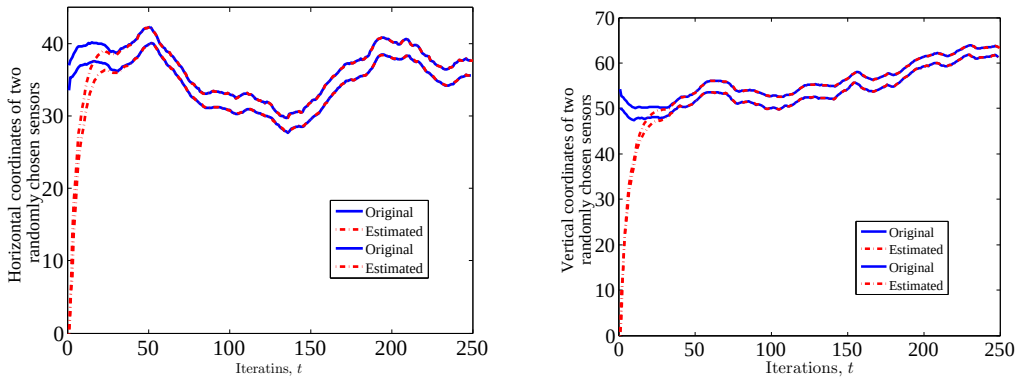


Figure 4.10: Performance comparison of the fixed $m+1$ neighbors with more than $m+1$ neighbors.



(a) (b)

Figure 4.11: Coordinated motion with deterministic drift: (a) The horizontal and (b) vertical coordinates of two randomly chosen sensors and the MDL estimates.

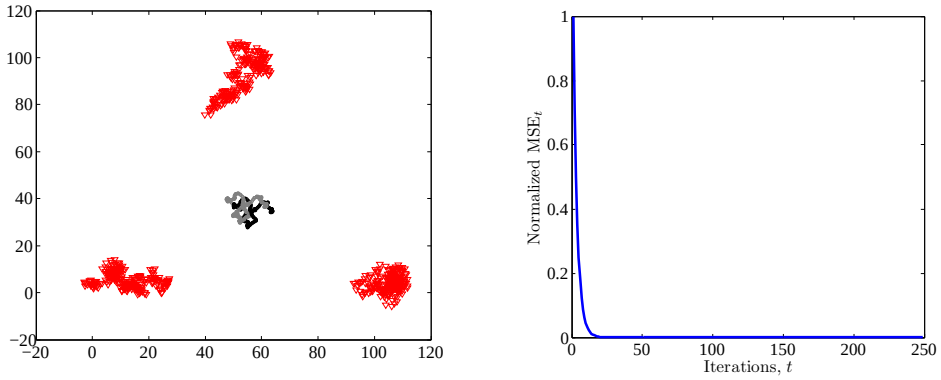
zero. Notice that MDL catches up with the motion and then tracks the motion of the sensors. Fig. 4.12(b) shows the normalized mean squared error, i.e.,

$$\text{MSE}_t = \frac{1}{M}\sum_{l=1}^{M}\sum_{i=1}^{m}\left(\mathbf{c}_{l,i}(t) - \mathbf{c}^*_{l,i}\right)^2, \qquad l \in \Omega, \tag{4.120}$$

in the estimated coordinates.

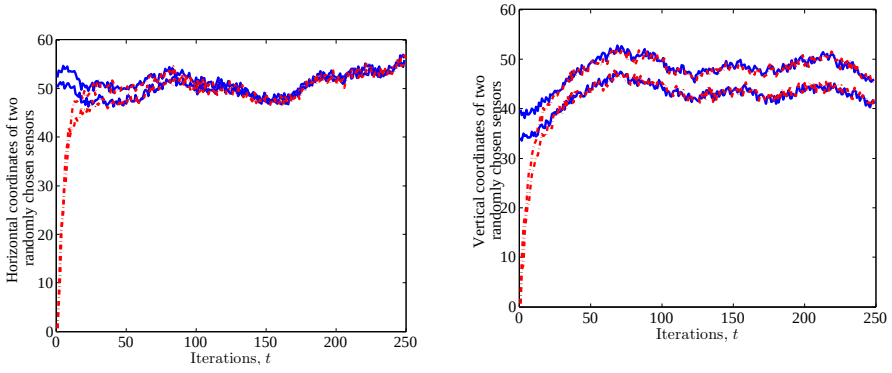
In the next experiment, we consider the same model as before but introduce a random drift to the sensors motion. In particular, we choose a uniform random variable on the interval $[-2\ 2]$ as random drift in each coordinate at each sensor. For a network of $N = 50$ nodes, we show the resulting estimates as dashed lines against the original coordinates as solid lines in Fig. 4.13(a) and Fig. 4.13(b) for two randomly chosen sensors. Fig. 4.14(a) shows the motion of the anchors and these sensors. We then plot the normalized mean squared error in Fig. 4.14(b). Notice that the estimates catch up the exact sensor coordinates but there is an error due to the random drift. We use log scale to show the steady state error.

Finally, we present a simulation for uncoordinated motion in a fixed region. We choose $\mathcal{A}= \mathbf{I}$ and choose zero known drift, i.e., $\mathbf{z} = \mathbf{0}$. We choose $\mathbf{y}_x$ such that each



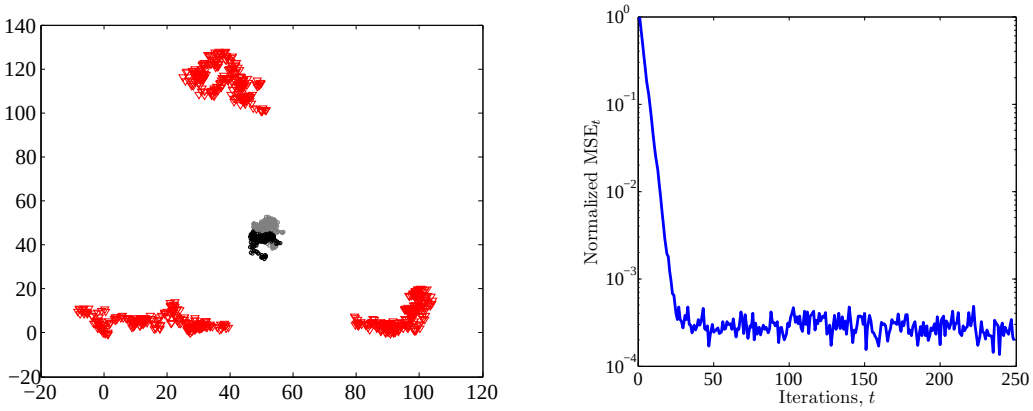
(a) (b)

Figure 4.12: Coordinated motion with known drift: (a) The motion of the anchors (shown as nablas) and two randomly chosen sensors (out of $N = 50$) shown as gray and black. (b) The normalized mean squared error.



(a) (b)

Figure 4.13: Coordinated motion with random drift: (a) The horizontal and (b) vertical coordinates of two randomly chosen sensors and the MDL estimates.



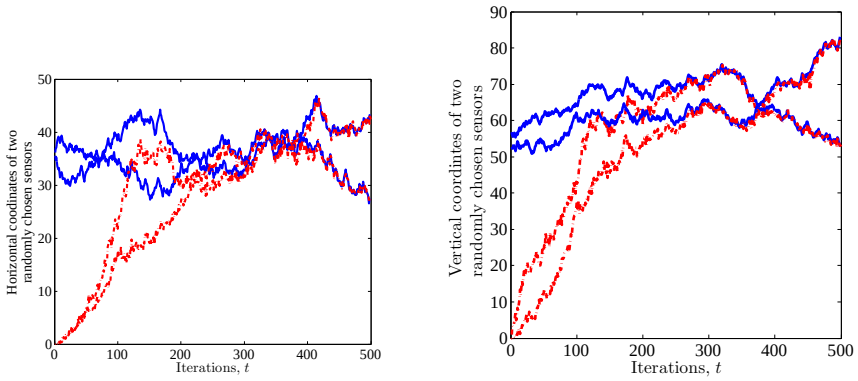
(a) (b)

Figure 4.14: Coordinated motion with random drift: (a) The motion of the anchors (shown as nablas) and two randomly chosen sensors (out of $N = 50$) shown as gray and black. (b) The log-normalized mean squared error.

of its element is a uniform random variable on the interval $[-2\ 2]$. Fig. 4.16(a) shows the motion for two randomly chosen sensors (notice that the anchors are fixed and do not move). For a network of $N = 50$ nodes, we show the resulting estimates as dashed lines against the original coordinates as solid lines in Fig. 4.15(a) and Fig. 4.15(b) for two randomly chosen sensors. We then plot the normalized mean squared error in Fig. 4.16(b). Notice that the estimates catch up the exact sensor coordinates but there is an error due to the random drift. We use log scale to show the steady state error.

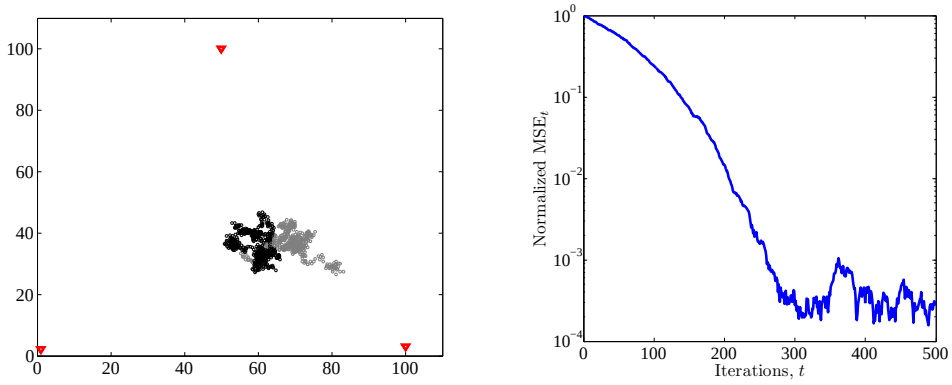
4.9 Conclusions

In this chapter, we present DILOC, a distributed iterative sensor localization algorithm in $m-$dimensional Euclidean space, $\mathbb{R}^m$ ($m \geq 1$), that finds the location coordinates of the sensors in a sensor network with only local communication. DILOC requires the minimal number, $m+1$, of anchors (network nodes with known location) to localize an arbitrary number, M, of sensors that lie in the convex hull of these $m+1$



(a) (b)

Figure 4.15: Uncoordinated motion in a fixed region with random drift: (a) The horizontal coordinates of two randomly chosen sensors and the MDL estimates. (b) The vertical coordinates of two randomly chosen sensors and the MDL estimates.



(a) (b)

Figure 4.16: Uncoordinated motion in a fixed region with random drift: (a) The motion of two randomly chosen sensors (out of $N = 50$) shown as gray and black. (b) The log-normalized mean squared error.

anchors. In the deterministic case, i.e, when there is no noise in the system, we show that our distributed algorithms, DILOC and DILOC-REL, lead to convergence to the exact sensor locations.

We then consider natural extensions to DILOC, i.e., when the network topology is not static, with more than $m + 1$ anchors, and with more than $m + 1$ neighbors. The convergence is proved in all cases and the convergence is shown to be exact. In particular, we provide the following results: (i) Choosing the sensor network communication topology dynamically improves the worst case performance; (ii) Increasing the number of anchors increases the convergence of the algorithm. (iii) Increasing the number of neighbors significantly improves the worst case performance.

We then present the MDL algorithm for localization and tracking of wireless mobile devices. The MDL algorithm requires the devices to be tracked to lie in the convex hull of at least $m + 1$ anchors that know their exact locations. We present a general model to capture the motion dynamics and discuss realistic practical cases where such motion can occur as special cases of this general model. Under minimal assumptions on network connectivity and deployment, we show that algorithm converges to exact locations in the mobile case when there is no random drift in the motion. Under random drift, we bound the steady state error that arises due to this randomness.

We further provide extensive simulations to support the theoretical claims.

Chapter 5

Banded matrix inversion

In this chapter, we specialize HDC to invert banded matrices. Recall that the distributed Jacobi algorithm provided in Section 3.4 can be employed to invert positive-definite matrices in a distributed fashion. As we will show, at each network node, the complexity of the distributed Jacobi algorithm scales linearly with n for inverting $n \times n$ matrices. We provide the distributed iterate-collapse inversion (DICI) algorithm for inverting banded matrices whose complexity is independent of n at each node. The DICI algorithm exploits the structure of banded matrices and appends a non-linear collapse step to the HDC iterations. We apply DICI algorithm to invert the information matrices in a computationally efficient distributed implementation of the Kalman filter (Chapter 6) and show its application towards inverting arbitrary sparse SPD matrices.

Parts of this chapter have been presented in [48, 35].

5.1 Introduction

Banded matrices are found frequently in signal processing, e.g., in the context of discretization of partial differential equations governing a random field and autoregressive or moving average image modeling. When they are constrained to be symmetric positive definite (SPD) matrices, they are the inverses of the covariance matrices of causal or non-causal Gauss-Markovian random processes [119]. Furthermore, in linear

algebra, solving a sparse large linear system of equations is a well studied problem, where a sparse matrix inverse is to be calculated. By employing matrix reordering algorithms [1], we can convert sparse SPD matrices to banded SPD matrices and use the theory developed in this chapter to compute the inverse.

The direct inverse of banded matrices can be computed centrally but that requires extensive storage, communication, and computation. Algorithms to compute direct inverses include Gauss-Jordan elimination. Most inversion algorithms for SPD matrices involve a Cholesky factorization that is efficient on a single processor implementation as long as computation power and memory requirements are within limits. Incomplete Cholesky factorization is also employed to solve large sparse SPD linear systems [76].

Recursive inversion of banded matrices can be found in [120, 121]. In [120], a forward-backward recursion algorithm is provided to compute the inverse of a tridiagonal matrix, which involves a forward recursion to start from the first node and reach the last node, and a backward recursion that proceeds in the opposite direction. Since, the iterations involve serial communication of the local matrices among all the nodes, the associated latency is impractical, besides requiring an inordinate amount of communication. Distributed Jacobi algorithm of Section 3.4 can also be employed to invert banded matrices but, as we will show, the computational complexity scales with the dimensions of the matrix.

In this chapter, we present a distributed iterate-collapse inversion algorithm, named DICI (pronounced die-see to sound like spicy), for $L-$banded symmetric positive-definite (SPD) matrices. The computational complexity of the DICI algorithm, at each node, is independent of the size of the matrix and only depends on its bandwidth, L. DICI algorithm exploits the structure of banded matrices and appends a non-linear collapse step to the HDC iterations. We now summarize the rest of the chapter. In Section 5.2, we provide some relevant properties of banded matrices and their inverses, whereas Section 5.3 provides the problem formulation. In Section 5.4, we show the complexity of the distributed Jacobi algorithm to scale with the size of $n \times n$ matrices. We then present the DICI algorithm in Section 5.5, and its application to sparse matrix inversion in subsection 5.6. Section 5.7 concludes the

chapter.

5.2 Properties of banded matrices

We start by some relevant definitions.

Definition 6 (L-banded matrices): A matrix, $\mathbf{Z} = \{z_{ij}\} \in \mathbb{R}^{n\times n}$, is referred to as an L-banded matrix ($L \geq 0$), if the elements outside the band defined by the Lth upper and the Lth lower diagonal are zero. Mathematically, if $\mathbf{Z} = \{z_{ij}\}$ is an L-banded matrix, then we have

$$z_{ij} = 0, \qquad |i-j| > L. \tag{5.1}$$

Note that a banded matrix is not necessarily a blocked diagonal matrix.

Definition 7 (Semi-separable matrices): The set, $\mathbf{\Xi}$, of semi-separable matrices [122, 123] is defined by

$$\mathbf{\Xi} = \left\{ \mathbf{S} \in \mathbb{R}^{n\times n} \,\middle|\, \mathbf{S}^{-1} \text{ is an } L\text{-banded, SPD matrix} \right\}. \tag{5.2}$$

We provide a relevant result on semi-separable matrices.

Lemma 24: Let $\mathbf{S} = \{s\}_{ij} \in \mathbf{\Xi}$ be a semi-separable matrix, then any non L-band element of $\mathbf{S}$, i.e., $s_{ij},\ |i-j| > L$, can be written as a function of its L-band elements, i.e., $s_{ij},\ |i-j| \leq L$. For $L = 1$, we have

$$s_{ij} = s_{i,j-1} s^{-1}_{i+1,j-1} s_{i+1,j}, \qquad |i-j| > L. \tag{5.3}$$

If any of the elements on the R.H.S of (5.3) is a non $L-$band element, then we write it first in terms of $L-$banded elements using (5.3). Lemma 24 uses a determinant maximizing completion of $\mathbf{S}$ assuming that its non L-band elements are unspecified. The proof along with appropriate formulae to replace (5.3) when $L > 1$ are provided in [121].

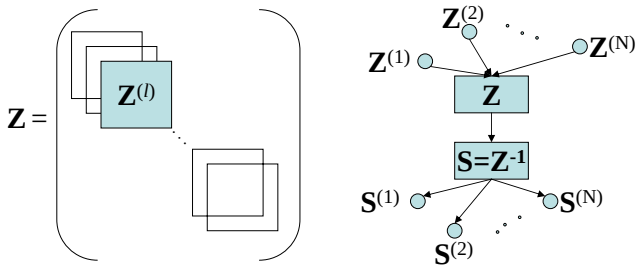


Figure 5.1: Composition of the $L-$band of $\mathbf{Z}$ from the local matrices, $\mathbf{Z}^{(l)}$, shown in the left figure. Centralized Implementation of $\mathbf{S} = \mathbf{Z}^{-1}$, shown in the right figure.

5.3 Problem formulation

Consider $\mathbf{Z}$ to be an $L-$banded symmetric positive-definite (SPD) matrix. We are interested in computing $\mathbf{S} = \mathbf{Z}^{-1}$, when the non-zero sub-matrices along the main diagonal of $\mathbf{Z}$ are distributed among N processing nodes[1]. This is shown in figure 5.1 (left). The lth node has a diagonal sub-matrix, $\mathbf{Z}^{(l)}$, that we refer to as the *local* matrix at node l. We would like to devise a distributed algorithm to compute $\mathbf{S}^{(l)}$ from $\mathbf{Z}^{(l)}$ (see Fig. 5.2) that requires local communication and low-order computation. Note that once we compute $\mathbf{S}^{(l)}$ (a sub-matrix on the main diagonal of $\mathbf{S}$ containing elements inside the L-band), any non-L band element can be computed using Lemma 24.

5.4 Distributed Jacobi algorithm for matrix inversion

The inversion of banded matrices is equivalent to solving a linear systems of equations

$$\mathbf{ZS} = \mathbf{I}, \tag{5.4}$$

[1]This arises in problems where the information is distributed in a large geographical region, e.g., estimation of a large-scale dynamical system through distributed sensing (see Chapter 6). On the other hand, we may be interested in solving a very large linear system of equations on a multiprocessor machine where parallelizing the algorithm is essential in load balancing and its real-time implementation, and hence the matrix is distributed among different available processors. Typically for such problems $L \ll n$.

for $\mathbf{S}$. Hence, the matrix version of the distributed Jacobi algorithm (as provided in Section 3.4) can be specialized to solve (5.4), which is given by the following iterations:

$$\mathbf{S}(k+1) = \mathbf{P}_\gamma \mathbf{S}(k) + \mathbf{B}_\gamma, \tag{5.5}$$

where

$$\begin{aligned} \mathbf{P}_\gamma &= (1-\gamma)\mathbf{I} + \gamma \mathbf{M}^{-1}(\mathbf{M} - \mathbf{Z}), & (5.6)\\ \mathbf{B}_\gamma &= \gamma \mathbf{M}^{-1}, & (5.7)\\ \mathbf{M} &= \mathrm{diag}(\mathbf{Z}), & (5.8) \end{aligned}$$

and $\gamma > 0$ is a sufficiently small relaxation parameter. Note that since $\mathbf{Z}$ is $L-$banded, $\mathbf{P}_\gamma$ is also $L-$banded. It can be shown that $\mathbf{S} = \mathbf{Z}^{-1}$ is the fixed point of (5.5). Choosing appropriate γ, we can show that $\rho(\mathbf{P}_\gamma) < 1$ and (5.5) converges to $\mathbf{Z}^{-1}$ as shown in Section 3.4.

We distribute the above algorithm among the network nodes as follows. The iteration for the ij-th element, s_{ij}, in $\mathbf{S}(k+1)$ can be written at time $k+1$ as

$$\begin{aligned} s_{ij}(k+1) &= \mathbf{p}_i \mathbf{s}^j(k) \qquad (i \neq j) & (5.9)\\ s_{ij}(k+1) &= \mathbf{p}_i \mathbf{s}^j(k) + b_{ii} \qquad (i = j) & (5.10) \end{aligned}$$

where $\mathbf{p}_i$ is the ith row of $\mathbf{P}_\gamma$, $\mathbf{s}^j(k)$ is the jth column of $\mathbf{S}(k)$, and b_{ii} is the (i,i) element of $\mathbf{B}_\gamma$. Since, $\mathbf{P}$ is $L-$banded, we note that the only non-zero elements in the ith row, $\mathbf{p}_i$, of $\mathbf{P}$ are located at most at the $i-L, \ldots, i+L$ locations and can be represented by $\{\mathbf{p}_i\}_q$, where q goes from $i-L, \ldots, i+L$. These non-zero elements pick the corresponding elements, $\{\mathbf{s}^j(k)\}_q$, in the jth column, $\mathbf{s}^j(k)$, of $\mathbf{S}(k)$, from (5.9) or (5.10), which are available at the nearby nodes.

Drawbacks for distributed Implementation: Recall that we are interested in computing $\mathbf{S}^{(l)}$ from $\mathbf{Z}^{(l)}$. Although it may seem like we only require to iterate on the elements inside $\mathbf{S}^{(l)}$, this is not the case. This is because $\mathbf{Z}$ is banded and non block-diagonal and $\mathbf{S}$, inverse of $\mathbf{Z}$, is, in general, a full matrix. This can be shown

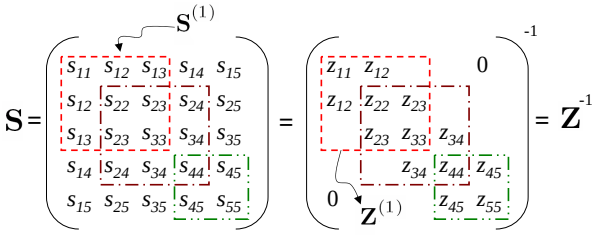


Figure 5.2: An example of a 5×5, $L = 1$-banded matrix inversion.

by writing out the iteration on the $L-$band element s_{45} from (5.9), see figure 5.2 for $L = 1-$banded $\mathbf{Z}$,

$$s_{45}(k+1) = p_{43}s_{35}(k) + p_{44}s_{45}(k) + p_{45}s_{55}(k), \tag{5.11}$$

where p_{ij} is the (i, j) element of $\mathbf{P}_\gamma$. Equation (5.11) shows that the iterations on an $L-$band element s_{45} requires a non $L-$band element s_{35}. The iterations on the non $L-$band element s_{35} are given by

$$s_{35}(k+1) = p_{32}s_{25}(k) + p_{33}s_{35}(k) + p_{34}s_{45}(k). \tag{5.12}$$

The computation in (5.12) involves $s_{25}(k)$ that lies in the non $L-$band of $\mathbf{S}(k)$, iterating on which, in turn, requires another non $L-$band element, $s_{15}(k)$, and so on. Computing the elements outside the $L-$band, thus, requires iterating on all the elements in a single column of $\mathbf{S}$, at the corresponding node. Hence, a single iteration of the algorithm, although requiring only local communication, sweeps the entire columns in $\mathbf{S}$ at the corresponding nodes and the complexity of this approach, at each node, scales with the size, n, of the linear system.

5.5 Distributed Iterate Collapse Inversion (DICI) Algorithm

In this section, we present the distribute iterate collapse inversion (DICI) algorithm. The DICI algorithm is divided into two steps:

(i) iterate step;

(i) collapse step.

The **iterate step** is applied to the L-band elements only and is given by

$$s_{ij}(k+1) = \begin{cases} \mathbf{p}_i \mathbf{s}^j(k), & i \neq j, \\ \mathbf{p}_i \mathbf{s}^j(k) + b_{ii}, & i = j, \end{cases} \qquad |i-j| \leq L, \tag{5.13}$$

where the symbols are defined as in Section 5.4. As we explained before in Section 5.4, the implementation of (5.13) requires non L-banded elements that, in turn, require further non L-banded elements. To address this problem, we introduce a **collapse step**, which uses Lemma 24. Hence, when a non L-band element is required by the iterate step, we use the collapse step so that further non L-band elements do not repeat.

In the context of Fig. 5.2, instead of iterating on s_{35} as in (5.12), we employ the collapse step,

$$s_{35}(k+1) = s_{34}(k) s_{44}^{-1}(k) s_{45}(k), \tag{5.14}$$

that prevents us from iterating further on the non L-band elements. The DICI algorithm can be easily extended to $L > 1$. The only modification required is in the collapse step, since (5.3) holds only for $L = 1$. The computation requirements for the DICI-OR algorithm are independent of n, at each sub-system, and it provides a scalable implementation of the matrix inversion problem.

5.5.1 Convergence of the DICI algorithm

The iterate and the collapse steps of the DICI algorithm can be combined in matrix form as follows.

$$\text{Iterate Step:} \qquad \mathbf{S}(k+1) = \mathbf{P}_\gamma \overline{\mathbf{S}}(k) + \mathbf{B}_\gamma, \qquad |i-j| \leq L \tag{5.15}$$

$$\text{Collapse Step:} \qquad \overline{\mathbf{S}}(k+1) = \zeta\left(\mathbf{S}(k+1)\right), \qquad |i-j| > L \tag{5.16}$$

The operator $\zeta(\cdot)$ is the collapse operator; it employs a maximizing determinant/entropy completion of a covariance matrix whose non L-band elements are unspecified using the results in [121]. The DICI algorithm is a composition of the linear function (the iterate step in (5.15)), $\mathbf{D} : \overline{\mathbf{S}}(k) \rightarrow \mathbf{P}_\gamma \overline{\mathbf{S}}(k) + \gamma \mathbf{M}^{-1}$, followed by the collapse operator, $\zeta(\cdot)$ given in (5.3) for $L = 1$ and in [121] for $L > 1$. Combining (5.15)–(5.16) summarizes the DICI algorithm for $\overline{\mathbf{S}}(k) \in \mathbf{\Xi}$ as,

$$\overline{\mathbf{S}}(k+1) = \zeta\left(\mathbf{P}_\gamma \overline{\mathbf{S}}(k) + \gamma \mathbf{M}^{-1}\right). \tag{5.17}$$

We define a composition map, $\mathbf{\Upsilon} : \mathbf{\Xi} \mapsto \mathbf{\Xi}$, as $\mathbf{\Upsilon} \doteq \zeta \circ \mathbf{D}$. To prove the convergence of the DICI algorithm, we show that the composition map, $\mathbf{\Upsilon}$, is a contraction map under some norm, that we choose to be the spectral norm $||\cdot||_2$, i.e., for some $\alpha \in [0, 1)$,

$$||\mathbf{\Upsilon}(\mathbf{X}_{\mathbf{\Xi}}) - \mathbf{\Upsilon}(\mathbf{Y}_{\mathbf{\Xi}})||_2 \leq \alpha ||\mathbf{X}_{\mathbf{\Xi}} - \mathbf{Y}_{\mathbf{\Xi}}||_2, \qquad \forall \mathbf{X}_{\mathbf{\Xi}}, \mathbf{Y}_{\mathbf{\Xi}} \in \mathbf{\Xi}. \tag{5.18}$$

The convergence of the iterate step of the DICI algorithm is based on the iterate operator, $\mathbf{P}_\gamma$, which is proved to be a contraction map in [58]. For the convergence of the collapse operator, ζ, we resort to a numerical procedure and show, in the following, that (5.18) is a contraction by simulating (5.18) 1.003×10^6 times.

For the simulations, we generate $n \times n$ matrices, $\mathbf{X}_{\text{rand}}$, with i.i.d. normally distributed elements and get, $\mathbf{X}_{\text{sym}} = \mathbf{X}_{\text{rand}} + \mathbf{X}_{\text{rand}}^T$. We eigen-decompose $\mathbf{X}_{\text{sym}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$. We replace $\mathbf{\Lambda}$ with a diagonal matrix, $\mathbf{\Lambda}_\Xi$, whose diagonal elements are drawn from a uniform distribution in the interval $(0, 10]$. This leads to a random symmetric positive definite matrix that lies in $\mathbf{\Xi}$ when we apply collapse operator, $\mathbf{X}_\Xi = \zeta\left(\mathbf{V}\mathbf{\Lambda}_\Xi\mathbf{V}^T\right)$. For $n = 100$ and L a random integer between 1 and $n/2 = 50$, we compute, by Monte Carlo simulations, the quotient of (5.18)

$$\frac{||\mathbf{\Upsilon}(\mathbf{X}_{\mathbf{\Xi}}) - \mathbf{\Upsilon}(\mathbf{Y}_{\mathbf{\Xi}})||_2}{||\mathbf{X}_{\mathbf{\Xi}} - \mathbf{Y}_{\mathbf{\Xi}}||_2}. \tag{5.19}$$

The number of trials is 1.003×10^6. The histogram of the values of α, in (5.19), (with 1000 bins) is plotted in Fig. 5.3. The maximum value of α found in these 1.003×10^6 simulations is 0.9851 and the minimum value is 0.8252. Since $\alpha \in (0, 1)$, i.e.,

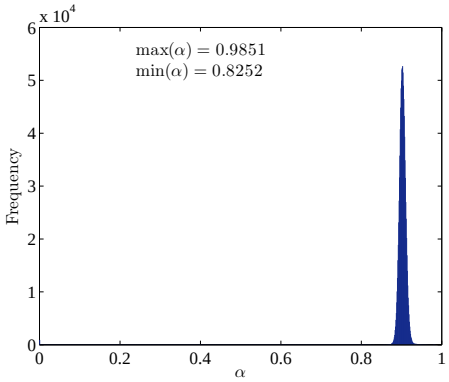


Figure 5.3: Histogram of α.

strictly less than 1, we assume that (5.18) is numerically verified.

5.5.2 Error bound for the DICI algorithm

Let the matrix produced by the DICI algorithm at the $k+1$-th iteration be $\widehat{\mathbf{S}}(k+1)$. The error process in the DICI algorithm is given by

$$\widehat{\mathbf{E}}(k+1) = \widehat{\mathbf{S}}(k+1) - \mathbf{S}. \tag{5.20}$$

Claim: The spectral norm of the error process, $||\widehat{\mathbf{E}}(k+1)||_2$, of the DICI algorithm is *bounded above* by the spectral norm of the error process, $||\widetilde{\mathbf{E}}(k+1)||_2$, of the distributed Jacobi algorithm. Since, the distributed Jacobi algorithm always converges for symmetric positive definite matrices, $\mathbf{Z}$, we deduce that the DICI algorithm converges.

We verify this claim numerically by Monte Carlo simulations. The number of trials is 4490, and we compute the error process, $\widehat{\mathbf{E}}(k+1)(K)$, of the DICI algorithm and the error process, $\widetilde{\mathbf{E}}(k+1)(K)$, of the distributed Jacobi algorithm. We choose the relaxation parameter, γ, to be 0.1. In Fig. 5.4(a), Fig. 5.4(b), and Fig. 5.4(c), we show the following,

$$\{\max_K,\ \min_K,\ \text{mean}_K\} \left(||\widetilde{\mathbf{E}}(k+1)(K)||_2 - ||\widehat{\mathbf{E}}(k+1)(K)||_2 \right),$$

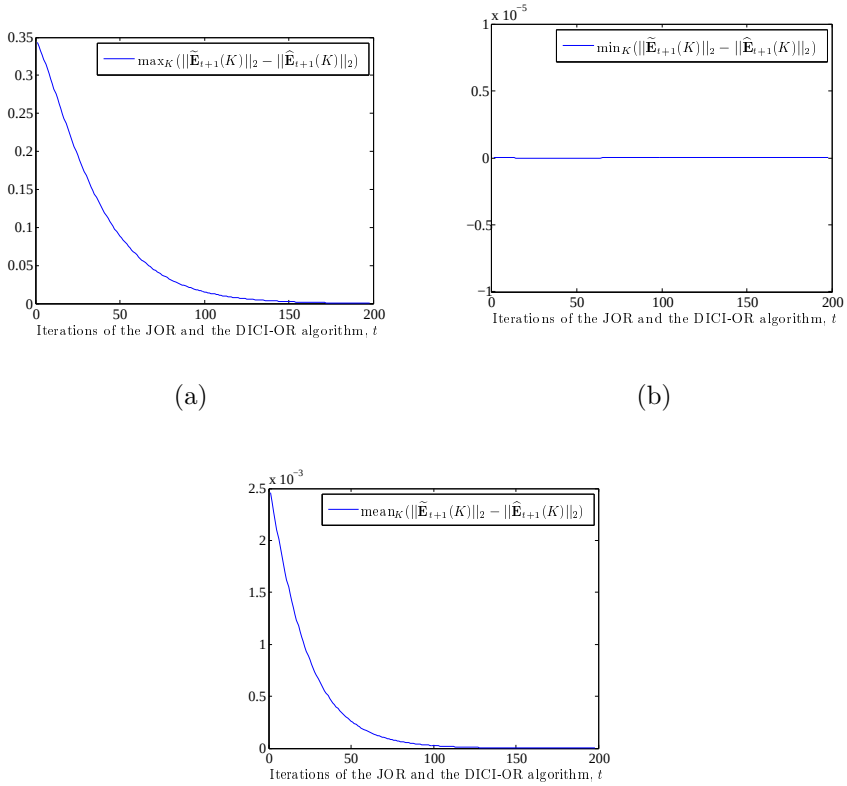


Figure 5.4: Simulation for the error bound of the DICI algorithm.

respectively, against the number of iterations of the distributed Jacobi and DICI algorithm. Since all the three figures show that the max, min, and the mean of the difference of the spectral norm of the two error processes, $||\widetilde{\mathbf{E}}(k+1)(K)||_2 - ||\widehat{\mathbf{E}}(k+1)(K)||_2$, is always ≥ 0, we deduce that

$$||\widehat{\mathbf{E}}(k+1)||_2 \quad \leq \quad ||\widetilde{\mathbf{E}}(k+1)||_2.$$

This verifies our claim numerically and provides us an upper bound on the spectral norm of the error process of the DICI algorithm.

5.6 Sparse matrix inversion

We show the extension of the DICI algorithm to invert sparse SPD matrices after applying matrix reordering algorithms to the sparse SPD matrices. These algorithms apply matrix bandwidth reduction methods, e.g., Reverse Cuthill Mckee (RCM) algorithm reordering [1], such that the sparse SPD matrices are converted to banded matrices by permutation of rows and columns.

Consider $\overline{\mathbf{Z}}$ to be an arbitrary sparse SPD matrix. We can apply the RCM algorithm to convert $\overline{\mathbf{Z}}$ into an $L-$banded matrix, $\mathbf{Z}$. The general reordering looks like

$$\mathbf{Z} = \mathbf{P}\overline{\mathbf{Z}}\mathbf{P}^T. \tag{5.21}$$

The inverse of $\overline{\mathbf{Z}}$ is given by

$$\overline{\mathbf{Z}}^{-1} = \mathbf{P}^T\mathbf{Z}^{-1}\mathbf{P}. \tag{5.22}$$

We can parallelize the computation of $\mathbf{Z}^{-1}$ on a multiprocessor machine using the DICI algorithm and computing $\overline{\mathbf{Z}}^{-1}$ reduces to low order computation at each processor (node) l, and two matrix multiplications. The matrix $\mathbf{P}$ is a permutation matrix and multiplying by it is a permutation of rows (and columns).

Remarks: It may seem that pre- and post-multiplication with the permutation matrix, $\mathbf{P}$, in (5.22), has to be implemented at a central location. In fact, this step can also be distributed by realizing that the permutation of rows and columns can be implemented by imposing a communication graph on the nodes using the structure of the permutation matrix, $\mathbf{P}$. Hence, $\mathbf{P}$, determines the communication topology required by the nodes to communicate the appropriate elements among the nodes.

We show the result of the RCM algorithm on a 100×100 sparse SPD matrix, $\overline{\mathbf{Z}}$, shown in figure 5.5(a), which is converted to a $L = 12-$band matrix, $\mathbf{Z}$, shown in figure 5.5(b), by the permutation matrix given by the RCM algorithm [1]. Depending on the number of nodes, the $L = 12-$banded matrix, $\mathbf{Z}$, shown in figure 5.5(b), is divided into overlapping local matrices. Note that the minimum size of the local matrix is $L + 1 \times L + 1$, which in the case ($L = 12$) is a 13×13 matrix.

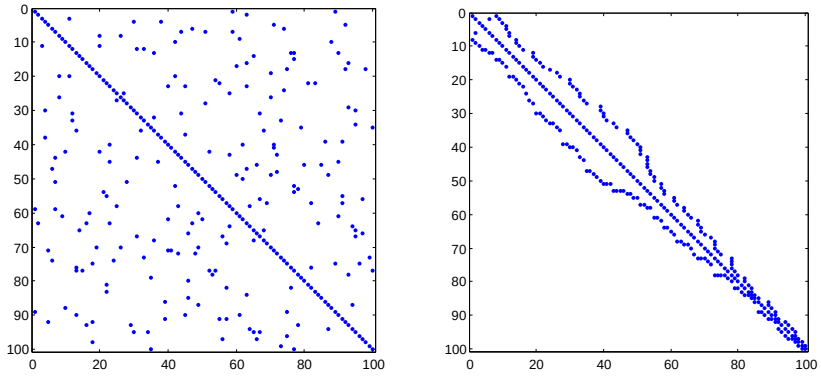


Figure 5.5: (a) A random sparse SPD matrix, $\overline{\mathbf{Z}}$, with sparseness density 0.03. Non-zero elements are shown in black. (b) $L = 12-$banded reordering of $\overline{\mathbf{Z}}$, shown in figure 5.5(a), using RCM algorithm [1].

5.7 Conclusions

We present a distributed inversion algorithm, DICI, for banded SPD matrices that is distributed both in terms of communication and computations. Each node does local communication and performs computation of order $O(L^4 t)$ with only local matrices, where $L << n$ and t is the number of iterations of the DICI algorithm. The algorithm has significant importance when applied to problems where partial information about a global phenomenon is available at the nodes and where parallelized solutions are sought under resource constraints for load balancing. We use the DICI algorithm to distribute the Kalman filter in the next chapter.

Chapter 6

Distributed estimation in large-scale systems

In this chapter, we present a *distributed* Kalman filter to estimate the state of a sparsely connected, large-scale, n-dimensional, dynamical system monitored by a network of N sensors. We implement local Kalman filters on n_l-dimensional subsystems, $n_l \ll n$, obtained by spatially decomposing the large-scale system. The distributed Kalman filter is optimal under an Lth order Gauss-Markov approximation to the centralized filter. Gauss-Markov covariances have banded inverses and we employ the DICI algorithm (Chapter 5) to compute these error covariances in a distributed fashion. Hence, the computation of the centralized (approximated) Riccati and Lyapunov equations is distributed and requires local communication and low-order computation. We fuse the observations that are common among the local Kalman filters using bipartite fusion graphs and a linear average-consensus algorithm. The proposed algorithm achieves full distribution of the Kalman filter. Nowhere in the network, storage, communication, or computation of n-dimensional vectors and matrices is needed; only $n_l \ll n$ dimensional vectors and matrices are communicated or used in the local computations at the sensors.

Parts of this chapter have been presented in [46, 47, 48, 35].

6.1 Introduction

Centralized implementation of the Kalman filter [124, 125], although possibly optimal, is neither robust nor scalable to complex large-scale dynamical systems with their measurements distributed on a large geographical region. The reasons are twofold: (i) the large-scale systems are very high-dimensional, and thus require extensive computations to implement the centralized procedure; and (ii) the span of the geographical region, over which the large-scale system is deployed or the physical phenomenon is observed, poses a large communication burden and thus, among other problems, adds latency to the estimation mechanism. To remove the difficulties posed by centralization, we decompose the large-scale system into n_l-dimensional sub-systems and distribute the estimation algorithm with a low order Kalman filter implemented at each of these sub-systems. To account for the processing, communication, and limited resources at the sub-systems, the local Kalman filters involve computations and communications with local quantities only, i.e., vectors and matrices of low dimensions, $n_l \ll n$, where n is the dimension of the state vector—no sensor computes, communicates, or stores any n-dimensional quantity.

Much of the existing research on distributed Kalman filters focuses on sensor networks monitoring *low*-dimensional systems, where an nth order Kalman filter is replicated at each sensor. This replication is only practical when the dimension of the state is small, for example, when multiple sensors mounted on a small number of robot platforms are used for target tracking [126, 127, 128]. The problem in such scenarios reduces to how to efficiently incorporate the distributed observations, which is also referred to in the literature as 'data fusion,' see also [129]. Data fusion for Kalman filters over arbitrary communication networks is discussed in [37], using iterative consensus protocols in [130]. The consensus protocols in [130] are assumed to converge asymptotically, thus, between any two time steps of the Kalman filter, the consensus protocols require an infinite number of iterations to achieve convergence. References [131, 127] incorporate packet losses, intermittent observations, and communication delays in the data fusion process. Because they replicate an n-dimensional Kalman filter at each sensor, they communicate and invert $n \times n$ matrices locally,

which, in general, is an $O(n^3)$ computation. This may be viable for low dimensional systems, as in tracking, but, unacceptable in the problems we consider where the state dimension, n, is very large, for example, in the range of 10^2 to 10^9. In such problems, replication of the global dynamics in the local Kalman filters is either not practical or not possible.

Kalman filters with reduced order models have been studied in, e.g., [132, 133] to address the computation burden posed by implementing nth order models. In these works, the reduced models are decoupled, which is sub-optimal, as important coupling among the system variables is ignored. Furthermore, the network topology is either fully connected [132], or is close to fully connected [133], requiring long distance communication that is expensive. We are motivated by problems where the large-scale systems, although sparse, cannot be decoupled, and where, due to the sensor constraints, the communication and computation should both be local.

We present a distributed Kalman filter that addresses *both* the computation and communication challenges posed by complex large-scale dynamical systems, while preserving its coupled structure; in particular, nowhere in our distributed Kalman filter, do we store, communicate, or compute any n-dimensional quantity. As an interesting remark, nowhere either in the network is there a copy of the entire state estimate; in other words, knowledge about the state is intrinsically distributed. We briefly explain the key steps and approximations in our solution.

Spatial decomposition of complex large-scale systems: To distribute the Kalman filter, we provide a spatial decomposition of the complex large-scale dynamical system (of dimension n) that we refer to as *the overall system* into several, possibly many, local coupled dynamical systems (of dimension n_l, such that $n_l \ll n$) that we refer to as *sub-systems* in the following. The large-scale systems we consider are sparse and localized. Physical systems with such characteristics are described in Section 6.2.1, as resulting, for example, from a spatio-temporal discretization of random fields. These sub-systems overlap, i.e., they share states, and thus the resulting local Kalman filters also overlap. In addition to this overlap, the sub-systems are connected by local interactions that account for the coupling between the sub-systems. We preserve this coupling by modeling explicitly this information exchange among

the sub-systems.

Overlapping dynamics at the sub-systems: Bipartite fusion graphs: The sub-systems that we extract from the overall system overlap. In particular, some state variables are observed by several sub-systems. To fuse this shared information, we implement a fusion algorithm using bipartite fusion graphs, which we introduced in [46], and an average-consensus algorithm [17]. The interactions required by the fusion procedure are constrained to a small neighborhood and with particular choices of the communication topology, the observation fusion procedure remains single hop.

Assimilation of the local error covariances—DICI algorithm: A key issue when distributing the Kalman filter is that the local error covariances should approximate the centralized error covariance in a meaningful way. If the local error covariances evolve independently at each sub-system they may lose any coherence with the centralized error covariance. For example, in the estimation scheme in [134], the coupled states are applied as inputs to the local observers, but, the error covariances remain decoupled and no structure of the centralized error covariance is retained by the local filters. To keep coherence between the local covariances and the centralized covariance, we employ a *cooperative assimilation procedure* among the local error covariances that is based on approximating the centralized error process by a low dimensional Gauss-Markov error process[1]. The assimilation procedure is carried out with the DICI algorithm in Chapter 5.

In summary, spatial decomposition of complex large-scale systems, fusion algorithms for fusing observations, and the DICI algorithm to assimilate the local error covariances combine to give a robust, scalable, and distributed implementation of the Kalman filter.

We describe the rest of the chapter. Section 6.2 motivates the discrete-time models and describes the centralized Information filters[2] (CIFs) and the centralized L-banded Information filters (CLBIFs). Section 6.3 covers the model distribution step.

[1]In the error covariance domain, this approximation corresponds to the determinant/entropy maximizing completion of a partially specified (L-band, in our case) covariance matrix [135, 119, 120, 136]. Such a completion results into a covariance matrix whose inverse is L-banded.

[2]We use the Kalman filter in the Information filter format [137, 133]. Information filter is algebraically equivalent to the Kalman filter.

We introduce the local Information filters in Section 6.3.2 along with the necessary notation. Section 6.4 gives the observation fusion step of the local Information filters. We provide the filter step of the local Information filters in Section 6.6, and the prediction step of the local Information filters in Section 6.7. We conclude the chapter with results in Section 6.8 and conclusions in Section 7.2.4. Appendix C.1 discusses the L-banded inversion theorem, [120].

6.2 Background

In this section, we motivate the type of applications and large-scale dynamical systems of interest to us. The context is that of a time-varying random field governed by partial differential equations (PDEs); these systems can also be generalized to arbitrary dynamical systems belonging to a particular structural class, as we elaborate in Subsection 6.2.1. To fix notation, we then present the centralized version of the Information filter.

6.2.1 Global model

Global dynamical system

Our goal here is to motivate how discrete linear models occur that exhibit a sparse and localized structure that we use to distribute the model in Section 6.3. Examples include physical phenomena [138, 139, 140, 141, 142], e.g., ocean/wind circulation and heat/propagation equations, that can be broadly characterized by a PDE of the Navier-Stokes type. These are highly non-linear and different regimens arise from different assumptions. For data assimilation, i.e., combining models with measured data, e.g., satellite altimetry data in ocean models, it is unfeasible to use non-linear models; rather, linearized approximations (dynamical linearization) are employed. Hence, we take a very simplistic example and consider the discretization of a spatio-temporal dynamical system,

$$\dot{x}_t = \mathcal{L}x_t + u_t, \tag{6.1}$$

where $\dot{x}_t$ is the time partial derivative of a continuous-time physical phenomenon (e.g., heat, wave or wind), u_t is random noise, and $\mathcal{L}$, for example, is a 2nd order elliptical operator (that arises in the heat equation in diffusion),

$$\mathcal{L} = \alpha \frac{\partial^2}{\partial \rho_x^2} + \beta \frac{\partial^2}{\partial \rho_y^2}, \tag{6.2}$$

where ρ_x and ρ_y represent the horizontal and vertical dimensions, respectively, and α, β are constants pertinent to the specific application.

We start by discretizing the elliptical operator (6.2), using a standard 2nd order difference approximation on an $M \times J$ uniform mesh grid,

$$\frac{\partial^2 x_t}{\partial \rho_x^2} \sim x_{i+1,j} - 2x_{i,j} + x_{i-1,j}, \qquad \frac{\partial^2 x_t}{\partial \rho_y^2} \sim x_{i,j+1} - 2x_{i,j} + x_{i,j-1}, \tag{6.3}$$

where x_{ij} is the value of the random field, x_t, at the ij-th location in the $M \times J$ grid. We collect the variables, $\{x_{ij}\}$, in a state vector, $\mathbf{x}$, by, for example, using lexicographic ordering. Let $\mathbf{A}$ be a tridiagonal matrix, with zeros on the main diagonal and ones on the upper and lower diagonal; approximating the time derivative in (6.1) by using the forward Euler method, we can write the spatio-temporal discretization of (6.1) as

$$\mathbf{x}_{k+1} = (\mathbf{I} + \mathbf{F}_c)\, \mathbf{x}_k + \mathbf{G}\mathbf{u}_k, \qquad k \geq 0, \tag{6.4}$$

where k is the discrete-time index and the matrix $\mathbf{F}_c$ is given by

$$\mathbf{F}_c = \begin{bmatrix} \mathbf{B} & \mathbf{C} & \\ \ddots & \ddots & \ddots \\ & \mathbf{C} & \mathbf{B} \end{bmatrix} = \mathbf{I} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C}, \tag{6.5}$$

where $\mathbf{B} = \mu\mathbf{I} + \beta_h \mathbf{A}$, and $\mathbf{C} = \beta_v \mathbf{I}$, the constants μ, β_h, β_v are in terms of α, β in (6.2), and $\otimes$ is the Kronecker product [143]. Putting $\mathbf{F} = \mathbf{I} + \mathbf{F}_c$, the discrete-time dynamical system takes the form

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{u}_k. \tag{6.6}$$

In the above model, $\mathbf{x}_0 \in \mathbb{R}^n$ are the state initial conditions, $\mathbf{F} \in \mathbb{R}^{n \times n}$ is the model matrix, $\mathbf{u}_k \in \mathbb{R}^{j'}$ is the state noise vector and $\mathbf{G} \in \mathbb{R}^{n \times j'}$ is the state noise matrix.

Remarks: Here, we note that the model matrix, $\mathbf{F}$, is highly sparse, since the matrices $\mathbf{B}$ and $\mathbf{C}$ are at most tridiagonal, and is perfectly banded in case of PDEs. We can relax this to sparse and localized matrices as when the coupling among the states decays with distance (in an appropriate measure), for example, see the spatially distributed systems in [144]. We mention briefly two other examples where such discrete-space-time models (with sparse and localized structure) also occur. In image processing, the dynamics at a pixel depends on neighboring pixel values [145], [146]; power grid models, under certain assumptions, exhibit banded structures, [66, 147, 49]. As a final comment, systems that are sparse but not localized can be converted to sparse and localized by using matrix bandwidth reduction algorithms [1].

Observation model

Let the system described in (6.6) be monitored by a network of N sensors. Observations at sensor l and time k are

$$\mathbf{y}_k^{(l)} = \mathbf{H}_l \mathbf{x}_k + \mathbf{w}_k^{(l)}, \tag{6.7}$$

where $\mathbf{H}_l \in \mathbb{R}^{p_l \times n}$ is the local observation matrix for sensor l, p_l is the number of simultaneous observations made by sensor l at time k, and $\mathbf{w}_k^{(l)} \in \mathbb{R}^{p_l}$ is the local observation noise. In the context of the systems we are interested in, it is natural to assume that the observations are localized. These local observations at sensor l may be, e.g., the temperature or height at location l or an average of the temperatures or heights at l and neighboring locations. Mathematically, this can be characterized by assuming that $\mathbf{H}\mathbf{H}^T$ is sparse and banded.

We stack the observations at all N sensors in the sensor network to get the global observation model as follows. Let p be the total number of observations at all the

sensors. Let the global observation vector, $\mathbf{y}_k \in \mathbb{R}^p$, the global observation matrix, $\mathbf{H} \in \mathbb{R}^{p \times n}$, and the global observation noise vector, $\mathbf{w}_k \in \mathbb{R}^p$, be

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{y}_k^{(1)} \\ \vdots \\ \mathbf{y}_k^{(N)} \end{bmatrix}, \qquad \mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_N \end{bmatrix}, \qquad \mathbf{w}_k = \begin{bmatrix} \mathbf{w}_k^{(1)} \\ \vdots \\ \mathbf{w}_k^{(N)} \end{bmatrix}. \tag{6.8}$$

Then the global observation model is given by

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k. \tag{6.9}$$

We further assume that the overall system (6.6) and (6.9) is coupled, irreducible, and the pair $(\mathbf{F}, \mathbf{H})$ to be observable.

Statistical assumptions

We adopt standard assumptions on the statistical characteristics of the noise. The state noise sequence, $\{\mathbf{u}_k\}_{k\geq 0}$, the observation noise sequence, $\{\mathbf{w}_k\}_{k\geq 0}$, and the initial conditions, $\mathbf{x}_0$, are independent, Gaussian, zero-mean, with

$$\mathbb{E}[\mathbf{u}_\iota \mathbf{u}_\tau^H] = \mathbf{Q}\delta_{\iota\tau} \text{ and } \mathbb{E}[\mathbf{w}_\iota \mathbf{w}_\tau^H] = \mathbf{R}\delta_{\iota\tau}, \text{ and } \mathbb{E}[\mathbf{x}_0 \mathbf{x}_0^H] = \mathbf{S}_0, \tag{6.10}$$

where the superscript H denotes the Hermitian, the Kronecker delta $\delta_{\iota\tau} = 1$, if and only if $\iota = \tau$, and zero otherwise. Since the observation noises at different sensors are independent, we can partition the global observation noise covariance matrix, $\mathbf{R}$, into blocks $\mathbf{R}_l \in \mathbb{R}^{p_l \times p_l}$ corresponding to the local observation noise covariance matrices at each sensor l, as

$$\mathbf{R} = \text{blockdiag}[\mathbf{R}_1, \ldots, \mathbf{R}_N]. \tag{6.11}$$

For the rest of the presentation, we consider time-invariant models, specifically, the matrices, $\mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{Q}, \mathbf{R}$, are time-invariant. The discussion, however, is not limited to either zero-mean initial conditions or time-invariant models and generalizations to the time-variant models will be added as we proceed.

6.2.2 Centralized Information Filter (CIF)

Let $\breve{\mathbf{S}}_{k|k}$ and $\breve{\mathbf{S}}_{k|k-1}$ be the (filtered and prediction, respectively) error covariances, and their inverses be the information matrices, $\breve{\mathbf{Z}}_{k|k}$ and $\breve{\mathbf{Z}}_{k|k-1}$. Let $\widehat{\mathbf{x}}_{k|k}$ and $\widehat{\mathbf{x}}_{k|k-1}$ be the filtered estimate and the predicted estimate of the state vector, $\mathbf{x}_k$, respectively. We have the following relations.

$$\breve{\mathbf{S}}_{k|k} = \breve{\mathbf{Z}}_{k|k}^{-1} \tag{6.12}$$

$$\breve{\mathbf{S}}_{k|k-1} = \breve{\mathbf{Z}}_{k|k-1}^{-1} \tag{6.13}$$

Define the n-dimensional global transformed state vectors as

$$\widehat{\mathbf{z}}_{k|k-1} = \breve{\mathbf{Z}}_{k|k-1}\widehat{\mathbf{x}}_{k|k-1}, \tag{6.14}$$

$$\widehat{\mathbf{z}}_{k|k} = \breve{\mathbf{Z}}_{k|k}\widehat{\mathbf{x}}_{k|k}. \tag{6.15}$$

Define the n-dimensional global observation variables as

$$\mathbf{i}_k = \mathbf{H}^T\mathbf{R}^{-1}\mathbf{y}_k, \tag{6.16}$$

$$\mathcal{I} = \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}, \tag{6.17}$$

and the n-dimensional local observation variables at sensor l as

$$\mathbf{i}_{l,k} = \mathbf{H}_l^T\mathbf{R}_l^{-1}\mathbf{y}_k^{(l)}, \tag{6.18}$$

$$\mathcal{I}_l = \mathbf{H}_l^T\mathbf{R}_l^{-1}\mathbf{H}_l. \tag{6.19}$$

When the observations are distributed among the sensors, see (7.15), the CIF can be implemented by collecting all the sensor observations at a central location; or, with observation fusion, by realizing that the global observation variables in (6.16)−(6.17)

can be written as, see [126, 133, 37],

$$\mathbf{i}_k = \sum_{l=1}^{N} \mathbf{i}_{l,k}, \tag{6.20}$$

$$\mathcal{I} = \sum_{l=1}^{N} \mathcal{I}_l. \tag{6.21}$$

The *filter step* of the CIF is

$$\check{\mathbf{Z}}_{k|k} = \check{\mathbf{Z}}_{k|k-1} + \sum_{l=1}^{N} \mathcal{I}_l, \qquad k \geq 0, \tag{6.22a}$$

$$\widehat{\mathbf{z}}_{k|k} = \widehat{\mathbf{z}}_{k|k-1} + \sum_{l=1}^{N} \mathbf{i}_{l,k}, \qquad k \geq 0. \tag{6.22b}$$

The *prediction step* ($k \geq 1$) of the CIF is

$$\check{\mathbf{Z}}_{k|k-1} = \check{\mathbf{S}}_{k|k-1}^{-1} = (\mathbf{F}\check{\mathbf{Z}}_{k-1|k-1}^{-1}\mathbf{F}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T)^{-1}, \qquad \check{\mathbf{Z}}_{0|-1} = \check{\mathbf{S}}_0^{-1}, \tag{6.23a}$$

$$\widehat{\mathbf{z}}_{k|k-1} = \check{\mathbf{Z}}_{k|k-1}\left(\mathbf{F}\check{\mathbf{Z}}_{k-1|k-1}^{-1}\widehat{\mathbf{z}}_{k-1|k-1}\right), \qquad \widehat{\mathbf{z}}_{0|-1} = \mathbf{0}. \tag{6.23b}$$

The CIF needs: (i) the knowledge of all the observations, $\mathbf{y}_k$, at a central location to compute (6.20), a non-trivial communication task when the number of sensors, N, is large; and (ii) global filter computations, e.g., (6.23), an infeasible challenge when the number of states, n, is large. Further, the CIF has the disadvantages of large latency and a single point of failure.

6.2.3 Centralized L-Banded Information Filters (CLBIF)

To avoid the $O(n^3)$ computations of the global quantities in (6.23), e.g., the inversion, $\check{\mathbf{Z}}_{k-1|k-1}^{-1}$, we may approximate the information matrices, $\check{\mathbf{Z}}_{k|k}$ and $\check{\mathbf{Z}}_{k|k-1}$, to be L-banded matrices, $\mathbf{Z}_{k|k}$ and $\mathbf{Z}_{k|k-1}$. We refer to the CIF with this approximation as the centralized L-banded Information filter (CLBIF). This approach is studied

in [148], where the information loss between $\breve{\mathbf{Z}}$ and $\mathbf{Z}$, is given by the divergence

$$\begin{aligned}\mathrm{Div}_k(\breve{\mathbf{Z}}_{k|q}, \mathbf{Z}_{k|q}) &= \frac{1}{2}\left|\left|\breve{\mathbf{Z}}_{k|q}^{-\frac{T}{2}}\left(\mathbf{Z}_{k|q}-\breve{\mathbf{Z}}_{k|q}\right)\mathbf{Z}_{k|q}^{-\frac{1}{2}}\right|\right|_F^2, \\ &\leq \frac{1}{2}\left(\sum_i \lambda_{i(\mathbf{Z}_{k|q})}^{-\frac{1}{2}}\right)^2\left(\sum_i \lambda_{i(\breve{\mathbf{Z}}_{k|q})}^{-\frac{1}{2}}\right)^2\left|\left|\mathbf{Z}_{k|q}-\breve{\mathbf{Z}}_{k|q}\right|\right|_F^2, \quad (6.24)\end{aligned}$$

where $q = k$ for estimation and $q = k-1$ for prediction, $||\cdot||_F$ is the Frobenius norm and $\lambda_{i(\mathbf{Z}_{k|q})}$ is the ith eigenvalue of the matrix $\mathbf{Z}_{k|q}$. Although, for a fixed k the divergence in (6.24) is bounded, the sequence, $\mathrm{Div}_k(\cdot,\cdot)$, may be unbounded for small values of L. Here, we assume that L is chosen large enough, i.e., $L \geq L_{\min}$, such that $\lim_{k\to\infty}\mathrm{Div}_k(\cdot,\cdot)$ exists. The choice of $L_{\min}$ varies for different dynamical systems and loosely speaking depends on the structure of the model matrices, $\mathbf{F}$.

This banded approximation of the information matrices is equivalent to the determinant/entropy maximizing completion of its inverse, a covariance matrix; part of whose elements are unspecified. In our case, the unspecified elements are the non L-band elements, and it is well known that such completion of the covariance matrices have banded inverses with the same bandwidth, see, for instance, [135, 120, 136] and the references within. Furthermore, such covariance matrices result from approximating the Gaussian error processes,

$$\epsilon_{k|k} = \mathbf{x}_k - \widehat{\mathbf{x}}_{k|k}, \qquad \epsilon_{k|k-1} = \mathbf{x}_k - \widehat{\mathbf{x}}_{k|k-1}, \quad (6.25)$$

to Gauss-Markov of Lth order [119] (for $L = 1$, this has also been studied in [149]). Reference [120] presents an algorithm to derive the approximation that is optimal in Kullback-Leibler or maximum entropy sense in the class of all L-banded matrices approximating the inverse of the error covariance matrix. In the sequel, we assume this optimal L-banded approximation.

The CLBIF (with the L-banded information matrices, $\mathbf{Z}_{k|k}$ and $\mathbf{Z}_{k|k-1}$) is given by the *filter step* in (6.22a)−(6.22b) and the *prediction step* in (6.23a)−(6.23b), where the optimal information matrices, $\breve{\mathbf{Z}}_{k|k}$ and $\breve{\mathbf{Z}}_{k|k-1}$, are replaced by their L-banded approximations. The algorithms in [120, 121] reduce the computational complexity

of the CLBIF to $O(n^2)$ but the resulting algorithm is still centralized and deals with the n-dimensional state. To distribute the CLBIF, we start by distributing the global model (6.6)−(6.9) in the following section.

6.3 Spatial decomposition of large-scale systems

Instead of implementing CLBIF based on the global model, we implement local Information filters at the sub-systems obtained by spatially decomposing the overall system. Subsection 6.3.1 deals with this decomposition by exploiting the sparse and localized structure of the model matrix, $\mathbf{F}$.

6.3.1 Reduced model at each Sensor

This subsection shows how to distribute the global model (6.6) and (6.9), in order to get the reduced order sub-systems. We illustrate the procedure with a simple example that reflects our assumptions on the dynamical system structure. Consider a five dimensional system with the global dynamical model

$$\begin{aligned}\mathbf{x}_{k+1} &= \begin{bmatrix} f_{11} & f_{12} & 0 & 0 & 0 \\ f_{21} & f_{22} & 0 & f_{24} & 0 \\ f_{31} & 0 & f_{33} & 0 & 0 \\ 0 & 0 & f_{43} & 0 & f_{45} \\ 0 & 0 & 0 & f_{54} & f_{55} \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & g_{32} \\ 0 & 0 \\ g_{51} & 0 \end{bmatrix} \mathbf{u}_k \\ &= \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{u}_k. \end{aligned} \tag{6.26}$$

The system has two external noise sources $\mathbf{u}_k = [u_{1k}, u_{2k}]^T$. We monitor this system with $N = 3$ sensors, having scalar observations, $y_k^{(l)}$, at each sensor l. The global

observation vector, $\mathbf{y}_k$, stacks the local observations, $y_k^{(l)}$, and is

$$\begin{aligned}\mathbf{y}_k = \begin{bmatrix} y_k^{(1)} \\ y_k^{(2)} \\ y_k^{(3)} \end{bmatrix} &= \begin{bmatrix} h_{11} & h_{12} & h_{13} & 0 & 0 \\ 0 & h_{22} & h_{23} & h_{24} & 0 \\ 0 & 0 & 0 & h_{34} & h_{35} \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} w_k^{(1)} \\ w_k^{(2)} \\ w_k^{(3)} \end{bmatrix} \\ &= \mathbf{H}\mathbf{x}_k + \mathbf{w}_k, \end{aligned} \tag{6.27}$$

where $\mathbf{H} = [\mathbf{H}_1^T \ \mathbf{H}_2^T \ \mathbf{H}_3^T]^T$; the elements $\{f_{ij}\}$ and $\{h_{ij}\}$ in (6.26)–(6.27), are such that the dynamical system is $(\mathbf{F}, \mathbf{H})$-observable. We distribute the global model of equations (6.26) and (6.27) in the following subsections.

Graphical representation using system digraphs

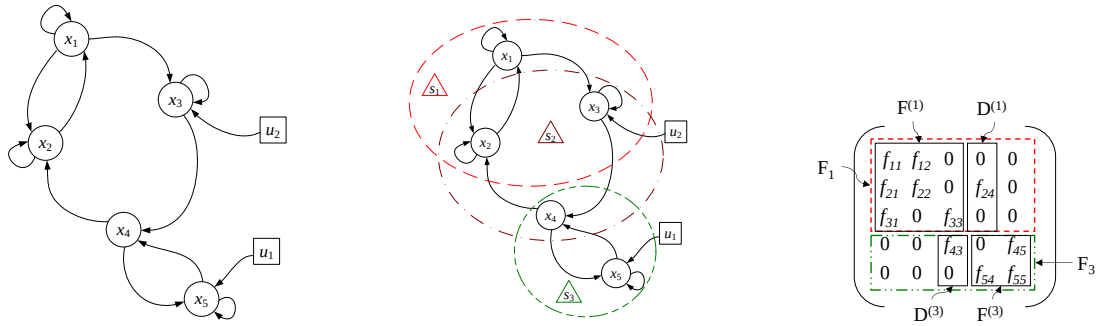
A system digraph visualizes the dynamical interdependence of the system. A system digraph, [134], $\mathcal{J} = [V, E]$, is a directed graphical representation of the system, where $V = X \cup U$ is the vertex set consisting of the states, $X = \{x_i\}_{i=1,\ldots,n}$, and the noise inputs, $U = \{u_i\}_{i=1,\ldots,j'}$. The interconnection matrix, E, is the binary representation (having a 1 for each non-zero entry) of the model matrix, $\mathbf{F}$, and the state noise matrix, $\mathbf{G}$, concatenated together. The interconnection matrix, E, for the system in (6.26) is,

$$E = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{6.28}$$

The system digraph is shown in Fig. 6.1(a).

Sub-system derivation using cut-point sets

We have $N = 3$ sensors monitoring the system through the observation model (6.27). We implement a sub-system at each sensor, thus, sub-system l corresponds to sensor l. We associate to each sub-system l a *cut-point set*, $V^{(l)}$, where $V^{(l)} \subseteq X$. We choose to include the states in a cut-point set that are observed by the sensors in



(a) (b) (c)

Figure 6.1: System Digraph and cut-point sets: (a) Digraph representation of the 5 dimensional system, (6.26)−(6.27). The circles represent the states, $\mathbf{x}$, and the squares represent the input noise sources, $\mathbf{u}$. (b) The cut-point sets associated to the 3 sub-systems ($\triangle$) are shown by the dashed circles. (c) Partitioning of the global model matrix, $\mathbf{F}$, into local model matrices, $\mathbf{F}^{(l)}$, and the local internal input matrices, $\mathbf{D}^{(l)}$, shown for sub-system 1 and sub-system 3, from the example system, (6.26)−(6.27).

its corresponding sub-system; see [150] for an alternate definition of the cut-point sets, and algorithms to find all cut-point sets and a minimal cut-point set, if it exists. The cut-point sets select the local states involved in the local dynamics at each sub-system. From (6.27), the cut-point sets[3] are shown in Fig. 6.1(b) where we have the following cut-point set, e.g., at sub-system 1, $V^{(1)} = \{x_1, x_2, x_3\}$.

Dimension of the sub-systems: The local states at sub-system l, i.e., the components of the local state vector, $\mathbf{x}_k^{(l)}$, are the elements in its associated cut-point set, $V^{(l)}$. The dimension of the local Kalman filter implemented at sub-system l is now n_l. The set of n_l-dimensional local Kalman filters will give rise, as will be clear later, to an L-banded centralized information matrix with $L = \min(n_1, \ldots, n_N)$. The loss in the optimality as a function of L is given by the divergence (6.24). Hence, for a desired level of performance, i.e., for a fixed L, we may need to extend (include

[3]For simplicity of the presentation, we chose here that each state variable is observed at at least one sub-system. We can easily account for this when this is not true by extending the cut-point sets, $V^{(l)}$, to $\overline{V}^{(l)}$, such that $\bigcup_{l=1}^{N} \overline{V}^{(l)} = X$.

additional states in a cut-point set) the cut-point sets, $V^{(l)}$, to $V_L^{(l)}$, such that

$$n_l = \left|V_L^{(l)}\right| \geq L, \qquad \forall\ l, \tag{6.29}$$

where $|\cdot|$ when applied to a set denotes its cardinality. This procedure of choosing an L based on a certain desired performance gives a lower bound on the dimension of each sub-system.

Coupled States as Inputs: The directed edges coming into a cut-point set are the inputs required by local model at that sub-system. In the context of our running illustration (6.26)−(6.27), we see that the local state vector for sub-system 1 is, $\mathbf{x}_k^{(1)} = [x_{1,k}, x_{2,k}, x_{3,k}]^T$, and the inputs to the local model consist of a subset of the state set, X, (at sub-system 1, $x_{4,k}$ is the input coming from sub-system 2) and a subset of the noise input set, U, ($u_{2,k}$ at sub-system s_1).

Local models: For the local model at sub-system l, we collect the states required as input in a local internal input vector, $\mathbf{d}_k^{(l)}$ (we use the word *internal* to distinguish from the externally applied inputs), and the noise sources required as input in a local noise input vector, $\mathbf{u}_k^{(l)}$. We collect the elements from $\mathbf{F}$ corresponding to the local state vector, $\mathbf{x}_k^{(l)}$, in a local model matrix, $\mathbf{F}^{(l)}$. Similarly, we collect the elements from $\mathbf{F}$ corresponding to the local internal input vector, $\mathbf{d}_k^{(l)}$, in a local internal input matrix, $\mathbf{D}^{(l)}$, and the elements from $\mathbf{G}$ corresponding to the local noise input vector, $\mathbf{u}_k^{(l)}$, in a local state noise matrix, $\mathbf{G}^{(l)}$. Fig. 6.1(c) shows this partitioning for sub-systems 1 and 3. We have the following local models from(6.26).

$$\begin{aligned}
\mathbf{x}_{k+1}^{(1)} &= \begin{bmatrix} f_{11} & f_{12} & 0 \\ f_{21} & f_{22} & 0 \\ f_{31} & 0 & f_{33} \end{bmatrix} \mathbf{x}_k^{(1)} + \begin{bmatrix} 0 \\ f_{24} \\ 0 \end{bmatrix} x_{4,k} + \begin{bmatrix} 0 \\ 0 \\ g_{32} \end{bmatrix} u_{2,k}, \\
&= \mathbf{F}^{(1)}\mathbf{x}_k^{(1)} + \mathbf{D}^{(1)}\mathbf{d}_k^{(1)} + \mathbf{G}^{(1)}\mathbf{u}_k^{(1)}.
\end{aligned} \tag{6.30}$$

$$\begin{aligned}
\mathbf{x}_{k+1}^{(2)} &= \begin{bmatrix} f_{22} & 0 & f_{42} \\ 0 & f_{33} & 0 \\ 0 & f_{43} & 0 \end{bmatrix} \mathbf{x}_k^{(2)} + \begin{bmatrix} f_{21} & 0 \\ f_{31} & 0 \\ 0 & f_{45} \end{bmatrix} \begin{bmatrix} x_{1,k} \\ x_{5,k} \end{bmatrix} + \begin{bmatrix} 0 \\ g_{32} \\ 0 \end{bmatrix} u_{2,k}, \\
&= \mathbf{F}^{(2)}\mathbf{x}_k^{(2)} + \mathbf{D}^{(2)}\mathbf{d}_k^{(2)} + \mathbf{G}^{(2)}\mathbf{u}_k^{(2)} \qquad (6.31) \\
\mathbf{x}_{k+1}^{(3)} &= \begin{bmatrix} 0 & f_{45} \\ f_{54} & f_{55} \end{bmatrix} \mathbf{x}_k^{(3)} + \begin{bmatrix} f_{43} \\ 0 \end{bmatrix} x_{3,k} + \begin{bmatrix} 0 \\ g_{51} \end{bmatrix} u_{1,k}, \\
&= \mathbf{F}^{(3)}\mathbf{x}_k^{(3)} + \mathbf{D}^{(3)}\mathbf{d}_k^{(3)} + \mathbf{G}^{(3)}\mathbf{u}_k^{(3)} \qquad (6.32)
\end{aligned}$$

We may also capture the above extraction of the local states by the cut-point sets, with the following procedure. Let the total number of states in the cut-point set at sub-system l, $V^{(l)}$, be n_l. Let $\mathbf{T}_l$ be an $n_l \times n$ selection matrix, such that it selects n_l states in the cut-point set, $V^{(l)}$, from the entire state vector, $\mathbf{x}_k$, according to the following relation,

$$\mathbf{x}_k^{(l)} = \mathbf{T}_l \mathbf{x}_k. \qquad (6.33)$$

For example, the selection matrix, $\mathbf{T}_1$ at sub-system 1, is

$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \qquad (6.34)$$

We establish a *reduced* local observation matrix, $\mathbf{H}^{(l)}$, by retaining the terms corresponding to the local state vector, $\mathbf{x}_k^{(l)}$, from the local observation matrix, $\mathbf{H}_l$. We may write

$$\mathbf{H}^{(l)} = \mathbf{H}_l \mathbf{T}_l^{\#}, \qquad (6.35)$$

where '#' denotes the pseudo-inverse of the matrix. In the context of our running illustration, the reduced local observation matrix $\mathbf{H}^{(1)} = [1, 1, 1]$ is obtained from the local observation matrix $\mathbf{H}_1 = [1, 1, 1, 0, 0]$. Note that $\mathbf{H}_l$ picks the states from the global state vector, $\mathbf{x}_k$, whereas $\mathbf{H}^{(l)}$ picks the states from the local state vector, $\mathbf{x}_k^{(l)}$.

The reduced local observation models are given by

$$\mathbf{y}_k^{(l)} = \mathbf{H}^{(l)}\mathbf{x}_k^{(l)} + \mathbf{w}_k^{(l)}. \tag{6.36}$$

We now make some additional comments. For simplicity of the explanation, we refer to our running example, (6.26)−(6.27). We note that the sub-systems overlap, as shown by the overlapping cut-point sets in Fig. 6.1(b). Due to this overlap, observations corresponding to the shared states are available at multiple sub-systems that should be fused. We further note that the local model (6.30) at sub-system 1 is coupled to the local model (6.31) at sub-system 2 through the state $x_{4,k}$. The state $x_{4,k}$ at sub-system 1 does not appear in the local state vector, i.e., $x_{4,k} \notin \mathbf{x}_k^{(1)}$. But, it is still required as an internal input at sub-system 1 to preserve the global dynamics. Hence, sub-system 2 communicates the state $x_{4,k}$, which appears in its local state vector, i.e., $x_{4,k} \in \mathbf{x}_k^{(2)}$, to sub-system 1. Hence at an arbitrary sub-system l, we derive the reduced model to be

$$\mathbf{x}_{k+1}^{(l)} \;=\; \mathbf{F}^{(l)}\mathbf{x}_k^{(l)} + \mathbf{D}^{(l)}\mathbf{d}_k^{(l)} + \mathbf{G}^{(l)}\mathbf{u}_k^{(l)}. \tag{6.37}$$

Since the value of the state itself is unknown, sub-system 2 communicates its estimate, $\widehat{x}_{4,k|k}^{(2)}$, to sub-system 1. This allows sub-system 1 to complete its local model and preserve global dynamics, thus, taking into account the coupling sub-systems. This process is repeated at all sub-systems. Hence, the local internal input vector, $\mathbf{d}_k^{(l)}$, is replaced by its estimate, $\widehat{\mathbf{d}}_{k|k}^{(l)}$. It is worth mentioning here that if the dynamics were time-dependent, i.e., the matrices, $\mathbf{F}, \mathbf{G}, \mathbf{H}$, change with time, k, then the above decomposition procedure will have to be repeated at each k. This may result into a different communication topology over which the sub-systems communicate at each k.

6.3.2 Local Information filters

To distribute the estimation of the global state vector, $\mathbf{x}_k$, we implement local Information filters (LIFs) at each sub-system l, which are based on the sub-system models (6.37) and (6.36). Each LIF computes local quantities (matrices and vectors

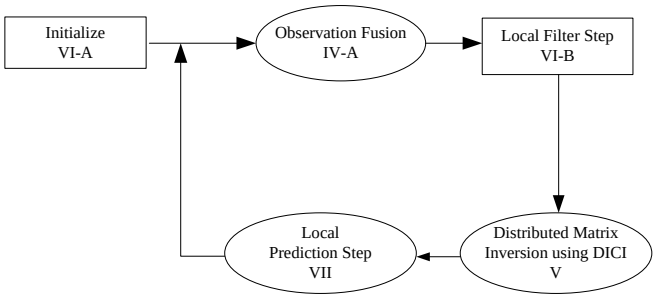


Figure 6.2: Block Diagram for the LIFs: Steps involved in the LIF implementation. The ovals represent the steps that require local communication.

of dimension n_l), which are then fused (if required) by exchanging information among the neighbors. Some of the update procedures are iterative. Although, no centralized knowledge of the estimation of the global state exists, the union of the local state vector represents, in a distributed way, the knowledge that exists in a centralized fashion in the CLBIF. In most applications nowhere in the network is there the need for this centralized knowledge.

The LIFs consist of initial conditions, a local filter step (including observation fusion and distributed matrix inversion) and a local prediction step (including estimate fusion), see Fig. 6.2. These steps are presented in the next four sections. To proceed with the next sections, we provide notation in the following Subsection.

Notation

The superscript (l) refers to a local reduced-order variable ($n_l \times 1$ vector or $n_l \times n_l$ matrix) at sub-system l. For example, the reduced observation vector, $\mathbf{i}_k^{(l)}$, and the reduced observation matrix, $\mathcal{I}^{(l)}$, are

$$\mathbf{i}_k^{(l)} = (\mathbf{H}^{(l)})^T \mathbf{R}_l^{-1} \mathbf{y}_k^{(l)}, \tag{6.38}$$

$$\mathcal{I}^{(l)} = (\mathbf{H}^{(l)})^T \mathbf{R}_l^{-1} \mathbf{H}^{(l)}. \tag{6.39}$$

The local error covariance matrices, $\mathbf{S}_{k|k}^{(l)}$ and $\mathbf{S}_{k|k-1}^{(l)}$, are the overlapping diagonal submatrices of the global error covariance matrices, $\mathbf{S}_{k|k}$ and $\mathbf{S}_{k|k-1}$. Let $\mathbf{Z}_{k|k}^{(l)}$ and $\mathbf{Z}_{k|k-1}^{(l)}$

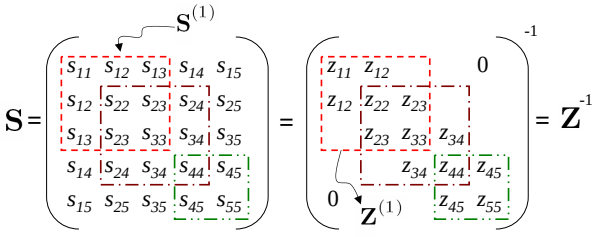


Figure 6.3: Relationship between the global error covariance matrices, $\mathbf{S}$ and their inverses, the global information matrices, $\mathbf{Z}$, with $L = 1$-banded approximation on $\mathbf{Z}$. The figure also shows how the local matrices, $\mathbf{S}^{(l)}$ and $\mathbf{Z}^{(l)}$, constitute their global counterparts. Since this relation holds for both the estimation and prediction matrices, we remove the subscripts.

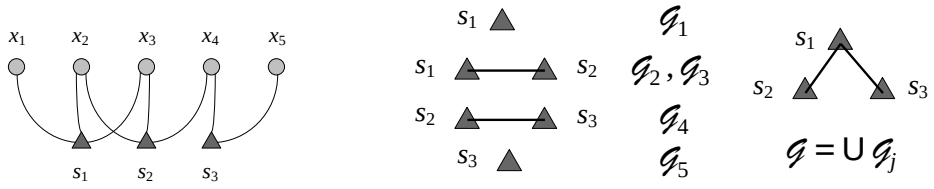
be the local information matrices. These local information matrices are overlapping diagonal sub-matrices of the global L-banded information matrices, $\mathbf{Z}_{k|k}$ and $\mathbf{Z}_{k|k-1}$. These local matrices overlap because the sub-systems overlap. Fig. 6.3 captures the relationship between the local error covariance matrices and the local information matrices given by (6.12) and (6.13).

6.4 Overlapping reduced models

After the spatial decomposition of the dynamical system, introduced in Section 6.3, the resulting sub-systems share state variables, as shown by the overlapped cut-point sets in Fig. 6.1(b). Since the sub-systems sharing the states have independent observations of the shared states, observations corresponding to the shared states should be fused. We present observation fusion in subsection 6.4.1 with the help of bipartite fusion graphs, [46].

6.4.1 Observation fusion

Equations (6.20) and (6.21) show that the observation fusion is equivalent to adding the corresponding n-dimensional local observation variables, (6.18)−(6.19). In CLBIF, we implement this fusion directly because each local observation variable in (6.18)−(6.19)



(a) (b)

Figure 6.4: (a) A bipartite Fusion graph, $\mathcal{B}$, is shown for the example system. (b) Subgraphs, $\mathcal{G}_j$, for observation fusion.

corresponds to the full n-dimensional state vector, $\mathbf{x}_k$. Since the n_l-dimensional *reduced* observation variables, (6.38)−(6.39), correspond to different local state vectors, $\mathbf{x}_k^{(l)}$, they cannot be added directly.

For simplicity and without loss of generality, we assume each local observation matrix, $\mathbf{H}_i$, to be a row. To achieve observation fusion, we introduce the following undirected bipartite fusion graph[4], $\mathcal{B}$. Let $S_N = \{s_1, \ldots, s_N\}$ be the set of sensors and X be the set of states. The vertex set of the bipartite fusion graph, $\mathcal{B}$, is $S_N \cup X$. We now define the edge set, $E_{\mathcal{B}}$, of the fusion graph, $\mathcal{B}$. The sensor s_i is connected to the state variable x_j, if s_i observes (directly or as a linear combination) the state variable x_j. In other words, we have an edge between sensor s_i and state variable x_j, if the local observation matrix, $\mathbf{H}_i$, at sensor s_i, contains a non-zero entry in its jth column. Fig. 6.4(a) shows the bipartite graph for the example system in (6.26)−(6.27). The set of sensors, S_j, that observe the jth state, x_j, come directly form the bipartite fusion graph, $\mathcal{B}$. For example, from Fig. 6.4(a), we see that S_1 contains s_1 as a single sensor, whereas S_2 contains the sensors s_1, s_2, and so on[5]. States having more than one sensor connected to them in the bipartite fusion graph, $\mathcal{B}$, are the states for which fusion is required, since we have multiple observations for that state.

[4]A bipartite graph is a graph whose vertices can be divided into two disjoint sets X and S_N, such that every edge connects a vertex in X to a vertex in S_N, and there is no edge between any two vertices of the same set, [151].

[5]Mathematically, this can be captured as follows. Let the number of non-zero elements in the jth column, $\mathbf{h}_j$, of the global observation matrix, $\mathbf{H}$, be given by N_j and let Ω_j be the set of the locations of these non-zero elements, with $|\Omega_j| = N_j$ and $\Omega_{j,\varsigma}$ be its ςth element. Let $\mathbf{V}_j$ be an $N_j \times N$ matrix with ones on the $(\varsigma, \Omega_{j,\varsigma})$-th locations and zeros elsewhere. Then $S_j = \mathbf{V}_j S_N$.

With the help of the above discussion, we establish the fusion of the reduced observation variables, (6.38)−(6.39). The reduced model at each sensor involves n_l state variables, and each element in the $n_l \times 1$ reduced observation vector, $\mathbf{i}_k^{(l)}$, corresponds to one of these states, i.e., each entry in $\mathbf{i}_k^{(l)}$ has some information about its corresponding state variable. Let the entries of the $n_l \times 1$ reduced observation vector, $\mathbf{i}_k^{(l)}$, at sensor l, be subscripted by the n_l state variables modeled at sensor l. In the context of the example given by system (6.26)−(6.27), we have

$$\mathbf{i}_k^{(1)} = \begin{bmatrix} i_{k,x_1}^{(1)} \\ i_{k,x_2}^{(1)} \\ i_{k,x_3}^{(1)} \end{bmatrix}, \qquad \mathbf{i}_k^{(2)} = \begin{bmatrix} i_{k,x_2}^{(2)} \\ i_{k,x_3}^{(2)} \\ i_{k,x_4}^{(2)} \end{bmatrix}, \qquad \mathbf{i}_k^{(3)} = \begin{bmatrix} i_{k,x_4}^{(3)} \\ i_{k,x_5}^{(3)} \end{bmatrix}. \tag{6.40}$$

For the jth state, x_j, the observation fusion is carried out on the set of sensors, S_j, attached to this state in the bipartite fusion graph, $\mathcal{B}$. The fused observation vectors denoted by $\mathbf{i}_{f,k}^{(l)}$ are given by

$$\mathbf{i}_{f,k}^{(1)} = \begin{bmatrix} i_{k,x_1}^{(1)} \\ i_{k,x_2}^{(1)} + i_{k,x_2}^{(2)} \\ i_{k,x_3}^{(1)} + i_{k,x_3}^{(2)} \end{bmatrix}, \quad \mathbf{i}_{f,k}^{(2)} = \begin{bmatrix} i_{k,x_2}^{(2)} + i_{k,x_2}^{(1)} \\ i_{k,x_3}^{(2)} + i_{k,x_3}^{(1)} \\ i_{k,x_4}^{(2)} + i_{k,x_4}^{(3)} \end{bmatrix}, \quad \mathbf{i}_{f,k}^{(3)} = \begin{bmatrix} i_{k,x_4}^{(3)} + i_{k,x_4}^{(2)} \\ i_{k,x_5}^{(3)} \end{bmatrix}. \tag{6.41}$$

Generalizing to the arbitrary sensor l, we may write the entry, $i_{f,k,x_j}^{(l)}$, corresponding to x_j in the fused observation vector, $\mathbf{i}_{f,k}^{(l)}$ as

$$i_{f,k,x_j}^{(l)} = \sum_{s \in S_j} i_{k,x_j}^{(s)}, \tag{6.42}$$

where $i_{k,x_j}^{(s)}$ is the entry corresponding to x_j in the reduced observation vector at sensor s, $\mathbf{i}_k^{(s)}$.

6.4.2 Implementation

We now provide further notation on the communication topology to formulate the observation fusion procedure precisely. For the jth state, x_j, let $\mathcal{G}_j^{(fc)} = \{S_j, E_j^{(fc)}\}$

be a fully-connected[6] (fc) sub-graph, such that its vertices are all the sensors that observe x_j. With this definition we can define $E_j^{(fc)} = \mathbf{V}_j \overline{\left(\mathbf{I} - \overline{\mathbf{h}_j \mathbf{h}_j^T}\right)} \mathbf{V}_j^{\#}$, where $\mathbf{V}$ is given in footnote 5 and the overline denotes the binary representation, as introduced in Section 6.3.1. For the jth state, x_j, let $\mathcal{G}_j^{(c)} = \{S_j, E_j^{(c)} \subset E_j^{(fc)}\}$ be a connected[7] sub-graph. Now, the overall sensor network topology required for the observation fusion with fully-connected sub-graphs, $\mathcal{G}_j^{(fc)}$, can be given by $\mathcal{G} = \left\{S_N, \overline{\sum_j \mathbf{V}_j^{\#} E_j^{(fc)} \mathbf{V}_j} = \overline{\mathbf{I} - \overline{\mathbf{H}\mathbf{H}}^T}\right\}$ and for the connected sub-graphs, $\mathcal{G}_j^{(c)}$, can be given by $\widetilde{\mathcal{G}} = \left\{S_N, \overline{\sum_j \mathbf{V}_j^{\#} E_j^{(c)} \mathbf{V}_j}\right\}$. These graphs are shown in Fig. 6.4(b).

Remarks: We now make some comments.

(i) If we choose the overall sensor communication graph, $\mathcal{G}$, that results from the union of the fully-connected sub-graphs, $\mathcal{G}_j^{(fc)}$ $\forall j$, the observation fusion procedure *does not* require an iterative procedure and is realized in a single step.

(ii) If we choose the overall sensor communication graph, $\widetilde{\mathcal{G}}$, that results from the union of the connected sub-graphs, $\mathcal{G}_j^{(c)}$ $\forall j$, the observation fusion procedure requires an iterative consensus algortihm[8] and can be realized using an average-consensus algorithm [17].

(iii) With the assumption of a localized global observation matrix, $\mathbf{H}$, as motivated in Section 6.2.1, the overall sensor communication network, $\mathcal{G}$ or $\widetilde{\mathcal{G}}$, is not necessarily fully-connected, as shown in Fig. 6.4(b).

(iv) With any choice of the overall sensor communication graph, $\mathcal{G}$ or $\widetilde{\mathcal{G}}$, the communication required for observation fusion is single-hop.

[6] A fully-connected graph, $\mathcal{G}^{(fc)} = \{S_N, E^{(fc)}\}$, is such that every pair of distinct vertices in S_N is connected by an edge in $E^{(fc)}$.

[7] A connected graph, $\mathcal{G}^{(c)} = \{S_N, E^{(c)}\}$, is such that there exists a path from any vertex in S_N to any other vertex in S_N.

[8] In this case, we assume that the communication is fast enough so that the consensus algorithm can converge, see [152] for a discussion on distributed Kalman filtering based on consensus strategies. The convergence of the consensus algorithm is shown to be geometric and the convergence rate can be increased by optimizing the weight matrix for the consensus iterations using semidefinite programming [17]. The communication topology of the sensor network can also be improved to increase the convergence speed of the consensus algorithms [153].

(v) It is worth mentioning that the observation fusion procedure implemented in [37] cannot be realized in single-step unless the overall sensor communication graph, $\mathcal{G}$, is fully-connected, which we *do not* require anywhere in our solution.

A similar procedure on the pairs of state variables and their associated subgraphs can be implemented to fuse the reduced observation matrices, $\mathcal{I}^{(l)}$. Since we assume the observation model to be stationary ($\mathbf{H}$ and $\mathbf{R}$ are time-independent), the fusion on the reduced observation matrix, $\mathcal{I}^{(l)}$, is to be carried out only once and can be an offline procedure. If that is not the case, and $\mathbf{H}$ and $\mathbf{R}$ are time dependent, fusion on $\mathcal{I}$ has to be repeated at each time, k.

A comment on estimate fusion. Since we fuse the observations concerning the shared states among the sensors, one may ask if it is required to carry out fusion of the estimates of the shared states. It turns out that consensus on the observations leads to consensus on the estimates. This will become clear with the introduction of the local filter and the local prediction step of the LIFs, therefore, we defer the discussion on estimate fusion to Section 6.7.3.

6.5 Distributed matrix inversion with local communication

In this section, we discuss the cooperative assimilation procedure on the local error covariances. Consider the example model (6.26)−(6.27), when we employ LIFs on the distributed models (6.30)−(6.32). The local estimation information matrices, $\mathbf{Z}^{(1)}_{k|k}$, $\mathbf{Z}^{(2)}_{k|k}$, and $\mathbf{Z}^{(3)}_{k|k}$, correspond to the overlapping diagonal sub-matrices of the global 5×5 estimation information matrix, $\mathbf{Z}_{k|k}$, see Fig. 6.3, with $L = 1$-banded approximation on $\mathbf{Z}_{k|k}$. It will be shown (Section 6.7.1) that the local prediction information matrix, $\mathbf{Z}^{(l)}_{k+1|k}$, is a function of the local error covariance matrices, $\mathbf{S}^{(l)}_{k|k}$, and hence we need to compute $\mathbf{S}^{(l)}_{k|k}$ from the local filter information matrices, $\mathbf{Z}^{(l)}_{k|k}$, which we get from the local filter step (Section 6.6). As can be seen from Fig. 6.3

and (6.12), for these local sub-matrices,

$$\mathbf{S}^{(l)} \neq \left(\mathbf{Z}^{(l)}\right)^{-1}. \tag{6.43}$$

Collecting all the local information matrices, $\mathbf{Z}_{k|k}^{(l)}$, at each sensor and then carrying out an $n \times n$ matrix inversion is *not* a practical solution for large-scale systems (where n may be large), because of the large communication overhead and $O(n^3)$ computational cost. Using the L-banded structure on the global estimation information matrix, $\mathbf{Z}_{k|k}$, we employ the DICI algorithm (Chapter 5) to compute the inverse of $\mathbf{Z}_{k|k}$ in a distributed fashion.

6.6 Local Information filters: Initial conditions and local filter step

The initial conditions and the local filter step of the LIFs are presented in the next subsections.

6.6.1 Initial conditions

The initial condition on the local predictor is

$$\widehat{\mathbf{z}}_{0|-1}^{(l)} = \mathbf{0}. \tag{6.44}$$

Since the local information matrix and the local error covariances are not the inverse of each other, (6.43), we obtain the initial condition on the prediction information matrix by using the L-banded inversion theorem [120], provided in Appendix C.1. This step may require a local communication step further elaborated in Section 6.7.1.

$$\mathbf{Z}_{0|-1}^{(l)} \xleftarrow{\qquad L\text{-Banded Inversion Theorem}\qquad} \mathbf{S}_{0}^{(l)} \tag{6.45}$$

6.6.2 Local filter step

In this section, we present the local filter step of the LIFs. The local filter step is given by

$$\mathbf{Z}_{k|k}^{(l)} = \mathbf{Z}_{k|k-1}^{(l)} + \mathcal{I}_f^{(l)}, \tag{6.46a}$$

$$\widehat{\mathbf{z}}_{k|k}^{(l)} = \widehat{\mathbf{z}}_{k|k-1}^{(l)} + \mathbf{i}_{f,k}^{(l)}, \tag{6.46b}$$

where $\mathcal{I}_f^{(l)}$ and $\mathbf{i}_{f,k}^{(l)}$ denote the fused observation variables. Fusion of the observations is presented in Section 6.4.1. The distribution of the addition operation, '+', in (6.22) is straightforward in (6.46). Recall that the observation fusion, (6.42), is carried out in single-step for the sensor communication graph, $\mathcal{G}$, or using the an average-consensus algorithm for the sensor communication graph, $\widetilde{\mathcal{G}}$. In case of $\widetilde{\mathcal{G}}$, the asymptotic convergence of this iterative algorithm is guaranteed under certain conditions, see [17] for details on such conditions. Hence, with the required assumptions on the sub-graph, $\mathcal{G}_j$, the observation fusion algorithm, (6.42), asymptotically converges, and hence (with a slight abuse of notation),

$$\bigcup_{l=1}^{N} \mathbf{i}_{f,k}^{(l)} \to \mathbf{i}_k \text{ and } \bigcup_{l=1}^{N} \mathcal{I}_f^{(l)} \to \mathcal{I}. \tag{6.47}$$

The above notation implies that the local fused information variables, $\mathcal{I}_f^{(l)}$ and $\mathbf{i}_{f,k}^{(l)}$, when combined over the entire sensor network, asymptotically converge to the global information variables, $\mathcal{I}$ and $\mathbf{i}_k$. This, in turn, implies that the local filter step of the LIFs asymptotically converges to the global filter step, (6.22), of the CLBIF.

Once the local filter step is completed, the DICI algorithm (Chapter 5) is employed on the local information matrices, $\mathbf{Z}_{k|k}^{(l)}$ obtained from (6.46a), to convert them into the local error covariance matrices, $\mathbf{S}_{k|k}^{(l)}$. Finally, to convert the estimates in the information domain, $\widehat{\mathbf{z}}_{k|k}^{(l)}$, to the estimates in the Kalman filter domain, $\widehat{\mathbf{x}}_{k|k}^{(l)}$, which is linear system of equations (6.15), we use the distributed Jacobi algorithm (Section 3.4).

6.7 Local Information filters: Local prediction step

This section presents the distribution of the global prediction step, (6.23), into the local prediction step at each LIF.

6.7.1 Computing the local prediction information matrix

Because of the coupled local dynamics of the reduced sensor-based models, each sensor may require that some of the estimated states be communicated as internal inputs, $\widehat{\mathbf{d}}_{k|k}^{(l)}$, to its LIF, as shown in (6.37). These states are the directed edges into each cut-point set in Fig. 6.1(b). Hence, the error associated to a local estimation procedure is also influenced by the error associated to the neighboring estimation procedure, from where the internal inputs are being communicated. This dependence is true for all sensors and is reflected in the local prediction error covariance matrix, $\mathbf{S}_{k|k-1}^{(l)}$, as it is a function of the global estimation error covariance matrix, $\mathbf{S}_{k-1|k-1}$. Equation (6.48) follows from (6.23a) after expanding (6.23a) for each diagonal submatrix, $\mathbf{S}_{k|k-1}^{(l)}$, in $\mathbf{S}_{k|k}$.

$$\mathbf{S}_{k|k-1}^{(l)} = \mathbf{F}_l \mathbf{S}_{k-1|k-1} \mathbf{F}_l^T + \mathbf{G}^{(l)} \mathbf{Q}^{(l)} \mathbf{G}^{(l)T}. \tag{6.48}$$

The matrix, $\mathbf{F}_l = \mathbf{T}_l \mathbf{F}$, (the matrix $\mathbf{T}_l$ is introduced in (6.33)) is an $n_l \times n$ matrix, which relates the state vector, $\mathbf{x}_k$, to the local state vector, $\mathbf{x}_k^{(l)}$. Fig. 6.1(c) shows that the matrix, $\mathbf{F}_l$, is further divided into $\mathbf{F}^{(l)}$ and $\mathbf{D}^{(l)}$. With this sub-division of $\mathbf{F}_l$, the first term on the right hand side of (6.48), $\mathbf{F}_l \mathbf{S}_{k-1|k-1} \mathbf{F}_l^T$, can be expanded, and (6.48) can be written as

$$\begin{aligned} \mathbf{S}_{k|k-1}^{(l)} &= \mathbf{F}^{(l)} \mathbf{S}_{k-1|k-1}^{(l)} \mathbf{F}^{(l)T} + \mathbf{F}^{(l)} \mathbf{S}_{k-1|k-1}^{x^{(l)}d^{(l)}} \mathbf{D}^{(l)T} \\ &+ \left(\mathbf{F}^{(l)} \mathbf{S}_{k-1|k-1}^{x^{(l)}d^{(l)}} \mathbf{D}^{(l)T} \right)^T + \mathbf{D}^{(l)} \mathbf{S}_{k-1|k-1}^{d^{(l)}d^{(l)}} \mathbf{D}^{(l)T} + \mathbf{G}^{(l)} \mathbf{Q}^{(l)} \mathbf{G}^{(l)T}, \end{aligned} \tag{6.49}$$

where:

- $\mathbf{S}_{k-1|k-1}^{(l)}$ is the local error covariance matrix, which is available from (6.46a) and the DICI algorithm at sensor l;

- $\mathbf{S}^{d^{(l)}d^{(l)}}_{k-1|k-1}$ is the local error covariance matrix, which is available from (6.46a) and the DICI algorithm at the sensors having the states, $\mathbf{d}^{(l)}_k$, in their reduced models;

- $\mathbf{S}^{x^{(l)}d^{(l)}}_{k-1|k-1}$ is the error cross correlation between the local state vector, $\mathbf{x}^{(l)}_k$, and the local internal input vector, $\mathbf{d}^{(l)}_k$.

The non L-banded entries in this matrix can be computed from equation (5.14) in Lemma 24. Since the model matrix, $\mathbf{F}$, is sparse, we do not need the entire error covariance matrix, $\mathbf{S}_{k-1|k-1}$, only certain of its sub-matrices. Since the model matrix, $\mathbf{F}$, is localized, long-distance communication is not required, and the sub-matrices are available at the neighboring sensors. Once we have calculated the local prediction error covariance matrix, $\mathbf{S}^{(l)}_{k|k-1}$, we realize (6.43) and compute the local prediction information matrix, $\mathbf{Z}^{(l)}_{k|k-1}$, using the L-banded Inversion Theorem (see [120] and Appendix C.1).

$$\mathbf{Z}^{(l)}_{k|k-1} \xleftarrow{\qquad L\text{-Banded Inversion Theorem}\qquad} \mathbf{S}^{(l)}_{k|k-1}. \tag{6.50}$$

From (C.2) in Appendix C.1, to calculate the local prediction information matrix, $\mathbf{Z}^{(l)}_{k|k-1}$, we only need the $\mathbf{S}^{(l)}_{k|k-1}$ from sensor 'l' and from some additional neighboring sensors. Hence $\mathbf{Z}^{(l)}_{k|k-1}$ is again computed with only local communication and n_lth order computation.

6.7.2 Computing the local predictor

We illustrate the computation of the local predictor, $\widehat{\mathbf{z}}^{(3)}_{k|k-1}$, for the 5-dimensional system, (6.26)−(6.27), with $L = 1$. The local predictor, $\widehat{\mathbf{z}}^{(3)}_{k|k-1}$, at sensor 3 follows from the global predictor, (6.23b), and is given by

$$\widehat{\mathbf{z}}^{(3)}_{k|k-1} = \mathbf{Z}^{(3)}_{k|k-1}\left(\mathbf{F}^{(3)}\widehat{\mathbf{x}}^{(3)}_{k-1|k-1} + \mathbf{D}^{(3)}\widehat{\mathbf{d}}^{(3)}_{k-1|k-1}\right) + \begin{bmatrix} z_{34}\left(f_{31}\widehat{x}^{(1)}_{1,k-1|k-1} + f_{33}\widehat{x}^{(2)}_{3,k-1|k-1}\right) \\ 0 \end{bmatrix}, \tag{6.51}$$

where z_{34} is the only term arising due to the $L = 1$-banded (tridiagonal) assumption on the prediction information matrix, $\mathbf{Z}_{k|k-1}$. Note that $f_{31}\widehat{x}^{(1)}_{1,k-1|k-1} + f_{33}\widehat{x}^{(2)}_{3,k-1|k-1}$ is a result of $\mathbf{f}_3\widehat{\mathbf{x}}_{k-1|k-1}$, where $\mathbf{f}_3$ is the third row of the model matrix, $\mathbf{F}$. A model matrix with a localized and sparse structure ensures that $\mathbf{f}_3\widehat{\mathbf{x}}_{k-1|k-1}$ is computed from a small subset of the estimated state vector, $\widehat{\mathbf{x}}^{(\mathcal{Q})}_{k-1|k-1}$, communicated by a subset $\mathcal{Q} \subseteq \mathcal{K}(l)$ of the neighboring sensors, which are modeling these states in their reduced models. This may require multi-hop communication.

Generalizing, the local predictor in the information domain, $\widehat{\mathbf{z}}^{(l)}_{k|k-1}$, is given by

$$\widehat{\mathbf{z}}^{(l)}_{k|k-1} = \mathbf{Z}^{(l)}_{k|k-1}\left(\mathbf{F}^{(l)}\widehat{\mathbf{x}}^{(l)}_{k-1|k-1} + \mathbf{D}^{(l)}\widehat{\mathbf{d}}^{(l)}_{k-1|k-1}\right) + f_1\left(\mathbf{Z}^{(\mathcal{V})}_{k|k-1}, \mathbf{F}^{(\mathcal{V})}, \widehat{\mathbf{x}}^{(\mathcal{Q})}_{k-1|k-1}\right) \quad (6.52)$$

for some $\mathcal{V}, \mathcal{Q} \subseteq \mathcal{K}(l)$, where $f_1(\cdot)$ is a linear function and depends on L.

6.7.3 Estimate fusion

We present the following fact, if any $\mathcal{G}_j$ is not fully connected.

Fact: Let m denote the number of iterations of the consensus algorithm that are employed to fuse the observations (recall (6.42)). As $m \to \infty$, the local estimates, $\widehat{\mathbf{z}}^{(l)}_{k|k}$, in (6.46b) also reach a consensus on the estimates of the shared states, i.e., the local estimates converge to the CLBIF estimates.

It is straightforward to note that, *if* the local predictors, $\widehat{\mathbf{z}}^{(l)}_{k|k-1}$, of the shared states are the same over the sensors that share these states, *then* as $m \to \infty$ we have a consensus on the estimates (of the shared states) in the local filter step (6.46b). To show that the shared local predictors are the same over the sensors that share these states, we refer back to our illustration and write the local predictors for sensor 2 as follows,

$$\begin{aligned}\widehat{\mathbf{z}}^{(2)}_{k|k-1} &= \mathbf{Z}^{(2)}_{k|k-1}\left(\mathbf{F}^{(2)}\widehat{\mathbf{x}}^{(2)}_{k-1|k-1} + \mathbf{D}^{(2)}\widehat{\mathbf{d}}^{(2)}_{k-1|k-1}\right) \\ &+ \begin{bmatrix} z_{12}\left(f_{11}\widehat{x}^{(1)}_{1,k-1|k-1} + f_{12}\widehat{x}^{(2)}_{2,k-1|k-1}\right) \\ 0 \\ z_{45}\left(f_{54}\widehat{x}^{(2)}_{4,k-1|k-1} + f_{55}\widehat{x}^{(3)}_{5,k-1|k-1}\right) \end{bmatrix}. \end{aligned} \quad (6.53)$$

The predictor for the shared state $x_{4,k}$ can now be extracted for sensor 3 from (6.51) and for sensor 2 from (6.53) and can be verified to be the following for each $l = 2, 3$.

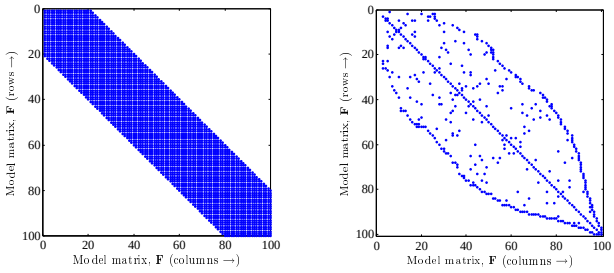
$$\begin{aligned} \widehat{z}^{(l)}_{4,k|k-1} &= z_{34} f_{31} \widehat{x}^{(1)}_{1,k-1|k-1} + (z_{34} f_{33} + z_{44} f_{43}) \widehat{x}^{(2)}_{3,k-1|k-1} \\ &+ z_{45} f_{54} \widehat{x}^{(3)}_{4,k-1|k-1} + (z_{44} f_{45} + z_{45} f_{55}) \widehat{x}^{(3)}_{5,k-1|k-1} \end{aligned} \tag{6.54}$$

The elements z_{ij} belong to the prediction information matrix, which is computed using the DICI algorithm and the L-banded inversion theorem. It is noteworthy that the DICI algorithm is not a consensus algorithm and thus the elements z_{ij} are the same across the sensor network at any iteration of the DICI algorithm. With the same local predictors, the iterations of the consensus algorithm on the observations lead to a consensus on the shared estimates.

6.8 Results

6.8.1 Summary of the Local Information filters (LIFs)

We summarize the distributed local Information filters. The initial conditions are given by (6.44) and (6.45). Observation fusion is carried out using (6.42). The fused observation variables, $\mathbf{i}^{(l)}_{f,k}$ and $\mathbf{I}^{(l)}_{f,k}$, are then employed in the local filter step, (6.46a) and (6.46b), to obtain the local information matrix and the local estimator, $\widehat{\mathbf{Z}}^{(l)}_{k|k}$ and $\mathbf{z}^{(l)}_{k|k}$, respectively. We then implement the DICI algorithm (Chapter 5) to compute the local error covariance matrix, $\mathbf{S}^{(l)}_{k|k}$, from the local information matrix, $\mathbf{Z}^{(l)}_{k|k}$. We employ the distributed Jacobi algorithm (Section 3.4) to compute the local estimates in the Kalman filter domain, $\widehat{\mathbf{x}}^{(l)}_{k|k}$, from the local estimator, $\widehat{\mathbf{z}}^{(l)}_{k|k}$. Finally, the local prediction step is completed by computing the local prediction error covariance matrix, $\widehat{\mathbf{S}}^{(l)}_{k|k-1}$, the local prediction information matrix, $\widehat{\mathbf{Z}}^{(l)}_{k|k-1}$, and, the local predictor, $\widehat{\mathbf{z}}^{(l)}_{k|k-1}$, from (6.49), (6.50), and (6.52), respectively.



(a) (b)

Figure 6.5: (a & b) Non-zero elements (chosen at random) of 100×100, $L = 20$-banded (Fig. 6.5(a)) and $L = 36$-banded (Fig. 6.5(b)) model matrices, $\mathbf{F}$, such that $||\mathbf{F}||_2 = 1$.

6.8.2 Simulations

We simulate an $n = 100$-dimensional system with $N = 10$ sensors monitoring the system. Figures 6.5(a) and 6.5(b) show the non-zero elements (chosen at random) of the model matrix, $\mathbf{F}$, such that its maximum eigenvalue is unity, i.e., $\max_{i\lambda_i(\mathbf{F})} = 1$. The model matrix in Fig. 6.5(a) is $L = 20$-banded. The model matrix in Fig. 6.5(b) is $L = 36$-banded that is obtained by employing the reverse Cuthill-Mckee algorithm [1] for bandwidth reduction of a sparse random $\mathbf{F}$. The non-zeros (chosen at random as Normal$(0, 1)$) of the global observation matrix, $\mathbf{H}$, are shown in Fig. 6.6. The lth row of the global 10×100 observation matrix, $\mathbf{H}$, is the local observation matrix, $\mathbf{H}_l$, at sensor l. Distributed Kalman filters are implemented on (i) $\mathbf{F}$ in 6.5(a) and $\mathbf{H}$ in Fig. 6.6; and (ii) $\mathbf{F}$ in 6.5(b) and $\mathbf{H}$ in Fig. 6.6. The trace of the error covariance matrix, $\mathbf{S}_{k|k}$, is simulated for different values of L in $[n, 1, 2, 5, 10, 15, 20]$ and the plots are shown (after averaging over 1000 Monte Carlo trials) in Fig. 6.7(a) for case (i); and in Fig. 6.7(b) for case (ii). The stopping criteria for the DICI algorithm and the consensus algorithm are such that the deviation in their last 10 iterations is less than 10^{-5}. In both Fig. 6.7(a) and Fig. 6.7(b), $\text{tr}(S_{k|k})$ represents the trace of the solution of the Riccati equation in the CIF (no approximation).

With 1000 Monte Carlo trials, we further simulate the trace of the error covariance, $\text{tr}(S_{k|k})$, for case (ii) with $L = 20$-banded approximation (on the information

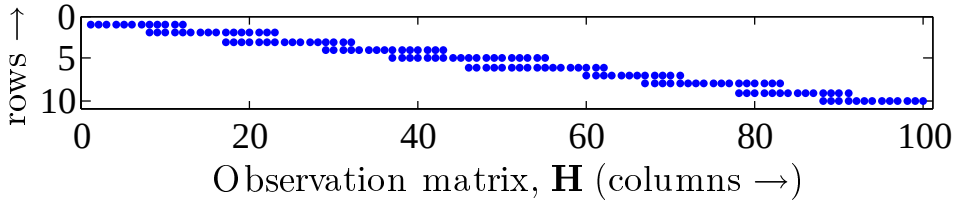


Figure 6.6: Global observation matrix, $\mathbf{H}$. The non-zero elements (chosen at random) are shown. There are $N = 10$ sensors, where the lth row of $\mathbf{H}$ corresponds to the local observation matrix, $\mathbf{H}_l$, at sensor l. The overlapping states (for which fusion is required) can be seen as the overlapping portion of the rows.

matrices) as a function of the number of iterations, t, of the DICI algorithm. We compare this with (a) the simulation obtained from the $O(n^3)$ direct inverse of the the error covariance (with $L = 20$-banded approximation on its inverse); and (b) tr $\left(\mathbf{S}_{k|k}\right)$, trace of the solution of the Riccati equation of the CIF (no approximation). We choose $t = [1, 10, 30, 100, 200]$ for the DICI algorithm and show the results in Fig. 6.8. As $t \uparrow$, the curves we obtain from the DICI algorithm get closer to the curve we obtain with the direct inverse.

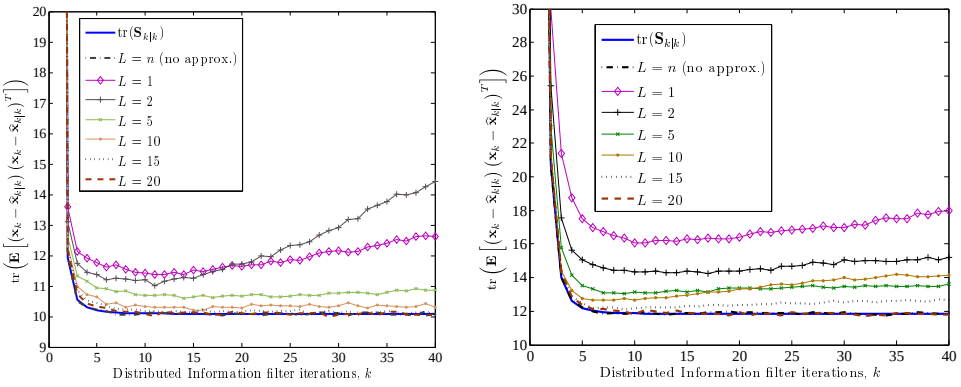
The simulations confirm the following:

(i) The LIFs asymptotically track the results of the CLBIF, see Fig. 6.8.

(ii) We verify that as $L \uparrow$, the performance is virtually indistinguishable from that of the CIF, as pointed out in [121]; this is in agreement with the fact that the approximation is optimal in Kullback-Leibler sense, as shown in [120]. Here, we also point out that, as we increase L, the performance increases, but, we pay a price in terms of the communication cost, as we may have to communicate in a larger neighborhood.

(iii) In Fig. 6.7(a) and Fig. 6.7(b), it can be verified that L should be greater than some L_{min} for the CLBIF to converge, as pointed out in Section 6.2.3.

(iv) Since, the trace of the error covariance is the trace of an expectation operator, we use Monte Carlo trials to simulate the expectation operator. If we increase



(a) (b)

Figure 6.7: (a & b) Distributed Kalman filter is implemented on the model matrices in Fig. 6.5(a)-6.5(b) and the global observation matrix, $\mathbf{H}$ (Fig. 6.6), in Fig. 6.7(a)-6.7(b). The expectation operator in the trace (on horizontal axis) is simulated over 1000 Monte Carlo trials.

the number of Monte Carlo trials the variations reduce, and the filters eventually follow the solution of the Riccati equation, tr $\left(\mathbf{S}_{k|k}\right)$.

(v) The curve in Fig. 6.8 with $t = 1$ shows the decoupled LIFs, when the local error covariances are not assimilated. This investigates the case where the distributed estimation scheme fails because the covariances at the local filters are decoupled. We next discuss the computational advantage of the LIFs over some of the existing methods.

6.8.3 Complexity

We regard the multiplication of two $n \times n$ matrices as an $O(n^3)$ operation, inversion of an $n \times n$ matrix also as an $O(n^3)$ operation, and multiplication of an $n \times n$ matrix and an $n \times 1$ vector as an $O(n^2)$ operation. For all of the following, we assume N sensors monitoring the global system, (6.6).

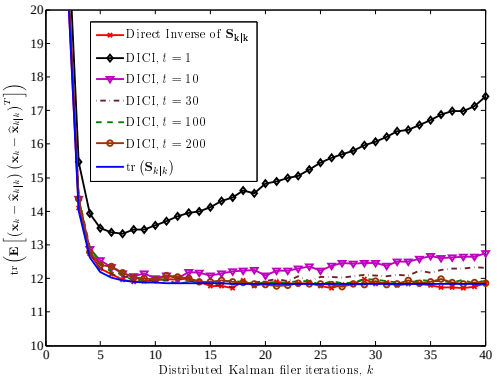


Figure 6.8: Performance of the DICI algorithm as a function of the number of DICI iterations, t.

Centralized Information Filter, CIF

This is the case where each sensor sends its local observation vector to a centralized location or a fusion center, where the global observation vector is then put together. The fusion center then implements the CIF, with an $O(n^3)$ computation complexity for each time k, so we have the complexity as $O(n^3k)$, with inordinate communication requirements (back and forth communication between the sensors and the central location).

Information filters with replicated models at each sensor and observation fusion

In this scheme, the local observation vectors are not communicated to the central location, but are fused over an all-to-all communication network, [126], or an arbitrary network, [37]. Computational complexity at each sensor is $O(n^3k)$ in [126]. In [37], let the complexity of the iterative algorithm be $O(\varphi(n))$ and let t_φ be the number of iterations of the consensus algorithm. Each sensor implements a CIF after fusing the observations. For each time index, k, each sensor requires $O(n^3)$ operations plus the operations required for the consensus algorithm, which are $O(\varphi(n)t_\varphi)$; so the total computation complexity is $O((n^3 + \varphi(n)t_\varphi)k)$ at each sensor. Communication requirements are global in [126], and local in [37].

Distributed Kalman filters: Local Information Filters, LIFs

The distributed Kalman filter presented in this chapter has three iterative algorithms. In all other steps, the computation complexity is dominated by $O(n_l^3)$, where $n_l \ll n$. Let t_o be the iterations required by the weighted averaging algorithm, where at each step of the iterative algorithm the computations are dominated by $O(n_l^2)$. Let t_{J_1} be the iterations required by the DICI algorithm for vectors, where at each step of the iterative algorithm the computations are dominated by an $O(L^2)$ operations. Let t_{J_2} be the iterations required by the DICI algorithm, where at each step of the iterative algorithm the computations are dominated by $O(L^4)$ operations. Recalling that $L \sim n_l$ from Section 6.3, the total computation complexity is $O((n_l^3+n_l^2t_o+n_l^2t_{J_1}+n_l^4t_{J_2})k)$. Let $t_\varpi = \max(t_o, t_{J_1}, t_{J_2})$, then the computation complexity is bounded by $O(n_l^4 t_\varpi k)$ at each sensor for the LIFs, which is much smaller than the computational cost of the solutions in 6.8.3 and 6.8.3. The communication requirement in the LIFs may be multi-hop but is always constrained to a neighborhood because of the structural assumptions on the model matrix, $\mathbf{F}$.

6.9 Conclusions

In this chapter, we present a *distributed* implementation of the Kalman filter for sparse large-scale systems monitored by sensor networks. In our solution, the communication, computing, and storage is local and distributed across the sensor network, no single sensor processes n-dimensional vectors or matrices, where n, usually a large number, is the dimension of the state vector representing the random field. We achieve this by solving three linked problems: (1) *Spatial decomposition* of the global dynamical system into sub-systems. These sub-systems are obtained using a graph-theoretic model distribution technique; (2) *Fusing*, through distributed averaging, multiple sensor observations of the state variables that are common across sub-systems; and (3) *Inverting* full n-dimensional matrices, with local communication only, by using the distributed iterate-collapse inversion (DICI) algorithm. The DICI algorithm only requires matrices and vectors of order n_l of the reduced state vectors.

The DICI algorithm preserves the coupling among the local Kalman filters. Our solution is optimal when the error processes of the Kalman filter are constrained to Gauss-Markov random processes and contrasts with existing Kalman filter solutions for sensor networks that either replicate an n-dimensional Kalman filter at each sensor or reduce the model dimension at the expense of decoupling the field dynamics into lower-dimensional models. The former are infeasible for large-scale systems and the latter are not optimal and further cannot guarantee any structure of the centralized error covariances.

Simulations show that the distributed implementation with local Kalman filters implemented at each sensor converges to the global Kalman filter as the bandwidth, L, of the approximated information matrices is increased.

Chapter 7

Applications to smart grids

In this chapter, we apply the theory developed in this thesis to distributed estimation and inference in electric power grids. Future power grids (also referred to as *smart grids*) are envisioned to be a diverse mix of conventional (coal, gas) and renewable (wind, solar) energy sources. We view the smart grid as a network of such diverse inter-connected modules aided by distributed sensing and communication methodologies. In this context, the techniques developed in this thesis for information processing in large-scale networks provide a foundation for a computationally efficient, scalable, and robust operation of the smart grid.

A key question, in this regard, is appropriate modeling of the electric power system to reflect that it possess certain structural properties. To this end, we present a structure-preserving model that we term as *cyber-physical model* of the electric power grid. The cyber-physical model that we present is a hybrid of data-driven models and physics-based dynamical models. The data-driven models capture the dynamics of the system modules that cannot be modeled from basic principles, e.g., aggregate system load, whereas, the physics-based models describe the dynamics of system modules that are modeled from the underlying physics, e.g., generators, modeled by their Partial Differential Equations (PDEs). Based on the cyber-physical model, we present a computationally-efficient, distributed estimation algorithm for electric power systems, along the lines of Chapter 6.

Finally, we present a novel inference technique to estimate the phasors in electric

power grids by using a minimal number of optimally-placed Phasor Measurement Units (PMUs). Our algorithm is based on the sensor localization algorithm, DILOC (Chapter 4), specialized to one-dimensional localization.

Parts of this chapter have been presented in [49, 52, 42].

We now describe the rest of the chapter. In Section 7.1, we develop cyber-physical model of the electric power system. Section 7.2 describes distributed estimation in electric power systems based on the cyber-physical model. Finally, Section 7.3 presents a distributed inference algorithm for phase-angle estimation.

7.1 Cyber-physical model of the power system

In this section, we present a cyber-physical model for the electric power system. Consider an electric power network with K steam-turbine generators[1] and M loads where the loads are considered to be aggregated loads at the sub-station level. We model the generators using their Partial Differential Equation (PDE) descriptors. On the other hand, modeling the electric load from the underlying physics is a practically impossible task. This is because modeling the electric load in a typical energy system, with millions of diverse components ranging from appliances in residential households through medium to large-size industrial and commercial consumers, is highly non-trivial. To avoid this modeling difficulty, we postulate a cyber model for the electrical load based on sensor-based identification. We then combine the cyber and physical models of all of the system modules as they are interconnected via the electric transmission network. Below, we explain the procedure.

7.1.1 Physics based description of the generator module

We model the dynamics of the generators as a governor-turbine-generator (G-T-G) set [154]. The generator dynamics in discrete-time are given by (using a standard

[1]Here we use steam-turbine generators, but the results are generalizable to any other kind.

approximation of first order derivatives)

$$\begin{aligned}
\mathbf{x}_{g,k+1} &= \left(\mathbf{I} + \Delta_T \begin{bmatrix} -\frac{D}{J} & \frac{1}{J} & \frac{e_T}{J} \\ 0 & -\frac{1}{T_u} & \frac{K_T}{T_u} \\ -\frac{1}{T_g} & 0 & -\frac{r}{T_g} \end{bmatrix} \right) \mathbf{x}_{g,k} \\
&+ \begin{bmatrix} -\frac{\Delta_T}{J} \\ 0 \\ 0 \end{bmatrix} P_{g,k} + \begin{bmatrix} 0 \\ 0 \\ \Delta_T \omega_g^{\text{ref}} \end{bmatrix} + \mathbf{u}_{g,k}, \\
&= \mathbf{F}_g \mathbf{x}_{g,k} + \mathbf{c}_g P_{g,k} + \mathbf{b}_g + \mathbf{u}_{g,k}, \qquad (7.1)
\end{aligned}$$

where Δ_T denotes the sampling rate,

$$\mathbf{x}_{g,k} = [\omega_{g,k} \; P_{T,k} \; a_k]^T, \qquad (7.2)$$

collects of the generator's frequency, mechanical power, and valve opening. The vector, $\mathbf{x}_{g,k}$, is the state vector of the steam-turbine generator at time k. The generator's parameters, D, J, K_T, T_u, T_g, are the damping coefficient, moment of inertia, and the time constants of the turbine and the generator, respectively, $P_{g,k}$ is the power supplied by the generator, and $\mathbf{u}_{g,k} \sim \mathscr{N}(\mathbf{0}, \mathbf{Q})$ is a white noise input vector.

7.1.2 Sensor based identification of the load module

We characterize the electric load modules much as the same way as the G-T-G set is characterized. We postulate a cyber model for the electric load based on a Newton-like representation of load dynamics governed by the instantaneous mismatch between the power delivered to the load, $P_{L,k}$ at time k, and the power consumed at the load, L_k at time k. We have (after discretization using a standard approximation of the first order derivatives)

$$\omega_{L,k+1} = (1 - \Delta_T \frac{D_L}{J_L})\omega_{L,k} - \frac{\Delta_T}{J_L} P_{L,k} - \frac{1}{J_L} L_k + u_{L,k}, \qquad (7.3)$$

where J_L and D_L refer to the effective moment of inertia and the damping coefficient of the aggregate load[2]. We model the load consumed, L_k, by an auto-regressive (AR) process of order p driven by zero-mean white noise, v_k. The AR model is given by [49]

$$L_{k+1} = \sum_{j=1}^{p} \phi_j L_{k-j} + v_k, \tag{7.4}$$

where the ϕ_js are the coefficients of the AR model identified using standard statistical techniques. Let $\widehat{L}_k$ denote the estimate of the power consumed, then

$$\widehat{L}_k = \sum_{j=1}^{p} \phi_p L_{k-j}, \tag{7.5}$$

is the optimal estimate of the AR model in (7.4). We assume that the past samples, L_{k-j}, are available by using sensing methodologies [49].

7.1.3 Combined cyber and physical model

The power supplied by the generator, $P_{g,k}$, and the power delivered to the load, $P_{L,k}$ are related by the interconnection network of the generators and loads. Let

$$\mathbf{P}_{g,k} = [P^1_{g,k} \ \ldots \ P^K_{g,k}]^T, \tag{7.6}$$

be the vector of power supplied by the K generators, and let

$$\mathbf{P}_{L,k} = [P^1_{L,k} \ \ldots \ P^M_{L,k}]^T, \tag{7.7}$$

be the vector of power delivered to the M loads. Define

$$\mathbf{\Omega}_{g,k} \triangleq [\omega^1_{g,k} \ \ldots \ \omega^K_{g,k}]^T, \tag{7.8}$$

$$\mathbf{\Omega}_{g,k} \triangleq [\omega^1_{L,k} \ \ldots \ \omega^M_{L,k}]^T. \tag{7.9}$$

[2]The values of J_L and D_L can be obtained using systematic model identification methods at each sub-station [49].

Then $\mathbf{P}_{g,k}$ and $\mathbf{P}_{L,k}$ are related by

$$\begin{bmatrix} \mathbf{P}_{g,k+1} \\ \mathbf{P}_{L,k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{g,k} \\ \mathbf{P}_{L,k} \end{bmatrix} + \Delta_T \mathcal{H} \begin{bmatrix} \mathbf{\Omega}_{g,k+1} \\ \mathbf{\Omega}_{L,k+1} \end{bmatrix}, \tag{7.10}$$

where $\mathcal{H}$ is a $J \times J$ interconnection matrix. Equation (7.1), (7.3) and (7.10) complete the cyber-physical description of the dynamics generated by a power system with K steam-turbine generators and M arbitrary loads and can be written concisely as an $n-$dimensional vector

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{b} - \frac{1}{J_L}\mathbf{G}_L\widehat{\mathbf{L}}_k + \mathbf{u}_k, \tag{7.11}$$

where

$$\mathbf{x}_k = [\mathbf{x}_{g,k}^{1T} \ \ldots \ \mathbf{x}_{g,k}^{KT} \ \mathbf{\Omega}_{L,k}^T \ \mathbf{P}_{g,k}^T \ \mathbf{P}_{L,k}^T]^T, \tag{7.12}$$

is the global state vector of the entire system and $\mathbf{F}, \mathbf{b}, \mathbf{G}_L$ are the appropriate quantities derived from (7.1),(7.3), (7.4), and (7.10). For more details, see [49]. We assume the following global observation model,

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k, \tag{7.13}$$

where: $\mathbf{y}_k$ is the observation vector; $\mathbf{H}$ is the observation matrix; and $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is a white noise vector. The noise sequences $\{\mathbf{u}\}_{k\geq 0}$ and $\{\mathbf{w}\}_{k\geq 0}$ and the initial condition, $\mathbf{x}_0$, are all statistically independent.

In the next section, we will apply the distributed estimation algorithm of Chapter 6 to the model given by (7.11) and (7.13).

7.2 Distributed estimation in electric power systems

In this section, we present a distributed estimation scheme for electric power systems based on its cyber-physical model (Section 7.1). We consider the global power system

model (7.11)-(7.13) as a union of several, possibly many, low dimensional sub-systems. Each sub-system may represent a generating sub-station, a large aggregated load, or a combination of both. A structurally robust and computationally efficient modeling of the union of such sub-systems is where each sub-system has the knowledge of its own dynamics exactly, but, treats the rest of the dynamical system as interactions among its neighboring sub-systems. Such a scheme is robust to structural changes such that if a sub-station or load is added/removed to/from the overall dynamical system, only its neighboring sub-systems are to be updated. Further, this implementation does not require, anywhere in the network, the knowledge of the complete description of the entire large-scale dynamical system.

Treating the global power system model as a union of sub-systems lends itself to a low order (observers) Information filter implementation at each sub-system where each sub-system interacts only with neighboring sub-systems to achieve performance close to the performance of the global Information filter. As shown in Chapter 6, the local interacting Information filters have a computational complexity $O(n_l^4)$ (where n_l is the dimension of the lth sub-system), whereas the global Information filter has a computational complexity of $O(n^3)$ (where n is the dimension of the entire system). Clearly, by appropriate sub-system selection, we can make $n_l \ll n$. Hence, the resulting estimation scheme is computationally efficient, can be implemented in real-time and remains practical, since it does not require any global knowledge or long distance communication, but, relies only on local communication among neighboring sub-systems.

After spatially decomposing the cyber-physical model,[3] we implement local Information filters on the sub-systems where each local filter preserves a Gauss-Markovian structure on the error processes in the Information filter. This is achieved by approximating the information matrices (inverse of the error covariance) of the global Information filter by $L-$banded positive-definite matrices. Hence, the performance of the local filters is equivalent to the performance of the centralized filter with such approximation.

[3]We may use the spatial decomposition based on cut-point sets (Section 6.3) or other spatial model distribution techniques, for instance, ϵ-decomposition, overlapping ϵ-decompositions, and eigen-value based methods [134, 155, 150].

Formulating local observers on sub-systems when these sub-systems are observable is studied in [134]. The structure of such systems are studied in [155]. In [134], the local observers are formulated with the assumption that the local observers are observable and conditions are derived on the interactions (among the sub-systems) under which the global system, as a whole, may lose observability. In our algorithm, we do not assume any observability conditions on the local sub-systems, but require the global system to be observable. Under such conditions, we propose cooperation among the observers formulated at the sub-systems such that, after cooperation, the resulting system remains observable. The scheme we present is computationally efficient, the interactions and cooperation are always local and is robust to structural changes in the dynamical system.

7.2.1 Local models

We spatially decompose the power system model as a union of N sub-systems, where each sub-system measures a subset of the state variables (that are local to the sub-system) of the power system model in (7.11). The local observation vector[4] for the lth sub-system is given by

$$\mathbf{y}_k^{(l)} = \mathbf{H}_l \mathbf{x}_k + \mathbf{w}_k^{(l)}, \tag{7.15}$$

where $\mathbf{H}_l$ is the local observation matrix with the white observation noise vector $\mathbf{w}_k^{(l)} \sim \mathscr{N}(0, \mathbf{R}_l)$, related to the error covariance of the observation noise process as

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 & & & \\ & \mathbf{R}_2 & & \\ & & \ddots & \\ & & & \mathbf{R}_N \end{bmatrix}. \tag{7.16}$$

[4]We stack the local observations to get the global observation vector,

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{y}_k^{(1)} \\ \vdots \\ \mathbf{y}_k^{(N)} \end{bmatrix}, \qquad \mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_N \end{bmatrix}, \qquad \mathbf{w}_k = \begin{bmatrix} \mathbf{w}_k^{(1)} \\ \vdots \\ \mathbf{w}^{(N)} \end{bmatrix}. \tag{7.14}$$

At sub-system l, the dynamics after decomposing (6.6) can be written as an n_l-dimensional local dynamical sub-system as follows.

$$\mathbf{x}_{k+1}^{(l)} = \mathbf{F}^{(l)}\mathbf{x}_k^{(l)} + \sum_{j\in\mathcal{K}(l)} \mathbf{F}^{(lj)}\mathbf{x}_k^{(j)} + \mathbf{G}^{(l)}\mathbf{u}_k^{(l)}, \tag{7.17}$$

$$\mathbf{y}_k^{(l)} = \mathbf{H}^{(l)}\mathbf{x}_k^{(l)} + \mathbf{w}_k^{(l)}, \tag{7.18}$$

where: $\mathcal{K}(l)$ contains the neighboring sub-systems; $\mathbf{F}^{(lj)}$ defines the interaction between the sub-system l and its jth neighbor; and $\mathbf{H}^{(l)}$ is the local observation matrix pertinent to the local state variables, $\mathbf{x}_k^{(l)}$. Here we note that the interactions from the neighboring sub-systems, $\sum_{j\in\mathcal{K}(l)} \mathbf{F}^{(lj)}\mathbf{x}_k^{(j)}$, can only be treated as deterministic inputs if these states are available at the neighboring sub-systems exactly. Since, these states are unavailable, we use their estimates, $\sum_{j\in\mathcal{K}(l)} \mathbf{F}^{(lj)}\widehat{\mathbf{x}}_{k|k}^{(j)}$, as the local interactions.

7.2.2 Remarks

We now give some important remarks.

(**R1**): We assume that the global state-space dynamical system (6.6) is $(\mathbf{F}, \mathbf{H})-$observable.

It is clear from (**R1**) that we do not require any observability for the local sub-systems and each local sub-systems may *not* be $(\mathbf{F}, \mathbf{H}_l)-$observable . In such cases, a local Information filter implemented at sub-system, l, results into unstable local error covariances, $\mathbf{S}_{k|k}^{(l)}$, if it is not observable, i.e., the following limit

$$\lim_{k\to\infty} \text{trace}\left(\mathbf{S}_{k|k}^{(l)}\right) \tag{7.19}$$

does not exist and, further, the sequence is unbounded.

(**R2**): The coupling matrices of the entire dynamical system, $\mathbf{F}^{(lj)}\ \forall\ l, j,\ l \neq j$, are sparse.

(**R3**): The neighborhood, $\mathcal{K}(\cdot)$, at each sub-system only contains neighboring sub-systems.[5]

[5]It is worth mentioning here that the sparsity (**R2**) and the locality (**R3**) ensure local communication in the resulting algorithm.

(**R4**): Let the local information matrix at sensor l be given by $\mathbf{Z}_{k|k}^{(l)}$. We assume that the local information matrices, $\{\mathbf{Z}_{k|k}^{(l)}\}_{l=1,\ldots,N}$, preserve the $L-$band of the global $L-$banded information matrix, $\mathbf{Z}_{L,k|k}$. Since these matrices are $L-$banded and if the following condition (developed in Section 6.3.1: Equation (6.29))

$$n_l \geq L, \qquad \forall\ l, \tag{7.20}$$

holds then it can be shown that the approximated matrices can be preserved by the sub-systems, i.e., (with a slight abuse of notation),

$$\bigcup_{l=1}^{N} \mathbf{Z}_{k|k}^{(l)} = \mathbf{Z}_{L,k|k}. \tag{7.21}$$

It is straightforward to show that (**R4**) requires the sub-systems to be overlapping.

Using the results of Chapter 6, it turns out that, by cooperation among the local sub-systems and under the assumptions (**R1**)–(**R4**) with a choice of L such that $L \geq L_{\min}$, the trace of the local error covariances, trace $\left(\mathbf{S}_{k|k}^{(l)}\right)$, can be bounded, i.e., the limit in (7.19) exists. The steady state error for the collection of local Information filters implemented on the sub-systems is, thus, bounded and the bounded difference from the optimal (minimum) steady state error,

$$\text{trace}\left(\bigcup_{l=1}^{N} \mathbf{S}_{k|k}^{(l)}\right) - \text{trace}\left(\mathbf{S}_{k|k}\right) < \overline{M} < \infty, \tag{7.22}$$

($\overline{M} \in \mathbb{R}_{\geq 0}$) can be characterized by the information loss incurred by approximating the global information matrices, $\mathbf{Z}_{k|k}$ and $\mathbf{Z}_{k|k-1}$, to be $L-$banded information matrices, $\mathbf{Z}_{L,k|k}$ and $\mathbf{Z}_{L,k|k-1}$, and is given by the divergence in (6.24).

Aggregating sub-systems: As mentioned before and elaborated in Section 6.3.1, the dimension, n_l, of the sub-systems should follow (7.20). Hence, if sub-systems result in lower dimensional dynamical systems, aggregation is required on the sub-systems, such that the dimensions of the resulting aggregated sub-systems follow (7.20).

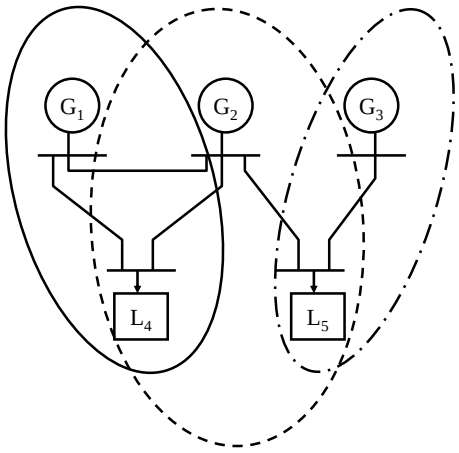


Figure 7.1: $N = 5-$bus system with $K = 3$ generators and $M = 2$ loads.

7.2.3 Illustration

We consider an $M = 5$ bus system with $K = 3$ generators and $L = 2$ loads, see Figure 7.1. The large ovals represent the $N = 3$ sub-systems. Each generator is considered to be a steam-turbine generator with the following parameters, inertia constant, $J = 1.26$, damping coefficient, $D = 2$, time constant of the turbine, $T_u = 0.2$, time constant of the generator, $T_g = 0.25$, coefficient of the valve position, $e_T = 0.15$, proportionality factor, $K_T = 0.95$, $r = 0.95$ and the sampling interval is $\Delta_T = 0.01$s. For the loads, we assume $J_L = 10$ and $D_L = 1$. The interconnection matrix, $\mathcal{H}$, represents the connectivity of the network, the non zero pattern is given by

$$\mathcal{H} = \begin{bmatrix} h_{11} & h_{12} & 0 & h_{14} & 0 \\ h_{21} & h_{22} & 0 & 0 & h_{25} \\ 0 & 0 & h_{33} & 0 & h_{35} \\ h_{41} & h_{42} & 0 & h_{44} & 0 \\ 0 & 0 & h_{53} & 0 & h_{55} \end{bmatrix}. \tag{7.23}$$

The power consumed at the loads is modeled as an AR process of order $p = 2$. The load at bus 4 is taken from the New York ISO from the month of August 2006 and the load at bus 5 is taken from the New England ISO for the same month. For

simplicity of illustration, we assume the loads to be at the same time-scale as the generators. The global observation model consists of the observations on the state variables

$$\omega_g^1,\ \omega_g^2,\ \omega_g^3,\ P_g^1,\ P_g^2,\ P_g^3,\ P_L^4,\ P_L^5,$$

i.e., the diagonal global observation matrix, $\mathbf{H}$, has non-zeros corresponding to the aforementioned states. The global model, thus, consists of the $K = 3$ steam-turbine generators with the above parameters, $L = 2$ loads (the power consumed for each of which is modeled using an AR(2) process) and the power flow equation with the $M \times M$ (5×5) interconnection matrix given above. The resulting model is an $n = 16-$dimensional dynamical system that is observable with the above observations; the observability Grammian, $\mathbf{\Theta_g}$, has the largest singular value of 12.08 and smallest singular value of 0.0108 and thus has rank $n = 16$.

We now formulate sub-systems on this network. Sub-system 1 has measurements corresponding to the states $\omega_g^1,\ P_g^1,\ P_L^4$, sub-system 2 has measurements corresponding to the states $\omega_g^2,\ P_g^2,\ P_L^4,\ P_L^5$ and sub-system 3 has measurements corresponding to the states $\omega_g^3,\ P_g^3,\ P_L^5$. It can be shown that each of the sub-systems is not observable with this set of measurements.

We now formulate sub-system dynamics. The $N = 3$ sub-systems has the following state vectors,

$$\mathbf{x}^{(1)} = [\mathbf{x}_g^{(1)T}\ P_g^1\ \omega_L^4\ P_L^4]^T, \tag{7.24}$$

$$\mathbf{x}^{(2)} = [\mathbf{x}_g^{(2)T}\ P_g^2\ \omega_L^4\ P_L^4\ \omega_L^5\ P_L^5]^T, \tag{7.25}$$

$$\mathbf{x}^{(3)} = [\mathbf{x}_g^{(3)T}\ P_g^3\ \omega_L^5\ P_L^5]^T. \tag{7.26}$$

The results are shown in Figures 7.2, 7.3, and 7.4. Figure 7.2 shows the trace of the local error covariance matrices at each sub-system without cooperation. Figure 7.3 shows the trace of the local error covariances at each sub-system when the DICI algorithm (Chapter 5) is used to assimilate the local error covariances together. Figure 7.4 compares the performance of the aggregated (all 3 sub-systems combined) Information filter with $L = 5-$banded approximation with the optimal Information

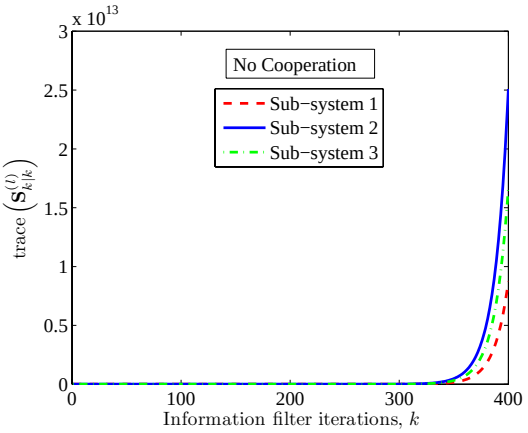


Figure 7.2: Trace of the local error covariance matrices at the sub-systems when no cooperation is employed with $L = 5-$banded approximation (on the information matrices).

filter (with no approximation).

Notice that the time-scale of the DICI algorithm, t, is different from the time-scale of the Information filter iterations, k. It is assumed that the communication is fast enough such that the DICI algorithm has reached a pre-defined performance criterion, between any two steps of the Information filter iterations. This is a safe assumption as the communication is always local and it can be assumed to be relatively faster than the evolution of the dynamical system.

7.2.4 Conclusions

In this section, we show that with the appropriate overlapping of the sub-systems and using an assimilation procedure on the local error covariances, the un-observable sub-systems can be made observable in the sense that the local error covariances remain stable. Local Information filters are implemented on the sub-systems that guarantee a Gauss-Markovian structure on the error processes. This is achieved by using an assimilation procedure (DICI algorithm, Chapter 5) among the local error covariances that preserves the Gauss-Markovian structure. The sum of the squared errors (trace of the error covariance) thus remains bounded and the aggregated performance over all of the sub-systems is equivalent to the Information filter with $L-$banded approximation

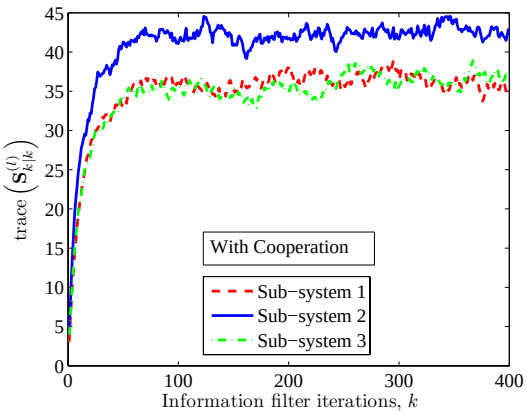


Figure 7.3: Trace of the local error covariance matrices at the sub-systems with cooperation using the DICI algorithm with $L = 5-$banded approximation (on the information matrices).

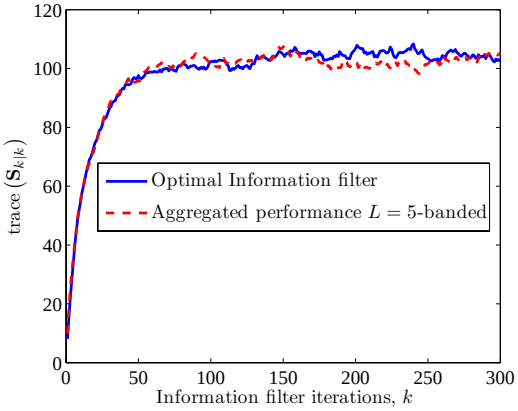


Figure 7.4: Aggregated performance of the sub-systems with $L = 5-$banded approximation (on the information matrices) and comparison with optimal Information filter.

on its information matrices.

7.3 Distributed phase-angle estimation

In this section, we present a distributed inference algorithm for phase-angle estimation in electric power systems, by deploying only a minimum number of phasor measurement units (PMUs). We assume that the nodes with unknown phase-angles have the knowledge of their relative phase-angles with their neighboring nodes and only a few nodes, which we call anchor nodes, have the absolute knowledge of their phase-angles, i.e., they are equipped with PMUs. By placing the PMUs strategically, it can be shown that the minimum number of anchor nodes required is 2. The proposed framework is robust to noisy measurements, noisy communication and link failures.

Phasor measurement units (PMUs) are highly accurate, but, expensive devices for measuring the phase-angles. In a power grid, where we are interested in measuring the phase-angles at all the nodes in the network, it is prohibitive to deploy a PMU at each node. A natural question to ask is what is the least number of PMUs to accurately estimate the phase-angles at every node and where should we deploy these PMUs.

We study the two questions posed above and further assume that each node in the network is equipped with an inexpensive measurement device that estimates the relative phase between the nodes. This can be measured, e.g., by current measurement devices that are relatively cheap. The difference in the current flowing between the nodes provides an estimate for the relative phase between the nodes.

We apply the distributed sensor localization algorithm (Chapter 4 specialized to $m = 1$-dimensions, since the phase-angles are one-dimensional quantities. To recapitulate, we showed in Chapter 4 that only a minimal number of $m + 1$ anchor nodes (with known locations, i.e., they are equipped with a GPS) are required for the sensor nodes to compute, iteratively, their unknown locations. We extend this to phase-angle estimation where a minimal number $m + 1$ of anchors (with known phase-angles, i.e., they are equipped with PMUs) are required for the rest of the nodes (that we call sensors) to compute, iteratively, their unknown phase-angles. We

assume that

(i) the phase-angles at the sensors lie in the convex hull of the phase-angles at the anchors;

(ii) the sensors form a connected network and each sensor can communicate to at least $m+1=2$ neighboring sensors such that the phase-angle of the sensor in question lies in the convex hull of the phase-angles at the $m+1=2$ neighboring nodes (the convex hull is determined under the relative phase-angle metric that we define later); and

(iii) each anchor can communicate to at least one sensor.

The assumption (i) provides us with the locations where the PMUs should be placed. Since we are only estimating the phase-angles, a scalar quantity, DILOC (Chapter 4) can be employed to localize the phase-angles, and we only require to place $m+1=2$ PMUs to estimate these phase-angles, iteratively, provided that (a) the PMUs are deployed so that the aforementioned assumptions (i)-(iii) are satisfied; and (b) all the nodes know the relative phase-angle between them and their (at least $m+1=2$) neighboring nodes. We are interested in implementing a distributed inference algorithm to iteratively estimate the phase-angles at the nodes in an electric power system. We summarize the requirements of the algorithm: (i) at each node, i, the state update depends at most on the states of the neighboring nodes; (ii) the state update is linear (convex); (iii) the coefficients of the linear state update are computed locally; (iv) only local communication within the neighborhood is allowed.

7.3.1 Notation and algorithm

Consider an electric power network with N nodes. Our algorithm entails two types of networks: (i) the electric network or the physical network; and (ii) the communication network or the information network. We term the interconnections in the physical network as the physical links and the interconnections in the information network as the information links. We denote the phase-angle at the ith node by δ_i. If node j is

connected to node i in the physical network, we represent their phase difference by

$$\delta_{ij} = \delta_i - \delta_j. \tag{7.27}$$

The phase difference between node i and node j can be computed from the following expression,

$$\delta_{ij} \propto \frac{y_{ij}}{z_{ij}}, \tag{7.28}$$

where y_{ij} and z_{ij} is the current and impedance of the bus between node i and node j, respectively. Let $\mathcal{K}(i)$ denote the physical neighbors of the ith node, i.e., the set of nodes connected to node i in the electric network.

A node, i, is said to be a *boundary node* if its phase-angle is such that

$$\delta_i > \delta_j, \qquad \forall j \in \mathcal{K}(i), \tag{7.29}$$

or,

$$\delta_i < \delta_j, \qquad \forall j \in \mathcal{K}(i). \tag{7.30}$$

where we let B to denote the set of boundary nodes.

A source node, $\overline{s}$, is the boundary node that has the largest phase-angle among all the nodes in $\mathcal{B}$ and a sink node, $\underline{s}$, is a boundary node that has the smallest phase-angle among all the boundary nodes. The set of the source node and the sink node is the set of anchors, denoted by κ, i.e., $\kappa = \{\overline{s}, \underline{s}\}$.

A *non-boundary* node is a node which is not a boundary node, where we let Ω to be the set of non-boundary nodes. Here, we assume that at each non-boundary node, $i \in \Omega$, there are at least two non-boundary and/or anchors nodes, $j, k \in \mathcal{K}(i) \cap (\Omega \cup \kappa)$, such that for node i, either

$$\delta_j \leq \delta_i \leq \delta_k, \tag{7.31}$$

or

$$\delta_j \geq \delta_i \geq \delta_k, \tag{7.32}$$

is true. The set of nodes, $\{j, k\}$, is said to be the triangulation set, Θ_i, of node i. If $\Theta_i \not\subset \Omega \cup \kappa$, then we include that ith node in the set of boundary nodes, B, also.

The set of all the nodes in the network is denoted by $\Theta = \Omega \cup B$. We make the following assumption.

(A1) Each node, i, knows its phase difference, δ_{ij}, with each of its neighbors, $j \in \mathcal{K}(i)$, in the physical network. This can be computed from (7.28).

(A2) There is a PMU deployed at the source and the sink node.

(A3) In the physical network, there is a path from the non-boundary nodes to each (boundary or non-boundary) node in the network.

(A4) Each non-boundary node has an information link to the nodes in its triangulation set. In the communication network, there is a path from the non-boundary nodes to each node in the network.

Algorithm: We now implement an iterative algorithm for phase-angle estimation at all the non-source and non-sink nodes in the network. We assume that each node maintains a state, its phase-angle estimate, which is updated as a linear (convex) combination of the states at the neighboring nodes. Let $c_m(t), m \in \Theta$, denote the state of an arbitrary node at time t. Let $u_1(t)$ and $u_2(t)$ denote the state of the source node and the sink node, respectively, at time t. Let $x_j(t), j \in \Theta$, denote the state of the jth non-boundary nodes at time t, and let $r_l(t), l \in B \setminus \kappa$, denote the state of the lth boundary node that is neither source nor sink, at time t. The iterative algorithm is given by

$$\begin{aligned} u_i(t+1) &= u_i(t), && i \in \kappa, \\ x_j(t+1) &= \textstyle\sum_{m \in \Theta_j} \upsilon_{jm} c_m(t), && j \in \Omega, \\ r_l(t+1) &= c_n(t+1) + \delta_{ln}, && n \in \mathcal{K}(l),\ l \in B \setminus \kappa, \end{aligned} \tag{7.33}$$

where

$$\upsilon_{jm} = \frac{|\delta_{jl}|}{\sum_{i \in \Theta_j} |\delta_{ji}|}, \qquad l = \Theta_j \setminus \{m\}, \tag{7.34}$$

and $\setminus$ denotes the set difference. It can be easily verified that

$$\upsilon_{im} \in [0,1], \qquad \text{and} \qquad \sum_{m \in \Theta_i} \upsilon_{im} = 1, \tag{7.35}$$

and that υ_{im} are the barycentric coordinates in 1-d space (see Section 4.2.2).

For the purpose of analysis, we now write the above algorithm in matrix form. Let $\mathbf{I}_2$ denote the 2×2 identity matrix and let $\mathbf{u}(t)$, $\mathbf{x}(t)$, and $\mathbf{r}(t)$ denote the vectors of the states at the nodes in κ, Ω, and $B \setminus \kappa$, respectively. We have

$$\begin{bmatrix} \mathbf{u}(t+1) \\ \mathbf{x}(t+1) \\ \mathbf{r}(t+1) \end{bmatrix} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{B} & \mathbf{P} & \mathbf{0} \\ \mathbf{B}_r & \mathbf{P}_r & \mathbf{A}_r \end{bmatrix} \begin{bmatrix} \mathbf{u}(t) \\ \mathbf{x}(t) \\ \mathbf{r}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{\Delta} \end{bmatrix}, \tag{7.36}$$

where $\mathbf{\Delta}$ represents the appropriate vector of $\delta_{(\cdot)(\cdot)}$ added to r_l in (7.33). The vector, $\mathbf{u}(t+1)$, denotes the state of the anchors (source/sink) that are not updated because they are equipped with a PMU, by assumption **(A2)**. **Convergence:** Notice that the top right 2×2 sub-matric in (7.33) is reminiscent of the DILOC matrix form (Chapter 4). Since $\mathbf{P}$ is the collection of barycentric coordinates (in $m = 1$ dimensions) of non-boundary nodes (sensors in Chapter 4), we have $\rho(\mathbf{P}) < 1$ and

$$\lim_{t \to \infty} \mathbf{x}(t+1) = (\mathbf{I} - \mathbf{P})^{-1} \mathbf{B} \mathbf{u}. \tag{7.37}$$

That $(\mathbf{I} - \mathbf{P})^{-1} \mathbf{B} \mathbf{u}(0)$ are the actual phases at the non0boundary nodes can be established from the fixed point of (7.33), along the lines of Lemma 20 or Lemma 22. The convergence of the non-anchor boundary nodes, i.e., nodes in $B \setminus \kappa$, is guaranteed, if the states of the non-boundary nodes converge, since each non-boundary node has a path to every node in the network (by **(A4)**). The convergence at these non-anchor boundary nodes is not governed by the geometric rate of the HDC. Instead, it depends on their distance (in the number of network hops) from the anchors and non-boundary nodes. If the density of such non-anchor boundary nodes is large, the convergence at these nodes may be slow.

7.3.2 Generalizations

We briefly consider some generalizations.

- PMU placement: We require to place the PMUs only at the maximum phase-angle and the minimum phase-angle nodes in the network. When the network

is dynamic, we may find the set of nodes such that the set always contains the maximum and minimum phase-angle nodes and place PMUs at all such nodes. This may increase the number of PMUs required. Nevertheless, if the max and min nodes do not change for the entire network operation, then we only require 2.

- Random environments: Each node may only be able to measure a noisy version of the phase difference (as assumed in **(A1)**). Mathematically, node i may measure the following quantity,

$$\widetilde{\delta}_{ij} = \delta_{ij} + \overline{\delta}_{ij}, \tag{7.38}$$

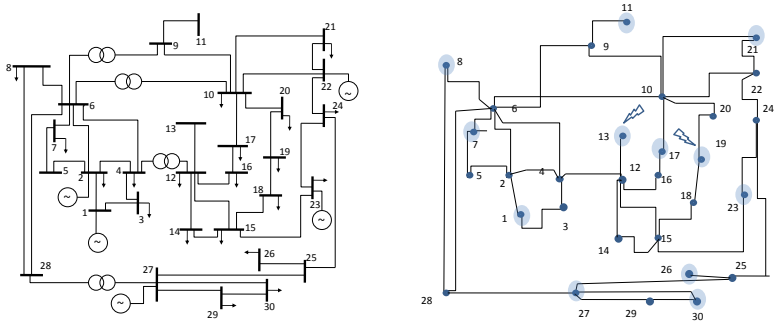
where $\overline{\delta}_{ij}$ is a random noise term with the following properties.

$$\mathbb{E}\left[\overline{\delta}_{ij}\right] = 0, \tag{7.39}$$
$$\mathbb{E}\left[\overline{\delta}_{ij}^2\right] < \infty. \tag{7.40}$$

In addition, the communication may suffer from communication noise and random link failures. In such cases, we modify the phase-angle estimation algorithm in (7.33) using a stochastic approximation. We provided discussion and details on such modifications in Section 3.5 and Section 4.7. Such modifications directly apply here.

7.3.3 Simulations

We apply the distributed phase-angle estimation algorithm to the IEEE 30-bus system shown in Fig. 7.5(a). We translate this system into a graphical representation in Fig. 7.5(b). The boundary nodes are encircled, whereas, the nodes with maximum and minimum phase-angles are shown with a lightning symbol. The rest are the non-boundary nodes. We provide the simulation results in Fig. 7.6(a)–Fig. 7.6(c).



(a) (b)

Figure 7.5: (a) IEEE 30-bus system and its (b) graphical representation.

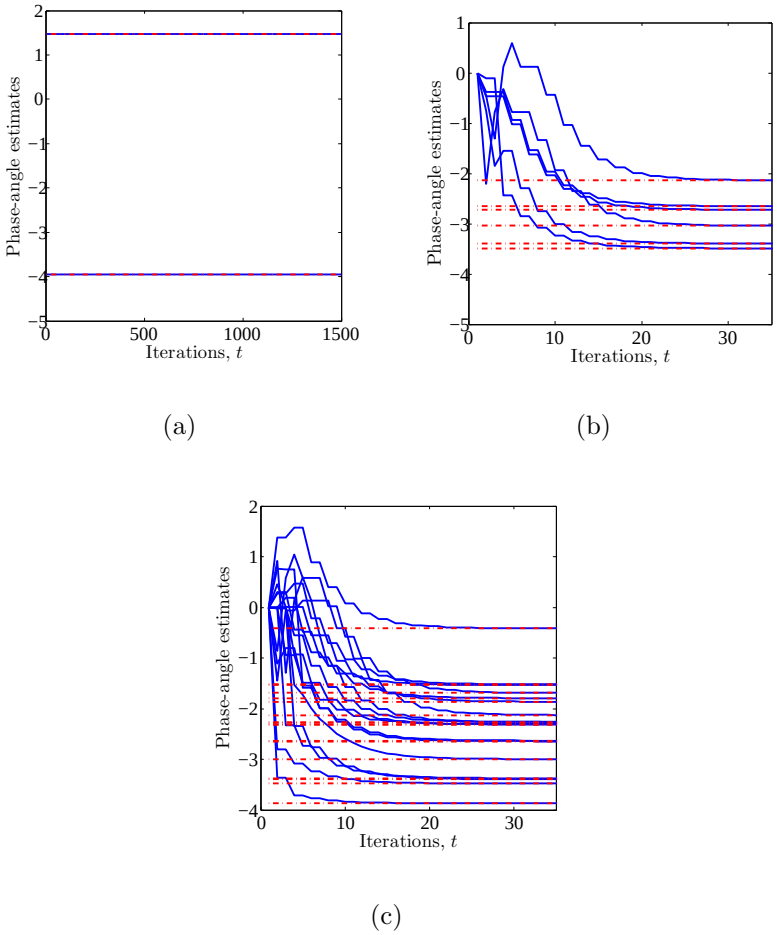


Figure 7.6: Distributed phase-angel algorithm: Estimates for (a) anchors, (b) non-anchor boundary nodes, and (c) non-boundary nodes.

Chapter 8

Epilogue

In this chapter, we revisit our contributions in this thesis (presented earlier in Section 1.1).

In this thesis, we developed the theory of High Dimensional Consensus (HDC) in large-scale networks. We further show the significance of HDC in relevant practical applications. We now provide a brief list of the results developed in this thesis.

- **High Dimensional Consensus (HDC):** We proposed a new class of distributed algorithms that we call High Dimensional Consensus (HDC). HDC includes several existing well-known algorithms as special cases, e.g., Jacobi algorithm [58], and linear average-consensus [17], and is not restricted to linear updates with convex or non-negative coefficients. In our formulation of HDC, we studied analysis and synthesis problems.

  - *Analysis: Convergence in the linear-case.* In Section 3.3, we established appropriate conditions for the convergence of HDC in the linear case, and derived the limiting state of the network, and the convergence rate of the HDC.

  - *Synthesis: Design in the linear case.* In Section 3.6, we designed the updating functions such that we achieve a desired limiting state, an arbitrary linear combination of the initial anchor states. Our solution results into updating matrices that are not necessarily stochastic or non-negative. We

have formulated this design problem as learning in large-scale networks, which we solve using multi-objective optimization. We cast the solutions to this MOP in the context of Pareto-optimality and have shown that the optimal solution of the learning problem is Pareto-optimal. In Section 3.8, we have studied the trade-offs between the performance of HDC versus its convergence speed.

  - *Random environments:* In Section 3.5, we studied the behavior and performance of HDC in random environments (communication noise, data packet drops, and imperfect knowledge of the underlying system). We have proposed a modification to HDC such that it is robust to the random phenomena.
  - *Classes of HDC:* We showed that HDC generalizes a large class of problems, including Jacobi algorithm [58], average-consensus [17], leader-follower (Chapter 3) algorithm, sensor localization (Chapter 4), banded matrix inversion (Chapter 5), and distributed estimation (Chapter 6).

- **Analysis and synthesis of non-linear average-consensus:** In Chapter 2, we studied the non-linear distributed average-consensus algorithms. In Section 2.3, we characterized the non-linear updating functions that guarantee average-consensus. We then carried out the synthesis problem in Section 2.4, where we have designed the non-linear average-consensus using sinusoidal updating functions.

- **Distributed sensor localization:** In Chapter 4, we provided a detailed study of distributed sensor localization that is a special case of HDC. In this context, we proposed DILOC in Section 4.2. DILOC requires a minimal number of known locations to localize an arbitrary number of sensors. DILOC uses the barycentric coordinates as coefficients of the linear combination that constitutes the state update at each sensor. This linear combination is convex so the resulting updating matrices are stochastic. Hence, the convergence of DILOC can be tied to the steady state distribution of an absorbing Markov chain. Although, we also prove convergence using fixed point arguments. Using a Poisson distribution on

the sensor deployment, we provided probabilistic bounds on the sensor density and the communication radius required for DILOC. In Section 4.5, we considered several enhancements to DILOC by relaxing some of the assumptions. In Section 4.6, we studied localization in networks of mobile agents, whereas, in Section 4.7, we studied distributed sensor localization in random environments.

- **Distributed Kalman filter:** In Chapter 6, we provided a distributed estimation algorithm based on spatial decomposition of large-scale dynamical systems into low-dimensional sub-systems. At the heart of our estimation algorithm lies a distributed banded matrix inversion algorithm that we term Distributed Iterated Collapse Inversion (DICI) algorithm (Chapter 5). DICI is a special case of HDC (appended with a non-linear collapse operator) that assimilates the local error covariances among the sub-systems in a computationally efficient and completely decentralized fashion.

- **Applications to electric power grids:** In Chapter 7, we showed the significance of HDC in the context of distributed estimation and inference in large-scale power systems. We specifically addressed the following problems.

  - *Estimation and modeling:* In Section 7.2, we explored the practical significance of the distributed Kalman filter of Chapter 6 in the context of a structure-preserving model of the electric power system that we term as cyber-physical model. We provided the cyber-physical model in Section 7.1. We showed that cooperation among the (potentially unobservable) sub-systems, derived from the cyber-physical model, leads to the observability of the overall system.

  - *Phase-angle estimation:* In Section 7.3, we provided a distributed inference algorithm for phase-angle estimation that is based on the HDC algorithm and borrows some concepts from our distributed localization algorithm. In this context, we studied the minimal number of Phasor Measurement Units (PMUs) and their optimal placement.

In addition, we conducted a detailed literature review for comparisons and contrasts with our work. This review is provided in each chapter. We further provided an extensive set of experimental results in each of the chapters.

Appendix A

High dimensional consensus

A.1 Important results

Lemma 25: If a matrix $\mathbf{P}$ is such that

$$\rho(\mathbf{P}) < 1,$$

then

$$\lim_{t\to\infty} \mathbf{P}^{t+1} = \mathbf{0}, \tag{A.1}$$

$$\lim_{t\to\infty} \sum_{k=0}^{t} \mathbf{P}^k = (\mathbf{I}-\mathbf{P})^{-1}. \tag{A.2}$$

Proof: The proof is straightforward. ■

Lemma 26: Let $r_{\mathbf{Q}}$ be the rank of the $M \times M$ matrix $(\mathbf{I}-\mathbf{P})^{-1}$, and $r_{\mathbf{B}}$ the rank of the $M \times n$ matrix $\mathbf{B}$, then

$$\text{rank}(\mathbf{I}-\mathbf{P})^{-1}\mathbf{B} \leq \min(r_{\mathbf{Q}}, r_{\mathbf{B}}), \tag{A.3}$$

$$\text{rank}(\mathbf{I}-\mathbf{P})^{-1}\mathbf{B} \geq r_{\mathbf{Q}} + r_{\mathbf{B}} - M. \tag{A.4}$$

Proof: The proof is available on pages $95-96$ in [156]. ■

A.2 Necessary condition

Below, we provide a necessary condition required for the existence of an exact solution of the *Learning Problem*.

Lemma 27: Let $\rho(\mathbf{P}) < 1$, $K < M$, and let $r_{\mathbf{W}}$ denote the rank of a matrix $\mathbf{W}$. A necessary condition for $(\mathbf{I} - \mathbf{P})^{-1}\mathbf{B} = \mathbf{W}$ to hold is

$$r_{\mathbf{B}} = r_{\mathbf{W}}. \tag{A.5}$$

Proof: Note that the matrix $\mathbf{I} - \mathbf{P}$ is invertible since $\rho(\mathbf{P}) < 1$. Let $\mathbf{Q} = (\mathbf{I} - \mathbf{P})^{-1}$, then $r_{\mathbf{Q}} = M$. From Lemma 26 in Appendix A.1 and since by hypothesis $K < M$,

$$\begin{aligned} \text{rank}(\mathbf{QB}) &\leq r_{\mathbf{B}}, & \text{(A.6)} \\ \text{rank}(\mathbf{QB}) &\geq M + r_{\mathbf{B}} - M = r_{\mathbf{B}}. & \text{(A.7)} \end{aligned}$$

The condition (A.5) now follows, since from (3.41), we also have

$$\text{rank}(\mathbf{QB}) = r_{\mathbf{W}}. \tag{A.8}$$

■

Appendix B

Localization in sensor networks

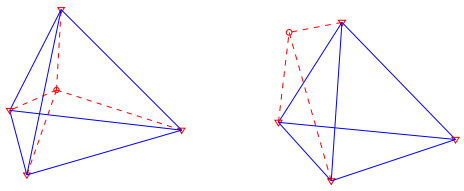
B.1 Convex hull inclusion test

We now give an algorithm that tests if a given sensor, $l \in \mathbb{R}^m$, lies in the convex hull of $m+1$ nodes in a set, κ, using only the mutual distance information among these $m+2$ nodes ($\kappa \cup \{l\}$). Let κ denote the set of $m+1$ nodes and let $\mathcal{C}(\kappa)$ denote the convex hull formed by the nodes in κ. Clearly, if $l \in \mathcal{C}(\kappa)$, then the convex hull formed by the nodes in κ is the same as the convex hull formed by the nodes in $\kappa \cup \{l\}$, i.e.,

$$\mathcal{C}(\kappa) = \mathcal{C}(\kappa \cup \{l\}), \qquad \text{if } l \in \mathcal{C}(\kappa). \tag{B.1}$$

With the above equation, we can see that, if $l \in \mathcal{C}(\kappa)$, then the generalized volumes of the two convex sets, $\mathcal{C}(\kappa)$ and $\mathcal{C}(\kappa \cup \{l\})$, should be equal. Let A_κ denote the generalized volume of $\mathcal{C}(\kappa)$ and let $A_{\kappa\cup\{l\}}$ denote the generalized volume of $\mathcal{C}(\kappa \cup \{l\})$, we have

$$\begin{aligned} A_\kappa &= A_{\kappa\cup\{l\}}, \\ &= \sum_{k\in\kappa} A_{\kappa\cup\{l\}\setminus\{k\}}, \qquad \text{if } l \in \mathcal{C}(\kappa). \end{aligned} \tag{B.2}$$



(a) (b)

Figure B.1: Convex Hull Inclusion Test ($m = 3$): The sensor l is shown by a '$\circ$', whereas, the anchors in κ are shown by '∇'. (a) $l \in \mathcal{C}(\kappa) \Rightarrow A_\kappa = A_{\kappa\cup\{l\}}$, (b) $l \notin \mathcal{C}(\kappa) \Rightarrow A_\kappa < A_{\kappa\cup\{l\}}$.

Hence, the test becomes

$$l \in \mathcal{C}(\kappa), \qquad \text{if } \sum_{k\in\kappa} A_{\kappa\cup\{l\}\setminus\{k\}} = A_\kappa, \tag{B.3}$$

$$l \notin \mathcal{C}(\kappa), \qquad \text{if } \sum_{k\in\kappa} A_{\kappa\cup\{l\}\setminus\{k\}} > A_\kappa. \tag{B.4}$$

This is also shown in Figure B.1. The above inclusion test is based entirely on the generalized volumes, which can be calculated using only the distance information in the Cayley-Menger determinants.

B.2 Cayley-Menger determinant

The Cayley-Menger determinant [86] is the determinant of an $m+2 \times m+2$ (symmetric) matrix that relates to the generalized volume, A_{Θ_l}, of the convex hull, $\mathcal{C}(\Theta_l)$, of the $m+1$ points in $\mathbb{R}^m$ through an integer sequence, s_{m+1}. Let $\mathbf{1}_{m+1}$ denote a column vector of $m+1$ 1s, the Cayley-Menger determinant is given by

$$A^2_{\Theta_l} = \frac{1}{s_{m+1}} \begin{vmatrix} 0 & \mathbf{1}^T_{m+1} \\ \mathbf{1}_{m+1} & \mathbf{Y} \end{vmatrix}, \tag{B.5}$$

where $\mathbf{Y} = \{d_{lj}^2\}, l, j \in \Theta_l$, is the matrix of squared distances, d_{lj}, among the $m+1$ points in Θ_l and

$$s_m = \frac{2^m (m!)^2}{(-1)^{m+1}}, \qquad m = \{0, 1, 2, \ldots\}. \tag{B.6}$$

Appendix C

Distributed estimation

C.1 L-banded inversion theorem

Let $\mathbf{Z} = \mathbf{S}^{-1}$ be L-banded. We apply the algorithm, given in [120], to obtain $\mathbf{Z}$ from the sub-matrices in the L-band of $\mathbf{S}$. We use the following notation, in (C.1), to partition matrix addition and subtraction in terms of its constituent sub-matrices. Also $\mathbf{S}_j^i$ represents the principal submatrix of $\mathbf{S}$ spanning rows i through j, and columns i through j.

$$\begin{bmatrix} a_1 & a_2 & 0 \\ a_3 & x+y+z & a_4 \\ 0 & a_5 & a_6 \end{bmatrix} = \begin{bmatrix} \boxed{\begin{matrix} a_1 & a_2 \\ a_3 & x \end{matrix}} + \boxed{y} + \boxed{\begin{matrix} z & a_4 \\ a_5 & a_5 \end{matrix}} \end{bmatrix} \tag{C.1}$$

The inverse of $\mathbf{S}$, when $\mathbf{Z} = \mathbf{S}^{-1}$ is L-banded, is given by (C.2), taken from [120], in terms of the L-banded sub-matrices of $\mathbf{S}$. Note that, to compute a principal submatrix in $\mathbf{Z}$, we do not need the entire $\mathbf{S}$, or even all the L-banded sub-matrices in $\mathbf{S}$. Instead, we only require neighboring sub-matrices in the L-band of $\mathbf{S}$. For proofs and further details, the interested reader can refer to [120].

$$\mathbf{Z} = \begin{bmatrix} \left[\mathbf{S}_{L+1}^{1}\right]^{-1} - \left[\mathbf{S}_{L+1}^{2}\right]^{-1} + \left[\mathbf{S}_{L+2}^{2}\right]^{-1} - & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & + \left[\mathbf{S}_{N-1}^{N-L-1}\right]^{-1} - \left[\mathbf{S}_{N-1}^{N-L}\right]^{-1} + \left[\mathbf{S}_{N}^{N-L}\right]^{-1} \end{bmatrix} \tag{C.2}$$

Bibliography

[1] E. Cuthill J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *Proceedings of the 24th National Conference*, New York, 1969, pp. 157–172.

[2] R. R. Tenney and Jr. N. R. Sandell, "Detection with distributed sensors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 17, no. 4, pp. 501–510, 1981.

[3] J. N. Tsitsiklis, *Problems in Decentralized Decision Making and Computation*, Ph.D., Massachusetts Institute of Technology, Cambridge, MA, 1984.

[4] J. N. Tsitsiklis, "Decentralized detection by a large number of sensors," *Mathematics of Control, Signals, and Systems*, vol. 1, no. 2, pp. 167–182, 1988.

[5] Pramod K. Varshney, *Distributed Detection and Data Fusion*, Springer-Verlag, Secaucus, NJ, 1996.

[6] J.-F. Chamberland and V. V. Veeravalli, "Decentralized detection in sensor networks," vol. 51, pp. 407–416, Feb. 2003.

[7] S. Aldosari and J. M. F. Moura, "Detection in sensor networks: The saddlepoint approximation," *IEEE Transactions on Signal Processing*, vol. 55, no. 1, pp. 327–340, January 2007.

[8] J. N. Tsitsiklis and M. Athans, "On the complexity of decentralized decision making and detection problems," *IEEE Transactions on Automatic Control*, vol. AC-30, no. 5, pp. 440–446, May 1985.

[9] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," vol. AC-31, no. 9, pp. 803–812, September 1986.

[10] H. J. Kushner and G. Yin, "Asymptotic properties of distributed and communicating stochastic approximation algorithms," *Siam J. Control and Optimization*, vol. 25, no. 5, pp. 1266–1290, Sept. 1987.

[11] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the Association for Computing Machinery*, vol. 32, no. 2, pp. 374–382, April 1985.

[12] N. A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1997.

[13] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," vol. AC-48, no. 6, pp. 988–1001, June 2003.

[14] C. Reynolds, "Flocks, birds, and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, pp. 25–34, 1987.

[15] T. Vicsek, A. Czirok, E. Ben Jacob, I. Cohen, and O. Schochet, "Novel type of phase transitions in a system of self-driven particles," *Physical Review Letters*, vol. 75, pp. 1226–1229, 1995.

[16] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automat. Contr.*, vol. 49, no. 9, pp. 1520–1533, Sept. 2004.

[17] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Controls Letters*, vol. 53, no. 1, pp. 65–78, Apr. 2004.

[18] S. Kar and José M. F. Moura, "Topology for global average consensus," in *40th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Oct. 2006.

[19] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference*, Portland, OR, June 2005, pp. 1859–1864.

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

[21] Y. Hatano, A. K. Das, and M. Mesbahi, "Agreement in presence of noise: pseudogradients on random geometric networks," in *44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05*, Seville, Spain, December 2005.

[22] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, pp. 2508 – 2530, June 2006.

[23] M. E. Yildiz and A. Scaglione, "Differential nested lattice encoding for consensus problems," in *ACM/IEEE Information Processing in Sensor Networks*, Cambridge, MA, April 2007.

[24] A. T. Salehi and A. Jadbabaie, "On consensus in random networks," in *The Allerton Conference on Communication, Control, and Computing*, Allerton House, IL, September 2006.

[25] M. Porfiri and D. J. Stilwell, "Stochastic consensus over weighted directed networks," in *Proceedings of the 2007 American Control Conference*, New York City, USA, July 11-13 2007.

[26] S. Kar and José M. F. Moura, "Distributed consensus algorithms in sensor networks: Link failures and channel noise," *IEEE Transactions on Signal Processing*, 2008, Accepted for publication, 30 pages.

[27] A. Kashyap, T. Basar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, pp. 1192–1203, July 2007.

[28] T. C. Aysal, M. J. Coates, and M. G. Rabbat, "Distributed average consensus with dithered quantization," *To appear in the IEEE Transactions of Signal Processing*, 2008.

[29] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Quantized data," Submitted for publication, 30 pages, see http://arxiv.org/abs/0712.1609, November 2007.

[30] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *Technical Report 2778, LIDS-MIT*, Nov. 2007.

[31] P. Frasca, R. Carli, F. Fagnani, and S. Zampieri, "Average consensus on networks with quantized communication," *Submitted to the Int. J. Robust and Nonlinear Control*, 2008.

[32] M. Huang and J. H. Manton, "Stochastic approximation for consensus seeking: mean square and almost sure convergence," in *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, LA, USA, Dec. 12-14 2007.

[33] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[34] H. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 863–868, 2007.

[35] Usman A. Khan and José M. F. Moura, "Distributing the Kalman filters for large-scale systems," *IEEE Transactions on Signal Processing*, vol. 56(1), no. 10, pp. 4919–4935, Oct. 2008.

[36] I. D. Schizas, A. Ribeiro, , and G. B. Giannakis, "Consensus-based distributed parameter estimation in ad hoc wireless sensor networks with noisy links," in

Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing, Honolulu, HI, 2007, pp. 849–852.

[37] R. Olfati-Saber, "Distributed Kalman filters with embedded consensus filters," in *44th IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005, pp. 8179 – 8184.

[38] M. Alanyali and V. Saligrama, "Distributed tracking in multi-hop networks with communication delays and packet losses," in *13th IEEE Workshop on Statistical Sig. Proc.*, Bordeaux, France, Jul. 2005, pp. 1190–1195.

[39] S. Kar, J. M. F. Moura, and K. Ramanan, "Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication," Submitted for publication, see also http://arxiv.org/abs/0809.0009, Aug. 2008.

[40] A. Rahmani and M. Mesbahi, "Pulling the strings on agreement: Anchoring, controllability, and graph automorphism," in *American Control Conference*, New York City, NY, July 11-13 2007, pp. 2738–2743.

[41] Usman A. Khan, Soummya Kar, and José M. F. Moura, "Distributed sensor localization in random environments using minimalnumber of anchor nodes," *IEEE Transactions on Signal Processing*, vol. 57, no. 5, pp. 2000–2016, May 2009.

[42] M. D. Ilić, L. Xie, U. A. Khan, and J. M. F. Moura, "Modeling, sensing, and control of future cyber-physical energy systems," *IEEE Transactions on Systems, Man and Cybernetics: Special Issue on Engineering Cyber-Physical Ecosystems*, Jul. 2008, accepted for publication.

[43] Usman A. Khan, Soummya Kar, and José M. F. Moura, "Higher dimensional consensus: Learning in large-scale networks," *IEEE Transactions on Signal Processing*, Apr. 2009, submitted.

[44] Usman A. Khan, Soummya Kar, and José M. F. Moura, "Sensor localization with noisy distance measurements," *IEEE Transactions on Signal Processing*, May 2009, submitted.

[45] Usman A. Khan, Soummya Kar, and José M. F. Mour, "Distributed algorithms in sensor networks," in *Handbook on sensor and array processing*, Simon Haykin and K. J. Ray Liu, Eds. Wily-Interscience, New York, NY, 2009, to appear.

[46] Usman A. Khan and José M. F. Moura, "Distributed Kalman filters in sensor networks: Bipartite fusion graphs," in *15th IEEE Workshop on Statistical Signal Processing*, Madison, WI, Aug. 2007, pp. 700–704.

[47] Usman A. Khan and José M. F. Moura, "Model distribution for distributed Kalman filters: A graph theoretic approach," in *41st Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2007, pp. 611–615.

[48] Usman A. Khan and José M. F. Moura, "Distributed Iterate-Collapse inversion (DICI) algorithm for $L-$banded matrices," in *33rd International Conference on Acoustics, Speech, and Signal Processing*, Las Vegas, NV, Mar. 2008, pp. 2529–2532.

[49] Marija D. Ilić, Le Xie, Usman A. Khan, and José M. F. Moura, "Modelling future cyber-physical energy systems," in *IEEE Power and Energy Society, General Meeting*, Pittsburgh, PA, Jul. 2008, pp. 1–9.

[50] Usman A. Khan, Soummya Kar, Bruno Sinopoli, and José M. F. Moura, "Distributed sensor localization in Euclidean spaces: Dynamic environments," in *46th Allerton Conference On Communication, Control, and Computing*, Monticello, IL, Sep. 2008, pp. 361–366.

[51] Usman A. Khan, Soummya Kar, and José M. F. Moura, "A linear distributed algorithm for sensor localization," in *42nd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Oct. 2008, pp. 1160–1164.

[52] Usman A. Khan, Marija D. Ilić, and José M. F. Moura, "Cooperation for aggregating complex electric power networks to ensure system observability," in *1st International Conference on Infrastructure Systems*, Rotterdam, Netherlands, Nov. 2008.

[53] Usman A. Khan, Soummya Kar, and José M. F. Moura, "Higher dimensional consensus algorithms in sensor networks," in *34th IEEE International Conference on Acoustics, Speech, and Signal Processing*, Taipei, Taiwan, Apr. 2009, pp. 2857–2860.

[54] Usman A. Khan, Soummya Kar, and José M. F. Moura, "Distributed localization in networks of mobile agents," in *47th Allerton Conference On Communication, Control, and Computing*, Monticello, IL, Sep. 2009, accepted for publication.

[55] Usman A. Khan, Soummya Kar, and José M. F. Moura, "Distributed average consensus: Beyond the realm of linearity," in *43rd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2009, accepted for publication.

[56] Usman A. Khan, Soummya Kar, and José M. F. Moura, "Asymptotic noise analysis of high-dimensional consensus," in *43rd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2009, accepted for publication.

[57] Usman A. Khan, Soummya Kar, and José M. F. Moura, "Distributed sensor localization using barycentric coordinates," in *3rd International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, Aruba, Dutch Antilles, Dec. 2009, submitted.

[58] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computations*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[59] F. R. K. Chung, *Spectral Graph Theory*, Providence, RI : American Mathematical Society, 1997.

[60] Y. Kuramoto, "Cooperative dynamics of oscillator community," *Progress of Theoretical Physics Suppl.*, vol. 79, pp. 223–240, 1984.

[61] C. W. Wu and L. O. Chua, "Synchronization in an array of linearly coupled dynamical systems," *IEEE Transactions on Circuits Systems I*, vol. 42, pp. 430–447, Aug. 1995.

[62] A. Jadbabaie, N. Motee, and M. Barahona, "On the stability of the Kuramoto model of coupled non-linear oscillators," in *American Control Conference*, Jul. 2004, vol. 5, pp. 296–4301.

[63] N Chopra and M. W. Spong, "On synchronization of kuramoto oscillators," in *44th IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005, pp. 3916–3922.

[64] A. Mauroy and R. Sepulchre, "Clustering behaviors in networks of integrate-and-fire oscillators," *Chaos: Special issue on Synchronization in Complex Networks*, vol. 18, no. 3, pp. 037122–037122–8, 2008.

[65] N. Varanese, O. Simeone, U. Spagnolini, and Y. Bar-Ness, "Distributed frequency-locked loops for wireless networks," in *10th IEEE International Symposium on Spread Spectrum Techniques and Applications*, Bologna, Italy, Aug. 2008, pp. 400–404.

[66] A. Bergen and D. Hill, "A structure preserving model for power system stability analysis," *IEEE Trans. on Power Apparatus and Systems*, vol. PAS-100, no. 1, pp. 25–35, Jan. 1981.

[67] F. R. Gantmacher, *Matrix Theory (Volume I)*, Chelsea Publishing Co., USA, 1959.

[68] P. Denantes, F. Benezit, P. Thiran, and M. Vetterli, "Which distributed averaging algorithm should I choose for my sensor network," in *INFOCOM*, Phoenix, AZ, Mar. 2008, pp. 986–994.

[69] V. Chankong and Y. Y. Haimes, *Multiobective decision making: Theory and methodology*, North-Holland series in system sciences and engineering, 1983.

[70] V. V. Kolpakov, "Matrix seminorms and related inequalities," *Journal of Mathematical Sciences*, vol. 23, no. 1, pp. 2094–2106, Sep. 1983, DOI: 10.1007/BF01093289.

[71] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, American Mathematical Society, 1997.

[72] A. Daghighi, "The Dirichlet problem for certain discrete structures," Tech. Rep. 4, Department of Mathematics, Uppsala University, Apr. 2005.

[73] F. Chung and S. T. Yau, "Discrete Green's functions," *Journal of Combinatorial Theory, Series A*, pp. 191–214, July 2000.

[74] L. S. Zurlo, P. E. Mercado, and C. E. de la Vega, "Parallelization of the linear load flow equations," *Power Tech Proceedings, 2001 IEEE Porto*, vol. 3, 2001.

[75] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," *46th IEEE Conference on Decision and Control*, pp. 2289–2294, Dec. 2007.

[76] G. Golub and C. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1996.

[77] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 433–455, Jun. 2004.

[78] H. G. Tanner, "On the controllability of nearest neighbor interconnections," in *43rd IEEE Conference on Decision and Control*, New York City, NY, Dec. 14-17 2004, pp. 2467–2472.

[79] R. Simmons, D. Apfelbaum, D. Fox, R. P. Goldman, K. Z. Haigh, D. J. Musliner, M. Pelican, and S. Thrun, "Coordinated deployment of multiple, heterogeneous robots," in *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, 2000, vol. 3, pp. 2254–2260 volume.3.

[80] E. Jones, B. Browning, M. B. Dias, B. Argall, M. M. Veloso, and A. Stentz, "Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks," in *International Conference on Robotics and Automation*, May 2006, pp. 570 – 575.

[81] J. Wang and M. Lewis, "Assessing cooperation in human control of heterogeneous robots," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, New York, NY, USA, 2008, pp. 9–16, ACM.

[82] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, NY, USA, 2004.

[83] R. T. Rockafellar, *Convex analysis*, Princeton University Press, Princeton, NJ, 1970. Reprint: 1997.

[84] G. Springer, *Introduction to Riemann Surfaces*, Addison-Wesley, Reading, MA, 1957.

[85] J. Hocking and G. Young, *Topology*, Addison-Wesley, MA, 1961.

[86] M. J. Sippl and H. A. Scheraga, "Cayley–Menger coordinates," *Proceedings of the National Academy of Sciences of U.S.A.*, vol. 83, no. 8, pp. 2283–2287, Apr. 1986.

[87] R. L. Moses, D. Krishnamurthy, and R. Patterson, "A self-localization method for wireless sensor networks," *EURASIP Journal on Applied Signal Processing*, , no. 4, pp. 348–358, Mar. 2003.

[88] N. Patwari, A. O. Hero III, M. Perkins, N. Correal, and R. J. ODea, "Relative location estimation in wireless sensor networks," *IEEE Trans. on Signal Processing*, vol. 51, no. 8, pp. 2137–2148, Aug. 2003.

[89] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from mere connectivity," in *4th ACM international symposium on mobile ad-hoc networking and computing*, Annapolis, MD, Jun. 2003, pp. 201–212.

[90] Y. Shang and W. Ruml, "Improved MDS-based localization," in *IEEE Infocom*, Hong Kong, Mar. 2004, pp. 2640–2651.

[91] F. Thomas and L. Ros, "Revisiting trilateration for robot localization," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 93–101, Feb. 2005.

[92] M. Cao, B. D. O. Anderson, and A. S. Morse, "Localization with imprecise distance information in sensor networks," Sevilla, Spain, Dec. 2005, pp. 2829–2834.

[93] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by local linear embedding," *Science*, vol. 290, pp. 2323–2326, Dec. 2000.

[94] N. Patwari and A. O. Hero III, "Manifold learning algorithms for localization in wireless sensor networks," in *IEEE ICASSP*, Montreal, Canada, Mar. 2004, pp. 857–860.

[95] D. Niculescu and B. Nath, "Ad-hoc positioning system," in *IEEE Globecom*, Apr. 2001, pp. 2926–2931.

[96] A. Savvides, C. C. Han, and M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *IEEE Mobicom*, Rome, Italy, Jul. 2001, pp. 166–179.

[97] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Intl. Workshop on Sensor Networks and Applications*, Atlanta, GA, Sep. 2002, pp. 112–121.

[98] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on an ad-hoc sensor network," in *2nd Intl. Workshop on Information Processing in Sensor Networks*, Palo Alto, CA, Apr. 2003, pp. 333–348.

[99] J. J. Caffery, *Wireless Location in CDMA Cellular Radio Systems*, Kluwer Academic Publishers, Norwell, MA, 1999.

[100] J. A. Costa, N. Patwari, and III A. O. Hero, "Distributed weighted-multidimensional scaling for node localization in sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 39–64, 2006.

[101] J. Albowicz, A. Chen, and L. Zhang, "Recursive position estimation in sensor networks," in *IEEE Int. Conf. on Network Protocols*, Riverside, CA, Nov. 2001, pp. 35–41.

[102] C. Savarese, J. M. Rabaey, and J. Beutel, "Locationing in distributed ad-hoc wireless sensor networks," in *IEEE ICASSP*, Salt Lake City, UA, May 2001, pp. 2037–2040.

[103] S. Čapkun, M. Hamdi, , and J. P. Hubaux, "GPS-free positioning in mobile ad-hoc networks," in *34th IEEE Hawaii Int. Conf. on System Sciences*, Wailea Maui, HI, Jan. 2001.

[104] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-calibration in sensor networks," in *IEEE ICASSP*, Montreal, Canada, May 2004.

[105] L. Hu and D. Evans, "Localization for mobile sensor networks," in *IEEE Mobicom*, Philadelphia, PA, Sep. 2004, pp. 45–57.

[106] M. Coates, "Distributed particle filters for sensor networks," in *IEEE Information Processing in Sensor Networks*, Berkeley, CA, Apr. 2004, pp. 99–107.

[107] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, Mar. 2002.

[108] A. Arora, S. Bapat P. Dutta, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. A. M. Nesterenko, A. Vora, and M. Miyashita, "Line in the sand: A wireless sensor network for target detection, classification, and tracking," in *Report OSU-CISRC-12/03-TR71, Ohio State University*, 2003.

[109] R. J. Orr and G. D. Abowd, "The smart floor: a mechanism for natural user identification and tracking," in *CHI 00 extended abstracts on Human factors in computing systems*, New York, NY, 2000, pp. 275–276.

[110] A. Chakraborty N. B. Priyantha and H. Balakrishnan, "The cricket location-support system," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, New York, NY, 2000, pp. 32–43.

[111] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *INFOCOM '00*, 2000, p. 775784.

[112] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurementsh," in *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems*, Baltimore, MD, Nov. 3-5 2004, pp. 50–61.

[113] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha, "Tracking moving devices with the cricket location system," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, New York, NY, 2004, p. 190202.

[114] K. Lorincz and M. Welsh, "Motetrack: A robust, decentralized approach to rf-based location tracking," in *Proceedings of the International Workshop on Location and Context- Awareness*, May 2005.

[115] N. Patwari, *Location estimation in sensor networks*, Ph.D., University of Michigan–Ann arbor, 2005.

[116] P. Hall, *Introduction to the theory of coverage processes*, John Wiley and Sons Inc., Chichester, UK, 1988.

[117] Y. Sung, L. Tong, and A. Swami, "Asymptotic locally optimal detector for large-scale sensor networks under the poisson regime," *IEEE Transactions on Signal Processing*, vol. 53, no. 6, pp. 2005–2017, Jun. 2005.

[118] T. S. Rappaport, *Wireless communications: Principles and practice*, Prentice Hall Inc., New Jersey, 1996.

[119] N. Balram and José M. F. Moura, "Noncausal Gauss Markov random fields: Parameter structure and estimation," *IEEE Trans. on Information Theory*, vol. 39, no. 4, pp. 1333–1355, Jul. 1993.

[120] A. Kavcic and José M. F. Moura, "Matrices with banded inverses: Inversion algorithms and factorization of Gauss-Markov processes," *IEEE Trans. on Information Theory*, vol. 46, no. 4, pp. 1495–1509, Jul. 2000.

[121] A. Asif and José M. F. Moura, "Inversion of block matrices with L-block banded inverse," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 630–642, Feb. 2005.

[122] F. R. Gantmacher and M. G. Krein, "Sur les matrices complètement non négatives et oscillatoires," *Compositio Mathematica*, vol. 4, pp. 445–476, 1937.

[123] R. Vandebril, M. Van Barel, G. Golub, and N. Mastronardi, "A bibliography on semiseparable matrices," *Calcolo*, vol. 43, no. 3-4, pp. 249–270, 2005.

[124] R. Kalman, "A new approach to linear filtering and prediction problems," *Trans. of the ASME - Journal of Basic Engineering*, vol. 82, no. 2, pp. 35–45, 1960.

[125] R. Kalman and R.Bucy, "New results in linear filtering and prediction theory," *ASME Journal of Basic Engineering*, vol. 83, pp. 95–108, 1961.

[126] B. Rao and H. Durrant-Whyte, "Fully decentralized algorithm for multisensor Kalman filtering," *IEE Proceedings-Control Theory and Applications*, vol. 138, pp. 413–420, Sep. 1991.

[127] V. Saligrama and D. Castanon, "Reliable distributed estimation with intermittent communications," in *45th IEEE Conference on Decision and Control*, San Diego, CA, Dec. 2006, pp. 6763–6768.

[128] T. Chung, V. Gupta, J. Burdick, and R. Murray, "On a decentralized active sensing strategy using mobile sensor platforms in a network," in *43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, Dec. 2004, vol. 2, pp. 1914–1919.

[129] H. Hashemipour, S. Roy, and A. Laub, "Decentralized structures for parallel Kalman filtering," *IEEE Trans. on Automatic Control*, vol. 33, no. 1, pp. 88–94, Jan. 1988.

[130] R. Olfati-Saber and J. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *44th IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005, pp. 6698 – 6703.

[131] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. Jordan, and S. Sastry, "Kalman filter with intermittent observations," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, Sep. 2004.

[132] T. Berg and H. Durrant-Whyte, "Model distribution in decentralized multi-sensor data fusion," Tech. Rep., University of Oxford, 1990.

[133] A. Mutambara, *Decentralized estimation and control for multisensor systems*, CRC Press, Boca Raton, FL, 1998.

[134] Dragoslav Šiljak, *Decentralized control of complex systems*, Academic Press Inc., Boston, MA, 1991.

[135] R. Grone, C. Johnson, E. Sa, and H. Wolkowicz, "Positive definite completions of partial Hermitian matrices," *Linear Algebra and its Applications*, vol. 58, pp. 109–124, Apr. 1984.

[136] A. B. Frakt, H. Lev-Ari, and A. S. Willsky, "A generalized Levinson algorithm for covariance extension with application to multiscale autoregressive modeling," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 411–424, Feb. 2003.

[137] B. Anderson and J. Moore, *Optimal filtering*, Prentice Hall, Englewood Cliffs, NJ, 1979.

[138] W. Ledsham and D. Staelin, "An extended Kalman-Bucy filter for atmospheric temperature profile retrieval with a passive microwave sounder," *Journal of Applied Meteorology*, vol. 17, pp. 1023–1033, Jul. 1978.

[139] R. Brammer, "Estimation of the ocean geoid near the Blake Escarpment using Geos-3 satellite altimetry data," *Journal of Geophysical Research*, vol. 84, no. B8, pp. 3843–3852, Jul. 1979.

[140] J. Galantowicz, D. Entekhabi, and E. Njoku, "Tests of sequential data assimilation for retrieving profile soil moisture and temperature from observed L-band radiobrightness," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 37, no. 4, pp. 1860–1870, Jul. 1999.

[141] M. Buehner and P. Malanotte-Rizzoli, "Reduced-rank Kalman filters applied to an idealized model of the wind-driven ocean circulation," *Journal of Geophysical Research*, vol. 108, no. C6, pp. 3192, 2003.

[142] M. Buehner, P. Malanotte-Rizzoli, A. Busalacchi, and T. Inui, "Estimation of the tropical Atlantic circulation from altimetry data using a reduced-rank stationary Kalman filter," *Interhemispheric water exchanges in the Atlantic Ocean, Elsevier Oceanographic Series*, vol. 68, no. 9, pp. 49–92, 2003.

[143] A. Graham, *Kronecker Products and Matrix Calculus with Applications*, Ellis Horwood, Chichester, UK, 1981.

[144] N. Motee and A. Jadbabaie, "Optimally control of spatially distributed systems," in *American Control Conference*, New York, NY, Jul. 2007, pp. 778–783.

[145] J. Brailean and A. Katsaggelos, "Simultaneous recursive displacement estimation and restoration of noisy-blurred image sequences," *IEEE Trans. on Image Processing*, vol. 4, no. 9, pp. 1236–1251, Sep. 1995.

[146] F. Khellah, P. Fieguth, M. Murray, and M. Allen, "Statistical processing of large image sequences," *IEEE Trans. on Image Processing*, vol. 14, no. 1, pp. 80–93, Jan. 2005.

[147] M. Ilic, E. Allen, J. Chapman, C. King, J. Lang, and E. Litvinov, "Preventing future blackouts by means of enhanced electric power systems control: From complexity to order," *Proceedings of the IEEE*, vol. 93, no. 11, pp. 1920–1941, Nov. 2005.

[148] T. Chin, W. Karl, and A. Willsky, "Sequential filtering for multi-frame visual reconstruction," *Special Issue on Multidimensional Signal Processing*, vol. 28, no. 3, pp. 311–333, 1992.

[149] W. W. Barrett and P. J. Feinsilver, "Gaussian families and a theorem on patterned matrices," *Journal of Applied Probability*, vol. 15, no. 3, pp. 514–522, Sep. 1978.

[150] H. Zhang, José M. F. Moura, and B. Krogh, "Estimation in sensor networks: A graph approach," in *4th International Symposium on Information Processing in Sensor Networks*, Los Angeles, CA, Apr. 2005, pp. 203–209.

[151] Béla Bollobás, *Modern graph theory*, Springer, New York, NY, 1998.

[152] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Distributed kalman filtering based on consensus strategies," Tech. Rep., Department of Information Engineering, University of Padova, 2007.

[153] Soummya Kar, Saeed Aldosari, and Jos M. F. Moura, "Topology for distributed inference on graphs," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2609–2613, June 2008.

[154] M. D. Ilić and J. Zaborszky, *Dynamics and Control of Large Electric Power Systems*, Wiley Interscience, New York, NY, 2000.

[155] Dragoslav Šiljak, *Large-scale Dynamic Systems: Stability and Structure*, North-Holland, New York, 1978.

[156] G. E. Shilov and R. A. Silverman, *Linear Algebra*, Courier Dover Publications, 1977.