CHAPTER

Distributed Kalman and Particle Filtering

1

Ali H. Sayed[*], **Petar M. Djurić**[a,**] **and F. Hlawatsch**[†]

[*]*Ecole Polytechnique Federale de Lausanne, EPFL, CH-1015 Lausanne;* [**]*Stony Brook University, Department of Electrical and Computer Engineering, Stony Brook, NY 11794, USA;* [†]*TU Wien, Institute of Telecommunications, 1040 Vienna, Austria*

[a]*Corresponding:* petar.djuric@stonybrook.edu

ABSTRACT

This chapter discusses distributed Kalman and particle filtering algorithms for state estimation in decentralized multi-agent networks. It is assumed that the spatially distributed agents acquire local measurements with information about a time-varying state described by some underlying state-space model. The agents seek to estimate the time-varying state in a decentralized manner. They are only allowed to interact locally by sharing data or estimates with their immediate neighbors. It is shown how the agents can construct local estimates of the state trajectory through a cooperative process of interactions. Both diffusion- and consensus-based strategies are presented.

Keywords: distributed sequential estimation, distributed Kalman filtering, distributed particle filtering, diffusion, distributed proposal adaptation, likelihood consensus, target tracking, wireless agent network.

1.1 DISTRIBUTED SEQUENTIAL STATE ESTIMATION

Distributed strategies can be applied to solving state-space filtering and smoothing problems [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. In these applications, agents interconnected by communication links seek to track the state of some underlying state-space model based on their own measurements and information ex-

changed with their neighbors.

Many natural and man-made systems can be modeled as dynamic systems described by state-space models [19, 20, 21]. The *state* of a dynamic system comprises certain system-related quantities of interest, such as the location and velocity of a moving vehicle or the concentration of a pollutant in a chemical plume. The state is time-varying and unknown. Each agent acquires local measurements and benefits from sharing information with neighboring agents. In this chapter, we consider distributed time-sequential methods for estimating the state from recurring measurements acquired by spatially distributed agents, where each agent is able to communicate only with a limited set of spatially close neighboring agents [1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18].

The objective of this chapter is to provide an overview of distributed state-space estimation for linear and nonlinear models, with emphasis on filtering and prediction problems while noting that similar techniques can be applied to smoothing problems [12]. In the first part of our treatment, we motivate and examine the diffusion Kalman filter for one-step prediction by following the general approach outlined in [12, 15, 18]. Similar diffusion strategies for fixed-lag and fixed-point smoothing appear in [12]. We focus on the diffusion Kalman form because it has been shown in [15, 18, 22, 23] that diffusion implementations have superior stability and tracking performance when optimizing aggregate costs in response to streaming data. In the second part of the chapter, we describe distributed particle filtering, where we focus on both consensus-based and diffusion-based approaches. The methods we present can be used in a wide range of applications including surveillance, localization and navigation, environmental and agricultural monitoring, target tracking, exploration, search and rescue, the Internet of Things, and logistics.

1.1.1 THE SETUP

The problem of interest is the estimation of a state $\boldsymbol{x}_n \in \mathbb{R}^N$, where $n \in \{1, 2, \ldots\}$ is a discrete time index, from sequences of measurements $\{\boldsymbol{z}_{1,k}, \boldsymbol{z}_{2,k}, \ldots, \boldsymbol{z}_{n,k}\}$, with $\boldsymbol{z}_{i,k} \in \mathbb{R}^p$, observed up to the current time n at K different agents $k \in \{1, 2, \ldots, K\}$. More specifically, we are interested in estimating $\boldsymbol{x}_n$ in a *time-sequential* and *distributed* manner. Time-sequential processing means that the estimation of $\boldsymbol{x}_n$ at time n recursively reuses relevant results from the estimation of $\boldsymbol{x}_{n-1}$ at time $n-1$ while incorporating the new measurements $\boldsymbol{z}_{n,k}$, $k = 1, 2, \ldots, K$. Distributed processing means that the estimation of $\boldsymbol{x}_n$ is done locally at the individual agents, which exchange relevant information in a spatially local manner. Such a distributed (decentralized) mode of operation has important advantages over a centralized mode [17, 16]. For example, it does not require a fusion center collecting all the measurements, which would constitute a "single point of failure." It also does not need communication between distant points or the use of complex routing strategies. The distributed solution is further robust to network node and link failures and is able to adapt to changing network topologies. Moreover, the computational complexity and communication cost per agent scale well with the network size (number of agents).

The topology of local information exchange is defined by a decentralized network of agents in which each agent k is able to communicate with a certain set of "neighbor" agents $\mathfrak{N}_k \subseteq \{1, 2, \ldots, K\}$. This set $\mathfrak{N}_k$ also includes agent k. Usually, the neighbors of an agent are located in its proximity. Practical examples of decentralized agent networks include robotic networks, networks of unmanned terrestrial, aerial or underwater vehicles, and networks of cameras. We assume that the agent network is connected, i.e., each agent is connected to any other agent via one or multiple "communication hops."

Since the measurements are dispersed among the agents rather than available at a single processing unit, an important aspect of any distributed estimation algorithm is the dissemination of agent-related (and possibly other) information through the network via local communication. Various dissemination schemes are available, such as consensus [24, 25], gossip [26, 27], and diffusion [12, 15, 18]. A performance benchmark for a distributed algorithm is the performance that would be achieved by a centralized algorithm that has direct access to all the measurements. The amount of communication that is required for good performance is an important property of a distributed algorithm, since communication increases the power consumption and processing delay of the agents.

A summary of the notation used in this chapter is presented in Table 1.1.

1.2 DISTRIBUTED KALMAN FILTERING

In the first part of our treatment, we motivate and examine the diffusion Kalman filter for one-step prediction by following the general approach outlined in [12, 15, 18]. We start with a brief description of the network topology and an introduction of the data model.

1.2.1 NETWORK TOPOLOGY

We consider a network consisting of K agents connected according to some graph topology. A left-stochastic $K \times K$ matrix $\boldsymbol{A} = (a_{\ell k})$ is associated with the topology, where each entry $a_{\ell k}$ denotes a nonnegative scaling factor that will be used to scale some of the data transitioning from agent ℓ to agent k (likewise for $a_{k\ell}$). The left-stochasticity of $\boldsymbol{A}$ means that it satisfies

$$\boldsymbol{A}^{\mathrm{T}} \mathbb{1}_K = \mathbb{1}_K, \tag{1.1}$$

so that the entries on each column of $\boldsymbol{A}$ add up to one (the symbol $\mathbb{1}_K$ represents a $K \times 1$ vector of ones). We assume that the graph is strongly-connected. The connectedness condition means that, for any two agents ℓ and k, there always exist paths with nonzero scaling weights in $\boldsymbol{A}$ linking ℓ and k in either direction; the path from ℓ to k does not need to agree with the reverse path from k to ℓ. Strong-connectedness additionally means that there exists at least one agent in the graph with $a_{kk} > 0$. That is,

Table 1.1 Summary of notation.

Notation	**Description**
$n \in \{1, 2, \ldots\}$	discrete time index
$\boldsymbol{x}_n \in \mathbb{R}^N$	state vector at time k
$k \in \{1, 2, \ldots, K\}$	agent index
K	number of agents
$\boldsymbol{z}_{n,k} \in \mathbb{R}^p$	measurement vector of agent k at time n
$\boldsymbol{z}_n \triangleq (\boldsymbol{z}_{n,1}^{\mathrm{T}} \boldsymbol{z}_{n,2}^{\mathrm{T}} \cdots \boldsymbol{z}_{n,K}^{\mathrm{T}})^{\mathrm{T}}$	vector of all agent measurements at time n
$\boldsymbol{z}_{1:n} \triangleq (\boldsymbol{z}_1^{\mathrm{T}} \boldsymbol{z}_2^{\mathrm{T}} \cdots \boldsymbol{z}_n^{\mathrm{T}})^{\mathrm{T}}$	all-agents measurement sequence up to time n
$\mathfrak{N}_k \subseteq \{1, 2, \ldots, K\}$	set of neighboring agents of agent k (including agent k)
$d_k = \lvert\mathfrak{N}_k\rvert$	cardinality of set $\mathfrak{N}_k$
$\boldsymbol{A}$	a left stochastic matrix
$\boldsymbol{J}$	adjacency matrix
$\mathbb{1}_K$	$K \times 1$ vector with elements equal to 1
δ_{nj}	Kronecker delta function
$\propto$	equal up to a constant factor
$\boldsymbol{F}_n$ and $\boldsymbol{G}_n$	state matrices of the linear state-space model
$\boldsymbol{H}_{n,k}$	local data matrix of the linear state-space model
$\boldsymbol{Q}_n$	covariance matrix of the state noise process
$\boldsymbol{R}_{n,k}$	covariance matrix of the observation noise process
$\boldsymbol{\Pi}_0$	covariance matrix of initial state
$\boldsymbol{I}_N$	$N \times N$ identity matrix
$\mathbf{g}_n(\boldsymbol{x}_{n-1}, \boldsymbol{u}_n)$	state-evolution function at time k
$\boldsymbol{u}_n$	state noise at time n
$f(\boldsymbol{x}_n \lvert \boldsymbol{x}_{n-1})$	state-transition probability density function
$\mathbf{h}_{n,k}(\boldsymbol{x}_n, \boldsymbol{v}_{n,k})$	observation function of agent k at time n
$\boldsymbol{v}_{n,k}$	observation noise of agent k at time n
$f(\boldsymbol{z}_{n,k} \lvert \boldsymbol{x}_n)$	local likelihood function of agent k
$f(\boldsymbol{z}_n \lvert \boldsymbol{x}_n)$	global (all-agents) likelihood function
$f(\boldsymbol{x}_n \lvert \boldsymbol{z}_{1:n})$	posterior probability density function
$\mathbb{E}[\cdot]$	expectation
$\boldsymbol{x}_n^{(m)}$	mth particle of the state $\mathbf{x}_n$
$w_n^{(m)}$	weight of the particle $\mathbf{x}_n^{(m)}$
M	number of particles
$q(\boldsymbol{x}_n \lvert \boldsymbol{x}_{n-1}^{(m)})$	importance probability density function
$\ln(\cdot)$	natural logarithm
$\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian probability density function

at least one agent places some trust or confidence on its local data. These conditions are satisfied by most graphs of interest. It follows from the strong-connectedness

condition that the left-stochastic matrix $\boldsymbol{A}$ is primitive [15].

1.2.2 LINEAR STATE-SPACE MODELS

We consider a linear state-space model defined by

$$\boldsymbol{x}_{n+1} = \boldsymbol{F}_n \boldsymbol{x}_n + \boldsymbol{G}_n \boldsymbol{u}_n, \quad n = 0, 1, \ldots, \tag{1.2}$$

$$\boldsymbol{z}_{n,k} = \boldsymbol{H}_{n,k} \boldsymbol{x}_n + \boldsymbol{v}_{n,k}, \quad n = 0, 1, \ldots; \; k = 1, 2, \ldots, K, \tag{1.3}$$

where $\boldsymbol{F}_n$ and $\boldsymbol{G}_n$ are state matrices, and $\boldsymbol{H}_{n,k}$ is a local data matrix. The noise vectors $\boldsymbol{u}_n$ and $\boldsymbol{v}_{n,k}$ are assumed to be realizations of zero-mean white processes with covariance matrices denoted by $\boldsymbol{Q}_n$ and $\boldsymbol{R}_{n,k}$, respectively,

$$\mathbb{E}\left[\begin{pmatrix} \boldsymbol{u}_n \\ \boldsymbol{v}_{n,k} \end{pmatrix}\begin{pmatrix} \boldsymbol{u}_j \\ \boldsymbol{v}_{j,k} \end{pmatrix}^{\mathrm{T}}\right] \stackrel{\Delta}{=} \begin{pmatrix} \boldsymbol{Q}_n & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_{n,k} \end{pmatrix}\delta_{nj}, \tag{1.4}$$

where δ_{nj} denotes the Kronecker delta: it is equal to one when $n = j$ and zero otherwise. The measurement noise vectors $\boldsymbol{v}_{n,k}$ are also assumed to be independent of each other across space (i.e., for different k), that is,

$$\mathbb{E}\,[\boldsymbol{v}_{n,k}\boldsymbol{v}_{j,\ell}^{\mathrm{T}}] = \boldsymbol{R}_{n,k}\delta_{k\ell}\delta_{nj}. \tag{1.5}$$

The initial state vector, $\boldsymbol{x}_0$, is assumed to have zero mean and covariance matrix

$$\mathbb{E}\,[\boldsymbol{x}_0\boldsymbol{x}_0^{\mathrm{T}}] = \boldsymbol{\Pi}_0 > \boldsymbol{0}, \tag{1.6}$$

and it is uncorrelated with $\boldsymbol{u}_n$ and $\boldsymbol{v}_{n,k}$, for all n and k. The notation $\boldsymbol{\Pi}_0 > \boldsymbol{0}$ signifies that $\boldsymbol{\Pi}_0$ is a positive definite matrix. We further assume that $\boldsymbol{R}_{n,k} > \boldsymbol{0}$. The parameter matrices $\{\boldsymbol{F}_n, \boldsymbol{G}_n, \boldsymbol{Q}_n, \boldsymbol{\Pi}_0, \boldsymbol{H}_{n,k}, \boldsymbol{R}_{n,k}\}$ are considered known by each agent k. Note that no Gaussian asssumptions are made.

1.2.3 NON-COOPERATIVE FILTERING

Assume initially that each agent k in the network acts individually and uses solely its local data stream $\boldsymbol{z}_{n,k}, n = 1, 2, \ldots$, to track the state vector $\boldsymbol{x}_n$. In this case, agent k can run the well-known Kalman filter in any of its various forms (covariance, measurement and time-update, or information form) to carry out this task [19, 28, 29]. For instance, let $\widehat{\boldsymbol{x}}_{n,k|j}^{\,\mathrm{ind}}$ denote the linear least-mean-squares error (LLMSE) estimate of $\boldsymbol{x}_n$ by agent k using all the local measurements up to time j, i.e., $\{\boldsymbol{z}_{1,k}, \boldsymbol{z}_{2,k}, \ldots, \boldsymbol{z}_{j,k}\}$. The superscript "ind" is used to emphasize that these estimators are based on individual (non-cooperative) behavior. We denote the corresponding estimation error and error-covariance matrix by

$$\widetilde{\boldsymbol{x}}_{n,k|j}^{\,\mathrm{ind}} \stackrel{\Delta}{=} \boldsymbol{x}_n - \widehat{\boldsymbol{x}}_{n,k|j}^{\,\mathrm{ind}}, \tag{1.7}$$

$$\boldsymbol{P}_{n,k|j}^{\mathrm{ind}} \stackrel{\Delta}{=} \mathbb{E}\left[\widetilde{\boldsymbol{x}}_{n,k|j}^{\,\mathrm{ind}}(\widetilde{\boldsymbol{x}}_{n,k|j}^{\,\mathrm{ind}})^{\mathrm{T}}\right]. \tag{1.8}$$

Then, it is well known [28, 29] that predicted (when $j = n - 1$) and/or filtered (when $j = n$) state estimates can be computed by agent k by one of several forms, listed below.

Covariance form of the non-cooperative Kalman filter

start with $\widehat{\boldsymbol{x}}^{\text{ind}}_{0,k|-1} = \boldsymbol{0}$, $\boldsymbol{P}^{\text{ind}}_{0,k|-1} = \boldsymbol{\Pi}_0$.

repeat at every agent k for $n \geq 0$:

$$
\begin{aligned}
\boldsymbol{e}^{\text{ind}}_{n,k} &= \boldsymbol{z}_{n,k} - \boldsymbol{H}_{n,k}\widehat{\boldsymbol{x}}^{\text{ind}}_{n,k|n-1} \\
\boldsymbol{R}^{\text{ind}}_{e,n,k} &= \boldsymbol{R}_{n,k} + \boldsymbol{H}_{n,k}\boldsymbol{P}^{\text{ind}}_{n,k|n-1}\boldsymbol{H}^{\text{T}}_{n,k} \\
\boldsymbol{K}^{\text{ind}}_{p,n,k} &= \boldsymbol{F}_n\boldsymbol{P}^{\text{ind}}_{n,k|n-1}\boldsymbol{H}^{\text{T}}_{n,k}\left(\boldsymbol{R}^{\text{ind}}_{e,n,k}\right)^{-1} \\
\widehat{\boldsymbol{x}}^{\text{ind}}_{n+1,k|n} &= \boldsymbol{F}_n\widehat{\boldsymbol{x}}^{\text{ind}}_{n,k|n-1} + \boldsymbol{K}^{\text{ind}}_{p,n,k}\boldsymbol{e}^{\text{ind}}_{n,k} \\
\boldsymbol{P}^{\text{ind}}_{n+1,k|n} &= \boldsymbol{F}_n\boldsymbol{P}^{\text{ind}}_{n,k|n-1}\boldsymbol{F}^{\text{T}}_n - \boldsymbol{K}^{\text{ind}}_{p,n,k}\boldsymbol{R}^{\text{ind}}_{e,n,k}\left(\boldsymbol{K}^{\text{ind}}_{p,n,k}\right)^{\text{T}} + \boldsymbol{G}_n\boldsymbol{Q}_n\boldsymbol{G}^{\text{T}}_n
\end{aligned}
\tag{1.9}
$$

end

Measurement and time-update form of the non-cooperative Kalman filter

start with $\widehat{\boldsymbol{x}}^{\text{ind}}_{0,k|-1} = \boldsymbol{0}$, $\boldsymbol{P}^{\text{ind}}_{0,k|-1} = \boldsymbol{\Pi}_0$.

repeat at every agent k for $n \geq 0$:

(measurement-update)

$$
\begin{aligned}
\boldsymbol{e}^{\text{ind}}_{n,k} &= \boldsymbol{z}_{n,k} - \boldsymbol{H}_{n,k}\widehat{\boldsymbol{x}}^{\text{ind}}_{n,k|n-1} \\
\boldsymbol{R}^{\text{ind}}_{e,n,k} &= \boldsymbol{R}_{n,k} + \boldsymbol{H}_{n,k}\boldsymbol{P}^{\text{ind}}_{n,k|n-1}\boldsymbol{H}^{\text{T}}_{n,k} \\
\widehat{\boldsymbol{x}}^{\text{ind}}_{n,k|n} &= \widehat{\boldsymbol{x}}^{\text{ind}}_{n,k|n-1} + \boldsymbol{P}^{\text{ind}}_{n,k|n-1}\boldsymbol{H}^{\text{T}}_{n,k}\left(\boldsymbol{R}^{\text{ind}}_{e,n,k}\right)^{-1}\boldsymbol{e}^{\text{ind}}_{n,k} \\
\boldsymbol{P}^{\text{ind}}_{n,k|n} &= \boldsymbol{P}^{\text{ind}}_{n,k|n-1} - \boldsymbol{P}^{\text{ind}}_{n,k|n-1}\boldsymbol{H}^{\text{T}}_{n,k}\left(\boldsymbol{R}^{\text{ind}}_{e,n,k}\right)^{-1}\boldsymbol{H}_{n,k}\boldsymbol{P}^{\text{ind}}_{n,k|n-1}
\end{aligned}
\tag{1.10}
$$

(time-update)

$$
\begin{aligned}
\widehat{\boldsymbol{x}}^{\text{ind}}_{n+1,k|n} &= \boldsymbol{F}_n\widehat{\boldsymbol{x}}^{\text{ind}}_{n,k|n} \\
\boldsymbol{P}^{\text{ind}}_{n+1,k|n} &= \boldsymbol{F}_n\boldsymbol{P}^{\text{ind}}_{n,k|n}\boldsymbol{F}^{\text{T}}_n + \boldsymbol{G}_n\boldsymbol{Q}_n\boldsymbol{G}^{\text{T}}_n
\end{aligned}
$$

end

Information form of the non-cooperative Kalman filter

start with $\widehat{\boldsymbol{x}}^{\text{ind}}_{k,0|-1} = \boldsymbol{0}, \left(\boldsymbol{P}^{\text{ind}}_{k,0|-1}\right)^{-1} = \boldsymbol{\Pi}_0^{-1}$.

repeat at every agent k for $n \geq 0$:

$$
\begin{aligned}
\left(\boldsymbol{P}^{\text{ind}}_{n,k|n}\right)^{-1} &= \left(\boldsymbol{P}^{\text{ind}}_{n,k|n-1}\right)^{-1} + \boldsymbol{H}^{\text{T}}_{n,k}\boldsymbol{R}^{-1}_{n,k}\boldsymbol{H}_{n,k} \\
\left(\boldsymbol{P}^{\text{ind}}_{n,k|n}\right)^{-1}\widehat{\boldsymbol{x}}^{\text{ind}}_{n,k|n} &= \left(\boldsymbol{P}^{\text{ind}}_{n,k|n-1}\right)^{-1}\widehat{\boldsymbol{x}}^{\text{ind}}_{n,k|n-1} + \boldsymbol{H}^{\text{T}}_{n,k}\boldsymbol{R}^{-1}_{n,k}\boldsymbol{z}_{n,k} \\
\widehat{\boldsymbol{x}}^{\text{ind}}_{n,k+1|n} &= \boldsymbol{F}_n\widehat{\boldsymbol{x}}^{\text{ind}}_{n,k|n} \\
\boldsymbol{P}^{\text{ind}}_{n+1,k|n} &= \boldsymbol{F}_n\boldsymbol{P}^{\text{ind}}_{n,k|n}\boldsymbol{F}^{\text{T}}_n + \boldsymbol{G}_n\boldsymbol{Q}_n\boldsymbol{G}^{\text{T}}_n
\end{aligned}
\tag{1.11}
$$

end

Clearly, these non-cooperative forms assume that the agents act individually.

However, since all the agents are tracking the same state vector $\boldsymbol{x}_n$, it is expected that the mean-square-error performance can improve through cooperation by having neighboring agents share with each other information about their own estimators. We consider one initial form of cooperation before extending it to arrive at the diffusion Kalman filter later in Subsection 1.2.7.

1.2.4 INCREMENTAL COOPERATION

Consider an arbitrary agent k and let $\mathfrak{N}_k$ denote its neighborhood, i.e., the collection of all agents ℓ that are connected to agent k by edges and that can share information with it. This set includes also the agent k. The set comprises all agents ℓ for which $a_{\ell k} > 0$; its cardinality is denoted by $d_k = |\mathfrak{N}_k|$.

In addition to its own measurement vector $\boldsymbol{z}_{n,k}$, at each time instant n, agent k has also access to the measurement vectors $\{\boldsymbol{z}_{n,\ell}\}$ from its neighbors, $\ell \in \mathfrak{N}_k \backslash \{k\}$. In this way, at every time instant n, agent k has access to $|\mathfrak{N}_k|$ measurement vectors (its own measurements and the ones collected from its neighbors) and denoted by $\{\boldsymbol{z}_{n,k_1}, \boldsymbol{z}_{n,k_2}, \ldots, \boldsymbol{z}_{n,k_{d_k}}\}$. In this notation, we are referring to the neighbors of agent k (including agent k itself) by the indices $\{k_1, k_2, \ldots, k_{d_k}\}$. Let $\widehat{\boldsymbol{x}}^{\,\mathrm{loc}}_{n,k|j}$ denote the LLMSE estimate of $\boldsymbol{x}_n$ obtained by agent k (including agent k) by using these local (or neighborhood) measurements from time 0 up to time j. The superscript "loc" is used to emphasize that these estimators are based on local neighborhood measurements. We denote the corresponding estimation error and error-covariance matrix by

$$\widetilde{\boldsymbol{x}}^{\,\mathrm{loc}}_{n,k|j} \triangleq \boldsymbol{x}_n - \widehat{\boldsymbol{x}}^{\,\mathrm{loc}}_{n,k|j}, \tag{1.12}$$

$$\boldsymbol{P}^{\mathrm{loc}}_{n,k|j} \triangleq \mathbb{E}\left[\widetilde{\boldsymbol{x}}^{\,\mathrm{loc}}_{n,k|j}(\widetilde{\boldsymbol{x}}^{\,\mathrm{loc}}_{n,k|j})^{\mathrm{T}}\right] \tag{1.13}$$

Then, as shown in listing (1.14), the predicted (when $j = n-1$) and filtered (when $j = n$) state estimates can be computed by agent k by running multiple measurement-updates, one for each neighbor of k [28, 29]. In this listing, we are denoting the state estimate at the end of the incremental loop through the neighborhood by $\boldsymbol{\psi}_{n,k}$, which is equal to the filtered estimate $\widehat{\boldsymbol{x}}^{\,\mathrm{loc}}_{n,k|n}$ that results from the d_k measurement updates. This equality is highlighted by the symbol $(\star)$ placed next to one of the equations in listing (1.14). The corresponding error covariance matrix of $\widehat{\boldsymbol{x}}^{\,\mathrm{loc}}_{n,k|n}$ is denoted by $\boldsymbol{P}^{\mathrm{loc}}_{n,k|n}$. Although the variables $\boldsymbol{\psi}_{n,k}$ and $\widehat{\boldsymbol{x}}^{\,\mathrm{loc}}_{n,k|n}$ are identical in the incremental implementation, we keep separate notations for both symbols because they will be distinct and will play different roles in the diffusion form derived further ahead in Subsection. 1.2.7. In particular, the $(\star)$ assignment will be replaced by (1.48).

Measurement and time-update incremental form

start with $\widehat{\boldsymbol{x}}^{\rm loc}_{0,k|-1} = \boldsymbol{0}$, $\boldsymbol{P}^{\rm loc}_{0,k|-1} = \boldsymbol{\Pi}_0$.

repeat at every agent k for $n \geq 0$:

(incremental measurement-updates)

set auxiliary variables:

$$\boldsymbol{\psi}^{(0)}_{n,k} \leftarrow \widehat{\boldsymbol{x}}^{\rm loc}_{n,k|n-1} \quad (N \times 1 \text{ vector})$$

$$\boldsymbol{P}^{(0)}_{n,k} \leftarrow \boldsymbol{P}^{\rm loc}_{n,k|n-1} \quad (N \times N \text{ matrix})$$

for each neighbor $m = 1, 2, \ldots, d_k$ do:

$$\begin{aligned}
&\boldsymbol{e}_{n,k_m} = \boldsymbol{z}_{n,k_m} - \boldsymbol{H}_{n,k_m}\boldsymbol{\psi}^{(m-1)}_{n,k} \\
&\boldsymbol{R}_{e,n,k_m} = \boldsymbol{R}_{n,k_m} + \boldsymbol{H}_{n,k_m}\boldsymbol{P}^{(m-1)}_{n,k}\boldsymbol{H}^{\rm T}_{n,k_m} \\
&\boldsymbol{\psi}^{(m)}_{n,k} = \boldsymbol{\psi}^{(m-1)}_{n,k} + \boldsymbol{P}^{(m-1)}_{n,k}\boldsymbol{H}^{\rm T}_{n,k_m}\boldsymbol{R}^{-1}_{e,n,k_m}\boldsymbol{e}_{n,k_m} \\
&\boldsymbol{P}^{(m)}_{n,k} = \boldsymbol{P}^{(m-1)}_{n,k} - \boldsymbol{P}^{(m-1)}_{n,k}\boldsymbol{H}^{\rm T}_{n,k_m}\boldsymbol{R}^{-1}_{e,n,k_m}\boldsymbol{H}_{n,k_m}\boldsymbol{P}^{(m-1)}_{n,k}
\end{aligned} \tag{1.14}$$

end

$$\boldsymbol{P}^{\rm loc}_{n,k|n} \leftarrow \boldsymbol{P}^{(d_k)}_{n,k}$$

$$\boldsymbol{\psi}_{n,k} \leftarrow \boldsymbol{\psi}^{(d_k)}_{n,k}$$

$$\widehat{\boldsymbol{x}}^{\rm loc}_{n,k|n} \leftarrow \boldsymbol{\psi}^{(d_k)}_{n,k} \qquad (\star)$$

(time-update)

$$\begin{aligned}
&\widehat{\boldsymbol{x}}^{\rm loc}_{n+1,k|n} = \boldsymbol{F}_n\widehat{\boldsymbol{x}}^{\rm loc}_{n,k|n} \\
&\boldsymbol{P}^{\rm loc}_{n+1,k|n} = \boldsymbol{F}_n\boldsymbol{P}^{\rm loc}_{n,k|n}\boldsymbol{F}^{\rm T}_n + \boldsymbol{G}_n\boldsymbol{Q}_n\boldsymbol{G}^{\rm T}_n
\end{aligned}$$

end

It can be verified that the following relations hold for the variables in the incremental form (1.14) — see the proof in [12, Appendix B]:

$$\left(\boldsymbol{P}^{\rm loc}_{n,k|n}\right)^{-1} = \left(\boldsymbol{P}^{\rm loc}_{n,k|n-1}\right)^{-1} + \boldsymbol{S}_{n,k}, \tag{1.15}$$

with

$$\boldsymbol{S}_{n,k} \stackrel{\Delta}{=} \sum_{\ell \in \mathfrak{N}_k} \boldsymbol{H}^{\rm T}_{n,\ell}\boldsymbol{R}^{-1}_{n,\ell}\boldsymbol{H}_{n,\ell}, \tag{1.16}$$

and

$$\left(\boldsymbol{P}^{\rm loc}_{n,k|n}\right)^{-1}\widehat{\boldsymbol{x}}^{\rm loc}_{n,k|n} = \left(\boldsymbol{P}^{\rm loc}_{n,k|n-1}\right)^{-1}\widehat{\boldsymbol{x}}^{\rm loc}_{n,k|n-1} + \boldsymbol{q}_{n,k}, \tag{1.17}$$

with

$$\boldsymbol{q}_{n,k} \stackrel{\Delta}{=} \sum_{\ell \in \mathfrak{N}_k} \boldsymbol{H}^{\rm T}_{n,\ell}\boldsymbol{R}^{-1}_{n,\ell}\boldsymbol{z}_{n,\ell}. \tag{1.18}$$

Recursions (1.14) compute the optimal local estimate $\widehat{\boldsymbol{x}}^{\rm loc}_{n,k|n}$ of agent k by incorporating solely the measurements $\{\boldsymbol{z}_{n,\ell}\}$ from the neighborhood $\mathfrak{N}_k$. This is a limited form of cooperation since the recursions (1.14) are not exploiting the fact that be-

sides the measurements $\{\boldsymbol{z}_{n,\ell}\}$, the neighbors of agent k also have their own estimates of $\boldsymbol{x}_n$. These estimates are given by $\widehat{\boldsymbol{x}}^{\text{loc}}_{n,\ell|n}$. It is expected that by additionally exploiting the neighborhood estimates, agent k should be able to achieve two objectives: (a) enhance its state estimation accuracy and (b) have its local estimate $\widehat{\boldsymbol{x}}^{\text{loc}}_{n,\ell|n}$ approach the desired global estimate $\widehat{\boldsymbol{x}}_{n|n}$ of the state vector $\boldsymbol{x}_n$ (which would be produced by an optimum centralized estimator that has access to all the measurements in the network). We shall use this observation to motivate and derive the diffusion Kalman filter. First, however, we take a brief digression and review a well-known data fusion construction [28, 29], which will be used to form *approximate* local estimates.

1.2.5 DATA FUSION

A problem that is encountered often in applications deals with the need to fuse together data collected from several sources in order to enhance the accuracy of the estimation process. This is similar to the scenario we are facing in the network context, except that the graph topology and the time evolution of the data add a new level of complexity in the form of cross-correlations.

Thus, assume for a moment that we have a collection of K agents that are distributed over some region in space without paying particular attention to whether they are connected by a communication link or not. All agents are interested in estimating the same parameter vector of size $N \times 1$, denoted generically by $\boldsymbol{x}$, which is assumed to be zero-mean and to have a positive definite covariance matrix $\boldsymbol{\Pi}_x = \mathbb{E}\,[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}]$. For example, the agents could be tracking a moving object with the goal of estimating its speed and direction of motion. Each agent k collects a measurement vector $\boldsymbol{z}_k$ of size $p \times 1$ that is related to the desired $\boldsymbol{x}$ via the linear measurement model (observe that in the discussion in this section we are not dealing with sequential processing of the data and, therefore, the time index is dropped),

$$\boldsymbol{z}_k = \boldsymbol{H}_k \boldsymbol{x} + \boldsymbol{v}_k, \tag{1.19}$$

where $\boldsymbol{H}_k$ is the model matrix that maps $\boldsymbol{x}$ to $\boldsymbol{z}_k$ at agent k and $\boldsymbol{v}_k$ is zero-mean measurement noise with a positive definite covariance matrix $\boldsymbol{R}_k = \mathbb{E}\,[\boldsymbol{v}_k \boldsymbol{v}_k^{\mathrm{T}}]$.

Assume initially that the measurements from all agents can be collected centrally at some fusion center. For example, each agent k could transmit its measurement vector $\boldsymbol{z}_k$ and its model parameters $\{\boldsymbol{H}_k, \boldsymbol{R}_k\}$ to the fusion center. The data collected at the fusion center thus satisfy the linear model:

$$\underbrace{\begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \\ \vdots \\ \boldsymbol{z}_K \end{pmatrix}}_{\triangleq \boldsymbol{z}} = \underbrace{\begin{pmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_2 \\ \vdots \\ \boldsymbol{H}_K \end{pmatrix}}_{\triangleq \boldsymbol{H}} \boldsymbol{x} + \underbrace{\begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_K \end{pmatrix}}_{\triangleq \boldsymbol{v}}. \tag{1.20}$$

We assume that the noises $\{\boldsymbol{v}_k\}$ across all agents k are *uncorrelated* with each other

so that the covariance matrix of the aggregate noise vector $\boldsymbol{v}$ is block diagonal and given by

$$\boldsymbol{R}_v = \text{blkdiag}\{\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_K\}. \tag{1.21}$$

Now, it is well known that the LLMSE estimate of $\boldsymbol{x}$ that is based on $\boldsymbol{z}$ is given by [28, 29]

$$\widehat{\boldsymbol{x}} = \left(\boldsymbol{\Pi}_x^{-1} + \boldsymbol{H}^{\mathrm{T}}\boldsymbol{R}_v^{-1}\boldsymbol{H}\right)^{-1} \boldsymbol{H}^{\mathrm{T}}\boldsymbol{R}_v^{-1}\boldsymbol{z}, \tag{1.22}$$

with the resulting minimum mean-square-error (MMSE) matrix equal to

$$\boldsymbol{P} \triangleq \mathbb{E}\,[\widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{x}}^{\mathrm{T}}] = \left(\boldsymbol{\Pi}_x^{-1} + \boldsymbol{H}^{\mathrm{T}}\boldsymbol{R}_v^{-1}\boldsymbol{H}\right)^{-1}, \tag{1.23}$$

where $\widetilde{\boldsymbol{x}} = \boldsymbol{x} - \widehat{\boldsymbol{x}}$ denotes the estimation error. Note that $\boldsymbol{P}$ is a matrix of size $N \times N$.

The solution (1.22) requires all agents to transmit their *raw* data $\{\boldsymbol{z}_k, \boldsymbol{H}_k, \boldsymbol{R}_k\}$ to the fusion center. A more efficient fusion method, one that reduces the communications overhead, can be derived by allowing the agents to perform some local processing and to share the results of this processing step rather than their raw data. Specifically, assume that each agent estimates $\boldsymbol{x}$ using first its own data $\boldsymbol{z}_k$. We denote the resulting estimator by $\widehat{\boldsymbol{x}}_k$ and it is given by

$$\widehat{\boldsymbol{x}}_k = \left(\boldsymbol{\Pi}_x^{-1} + \boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{R}_k^{-1}\boldsymbol{H}_k\right)^{-1} \boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{R}_k^{-1}\boldsymbol{z}_k. \tag{1.24}$$

The corresponding MMSE is

$$\boldsymbol{P}_k = \left(\boldsymbol{\Pi}_x^{-1} + \boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{R}_k^{-1}\boldsymbol{H}_k\right)^{-1}. \tag{1.25}$$

Next, assume that the agents share the processed data $\{\widehat{\boldsymbol{x}}_k, \boldsymbol{P}_k\}$ with the fusion center instead of the raw data $\{\boldsymbol{z}_k, \boldsymbol{H}_k, \boldsymbol{R}_k\}$. We now show that the desired global quantities $\{\widehat{\boldsymbol{x}}, \boldsymbol{P}\}$ can be recovered directly from the locally generated data $\{\widehat{\boldsymbol{x}}_k, \boldsymbol{P}_k\}$.

To begin with, observe that we can rework expression (1.23) for the global MMSE matrix as follows:

$$\boldsymbol{P}^{-1} = \boldsymbol{\Pi}_x^{-1} + \sum_{k=1}^{K} \boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{R}_k^{-1}\boldsymbol{H}_k = \sum_{k=1}^{K} \boldsymbol{P}_k^{-1} - (K-1)\boldsymbol{\Pi}_x^{-1}. \tag{1.26}$$

This expression allows the fusion center to determine $\boldsymbol{P}^{-1}$ directly from knowledge of the quantities $\{\boldsymbol{P}_k^{-1}, \boldsymbol{\Pi}_x^{-1}\}$. Note further that the global estimate expression (1.22) can be rewritten as

$$\boldsymbol{P}^{-1}\widehat{\boldsymbol{x}} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{R}_v^{-1}\boldsymbol{z} = \sum_{k=1}^{K} \boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{R}_k^{-1}\boldsymbol{z}_k = \sum_{k=1}^{K} \boldsymbol{P}_k^{-1}\widehat{\boldsymbol{x}}_k. \tag{1.27}$$

Therefore, we arrive at the following alternative method to fuse the data from multi-

ple agents:

$$\boldsymbol{P}^{-1} = \sum_{k=1}^{K} \boldsymbol{P}_k^{-1} - (K-1)\boldsymbol{\Pi}_x^{-1}, \tag{1.28}$$

$$\boldsymbol{P}^{-1}\widehat{\boldsymbol{x}} = \sum_{k=1}^{K} \boldsymbol{P}_k^{-1}\widehat{\boldsymbol{x}}_k. \tag{1.29}$$

Observe that the individual estimates are scaled by the inverses of their MMSE matrices. It is useful to note that the derivation of these expressions requires the noises across the agents to be uncorrelated with each other, so that $\boldsymbol{R}_v$ is block diagonal.

1.2.6 APPROXIMATE FUSION RELATIONS

Let us now return to the network context and use the fusion expressions (1.28) and (1.29) to motivate the diffusion Kalman filter. Let $\widehat{\boldsymbol{x}}_{n|n}$ denote the filtered estimate of $\boldsymbol{x}_n$ that is based on the observations $\boldsymbol{z}_{j,k}$ from across all agents $k = 1, 2, \ldots, K$ at all times j up to time n. Let $\boldsymbol{P}_{n|n}$ denote the corresponding error covariance matrix, i.e.,

$$\boldsymbol{P}_{n|n} \triangleq \mathbb{E}\,[\widetilde{\boldsymbol{x}}_{n|n}\widetilde{\boldsymbol{x}}_{n|n}^{\mathrm{T}}], \quad \text{where} \quad \widetilde{\boldsymbol{x}}_{n|n} \triangleq \boldsymbol{x}_n - \widehat{\boldsymbol{x}}_{n|n}. \tag{1.30}$$

These are the global quantities that we are interested in evaluating in a distributed manner. We can employ the fusion expressions (1.28) and (1.29) to approximate these global variables in terms of the non-cooperative variables $\{\widehat{\boldsymbol{x}}_{n,k|n}^{\mathrm{ind}}, \boldsymbol{P}_{n,k|n}^{\mathrm{ind}}\}$ as follows:

$$\boldsymbol{P}_{n|n}^{-1} \approx \sum_{k=1}^{K} \left(\boldsymbol{P}_{k,n|n}^{\mathrm{ind}}\right)^{-1} - (K-1)\boldsymbol{\Pi}_n^{-1}, \tag{1.31}$$

$$\boldsymbol{P}_{n|n}^{-1}\,\widehat{\boldsymbol{x}}_{n|n} \approx \sum_{k=1}^{K} \left(\boldsymbol{P}_{n,k|n}^{\mathrm{ind}}\right)^{-1} \widehat{\boldsymbol{x}}_{n,k|n}^{\mathrm{ind}}, \tag{1.32}$$

where $\boldsymbol{\Pi}_n \triangleq \mathbb{E}\,[\boldsymbol{x}_n\boldsymbol{x}_n^{\mathrm{T}}]$ denotes the covariance matrix of $\boldsymbol{x}_n$, which satisfies the recursion

$$\boldsymbol{\Pi}_{n+1} = \boldsymbol{F}_n\boldsymbol{\Pi}_n\boldsymbol{F}_n^{\mathrm{T}} + \boldsymbol{G}_n\boldsymbol{Q}_n\boldsymbol{G}_n^{\mathrm{T}}. \tag{1.33}$$

Relations (1.31) and (1.32) are approximate expressions, while relations (1.28) and (1.29) are exact fusion formulae. To see why, recall that the global estimate $\widehat{\boldsymbol{x}}_{n|n}$ is based on the observations $\boldsymbol{z}_{j,k}$ from time $j = 0$ up to time $j = n$ across all agents k. If we now appeal to the state-space model described by (1.2) and (1.3), it is easy to recognize that these observations cannot generally be related directly to the unknown $\boldsymbol{x}_n$ via a linear model of the form (1.19). Exact fusion formulae in this case are cumbersome and require the use of error cross-covariance matrices (see, e.g., [30, 31, 32]). The above approximations, however, are sufficient for our purposes and they are commonly used in the literature on data fusion methods, including the

following simpler form:

$$\boldsymbol{P}_{n|n}^{-1} \approx \sum_{k=1}^{K} \left(\boldsymbol{P}_{k,n|n}^{\text{ind}}\right)^{-1}, \tag{1.34}$$

where the term involving $\boldsymbol{\Pi}_n^{-1}$ is ignored. Expression (1.32) amounts to estimating $\widehat{\boldsymbol{x}}_{n|n}$ by using a covariance-weighted combination of the local estimates, $\widehat{\boldsymbol{x}}_{n,k|n}^{\text{ind}}$, while expression (1.34) estimates the error covariance matrix by pooling together the individual error covariances.

In principle, the same fusion expressions (1.28) and (1.29) could also be used to approximate the same global variables $\{\widehat{\boldsymbol{x}}_{n|n}, \boldsymbol{P}_{n|n}\}$ in terms of the incremental (neighborhood) estimates $\{\boldsymbol{\psi}_{n,k}, \boldsymbol{P}_{n,k|n}^{\text{loc}}\}$ defined in Subsection 1.2.4, say as:

$$\boldsymbol{P}_{n|n}^{-1}\widehat{\boldsymbol{x}}_{n|n} \approx \sum_{k=1}^{K} \left(\boldsymbol{P}_{n,k|n}^{\text{loc}}\right)^{-1} \boldsymbol{\psi}_{n,k}, \tag{1.35}$$

$$\boldsymbol{P}_{n|n}^{-1} \approx \sum_{k=1}^{K} \left(\boldsymbol{P}_{n,k|n}^{\text{loc}}\right)^{-1}, \tag{1.36}$$

where we recall that $\boldsymbol{\psi}_{n,k} = \widehat{\boldsymbol{x}}_{n,k|n}^{\text{loc}}$. However, observe now that the construction of the estimates $\boldsymbol{\psi}_{n,k}$ across agents relies on shared observations. For example, the observations from both agent k and its neighbors influence the value of $\boldsymbol{\psi}_{n,k}$; likewise, some of these same observations influence the value of other estimates because they belong to the neighborhoods of other agents as well. This data redundancy is not present in the computation of the individual estimates $\widehat{\boldsymbol{x}}_{n,k|n}^{\text{ind}}$, where each agent k relies solely on its local sequential data $\boldsymbol{z}_{n,k}$.

The presence of redundant information in the intermediate estimates $\boldsymbol{\psi}_{n,k}$ is due to the graph topology, which defines the neighborhoods over the network. We can exploit this topology to replace the approximations (1.35) and (1.36) by more revealing relations that bring forth the graph structure. To do so, we let $\boldsymbol{J}$ denote the $K \times K$ adjacency matrix of the network graph. This matrix consists of unit and zero entries representing the connectivity of the agents, namely,

$$J_{\ell k} = \begin{cases} 1, & \text{if } a_{\ell k} > 0 \\ 0, & \text{otherwise} \end{cases}. \tag{1.37}$$

Although unnecessary in the final statement of the algorithm, we shall assume for the sake of the derivation that there exists a vector $\boldsymbol{s}$ such that $\boldsymbol{J}\boldsymbol{s} = \mathbb{1}_K$. For example, if $\boldsymbol{J}$ happens to be invertible, then this vector $\boldsymbol{s}$ exists, is unique, and is given by $\boldsymbol{s} = \boldsymbol{J}^{-1}\mathbb{1}_K$. In general, however, it is not always guaranteed that a vector $\boldsymbol{s}$ satisfying $\boldsymbol{J}\boldsymbol{s} = \mathbb{1}_K$ exists. It is nevertheless possible to verify that the matrix $\boldsymbol{J}$ can be made invertible by flipping some of its diagonal entries (from zero to one and from one to zero). By turning a diagonal entry of $\boldsymbol{J}$ from zero to one, we are in effect allowing the agent at that location to rely on its local measurement. On the other hand, by turning a diagonal entry of $\boldsymbol{J}$ from one to zero, we are disabling this feature. This

action is tolerable because this measurement is not discarded by the network since it is still used by the neighbors of the agent to update their estimates (as long as at least one entry of the row of $\boldsymbol{J}$ with a diagonal entry equal to zero has a value equal to one). We denote the entries of $\boldsymbol{s}$ by γ_k, $k = 1, 2, \ldots, K$, or $\boldsymbol{s} = \mathrm{col}\{\gamma_k\}$ so that

$$\boldsymbol{J}\boldsymbol{s} = \mathbb{1}_K \iff \sum_{k=1}^{K} \gamma_k J_{\ell k} = 1, \quad \text{for every } \ell = 1, 2, \ldots, K. \tag{1.38}$$

In a manner similar to (1.32) and (1.34), we first use the fusion expressions (1.28) and (1.29) to approximate the incremental estimators $\{\boldsymbol{\psi}_{n,k}, \boldsymbol{P}^{\mathrm{loc}}_{n,k|n}\}$ by using the non-cooperative variables $\{\widehat{\boldsymbol{x}}^{\mathrm{ind}}_{n,k|n}, \boldsymbol{P}^{\mathrm{ind}}_{n,k|n}\}$ as follows:

$$\left(\boldsymbol{P}^{\mathrm{loc}}_{n,k|n}\right)^{-1} \boldsymbol{\psi}_{n,k} \approx \sum_{\ell=1}^{K} J_{\ell k} \left(\boldsymbol{P}^{\mathrm{ind}}_{n,\ell|n}\right)^{-1} \widehat{\boldsymbol{x}}^{\mathrm{ind}}_{n,\ell|n}, \tag{1.39}$$

$$\left(\boldsymbol{P}^{\mathrm{loc}}_{n,k|n}\right)^{-1} \approx \sum_{\ell=1}^{K} J_{\ell k} \left(\boldsymbol{P}^{\mathrm{ind}}_{n,\ell|n}\right)^{-1}. \tag{1.40}$$

Recall that the individual estimates $\widehat{\boldsymbol{x}}^{\mathrm{ind}}_{n,k|n}$ rely on independent streams of data and that $J_{\ell k} = 1$ if information can flow from agent ℓ to agent k, and $J_{\ell k} = 0$ otherwise.

Using the above relations we can now relate the global variables $\{\widehat{\boldsymbol{x}}_{n|n}, \boldsymbol{P}_{n|n}\}$ to the local variables $\{\boldsymbol{\psi}_{n,k}, \boldsymbol{P}^{\mathrm{loc}}_{n,k|n}\}$ and replace (1.35) and (1.36). Indeed, using the entries γ_k, we note that

$$\begin{aligned}\sum_{k=1}^{K} \gamma_k \left(\boldsymbol{P}^{\mathrm{loc}}_{n,k|n}\right)^{-1} \boldsymbol{\psi}_{n,k} &\overset{(1.39)}{\approx} \sum_{\ell=1}^{K} \underbrace{\left(\sum_{k=1}^{K} \gamma_k J_{\ell k}\right)}_{=1} \left(\boldsymbol{P}^{\mathrm{ind}}_{n,\ell|n}\right)^{-1} \widehat{\boldsymbol{x}}^{\mathrm{ind}}_{n,\ell|n} \\ &= \sum_{\ell=1}^{K} \left(\boldsymbol{P}^{\mathrm{ind}}_{n,\ell|n}\right)^{-1} \widehat{\boldsymbol{x}}^{\mathrm{ind}}_{n,\ell|n} \overset{(1.32)}{\approx} \boldsymbol{P}^{-1}_{n|n} \widehat{\boldsymbol{x}}_{n|n}\end{aligned} \tag{1.41}$$

and, similarly,

$$\sum_{k=1}^{K} \gamma_k \left(\boldsymbol{P}^{\mathrm{loc}}_{n,k|n}\right)^{-1} \overset{(1.40)}{\approx} \sum_{\ell=1}^{K} \left(\boldsymbol{P}^{\mathrm{ind}}_{n,\ell|n}\right)^{-1} \overset{(1.34)}{\approx} \boldsymbol{P}^{-1}_{n|n}. \tag{1.42}$$

In summary, we arrive at the following approximate relations between the global variables $\{\widehat{\boldsymbol{x}}_{n|n}, \boldsymbol{P}_{n|n}\}$ and the neighborhood variables $\{\boldsymbol{\psi}_{n,k}, \boldsymbol{P}^{\mathrm{loc}}_{n,k|n}\}$:

$$\boldsymbol{P}^{-1}_{n|n} \widehat{\boldsymbol{x}}_{n|n} \approx \sum_{k=1}^{K} \gamma_k \left(\boldsymbol{P}^{\mathrm{loc}}_{n,k|n}\right)^{-1} \boldsymbol{\psi}_{n,k}, \tag{1.43}$$

$$\boldsymbol{P}^{-1}_{n|n} \approx \sum_{k=1}^{K} \gamma_k \left(\boldsymbol{P}^{\mathrm{loc}}_{n,k|n}\right)^{-1}. \tag{1.44}$$

These relations replace (1.35) and (1.36), where the graph structure is represented

through the scalars γ_k.

Remark (Covariance-intersection method). Expressions (1.43) and (1.44) have a form similar to another commonly used fusion technique that relies on the use of the covariance-intersection (CI) method [33, 34] and which avoids the need for error cross-covariance matrices. In the CI method, the global variables $\{\widehat{\boldsymbol{x}}_{n|n}, \boldsymbol{P}_{n|n}\}$ are estimated according to (1.43) and (1.44) where the scalars γ_k are instead treated as *design* parameters. Specifically, they are chosen as nonnegative convex combination coefficients satisfying $\sum_{k=1}^{K} \gamma_k = 1, \gamma_k \geq 0$, and their values are selected to optimize the resulting $\boldsymbol{P}_{n|n}$, say, by minimizing the trace or determinant of $\boldsymbol{P}_{n|n}$, i.e.,

$$\min_{\{\gamma_k\}} \operatorname{Tr}(\boldsymbol{P}_{n|n}) \quad \text{or} \quad \min_{\{\gamma_k\}} \det(\boldsymbol{P}_{n|n}). \tag{1.45}$$

1.2.7 DIFFUSION COOPERATION

Continuing with (1.43) and (1.44), one difficulty with these expressions is that computation of the global estimator $\widehat{\boldsymbol{x}}_{n|n}$ requires access to the local estimators $\{\boldsymbol{\psi}_{n,\ell}\}$ from across the entire network. We can obtain a decentralized solution that relies solely on local interactions as follows. Substituting (1.44) into (1.43) gives

$$\widehat{\boldsymbol{x}}_{n|n} \approx \left[\sum_{k=1}^{K} \gamma_k \left(\boldsymbol{P}^{\text{loc}}_{n,k|n}\right)^{-1}\right]^{-1} \sum_{k=1}^{K} \gamma_k \left(\boldsymbol{P}^{\text{loc}}_{n,k|n}\right)^{-1} \boldsymbol{\psi}_{n,k}, \tag{1.46}$$

which has the form of a convex weighted average, namely,

$$\widehat{\boldsymbol{x}}_{n|n} \approx \sum_{k=1}^{K} \boldsymbol{\Gamma}_k \boldsymbol{\psi}_{n,k} \tag{1.47}$$

with nonnegative-definite coefficient matrices $\boldsymbol{\Gamma}_k$ that add up to the identity matrix, i.e, $\sum_{k=1}^{K} \boldsymbol{\Gamma}_k = \boldsymbol{I}_N$. The result (1.47) suggests one useful approximation by which the local estimators $\{\boldsymbol{\psi}_{n,\ell}\}$ can be fused within the neighborhood of every agent k to obtain a local version for $\widehat{\boldsymbol{x}}_{n|n}$, which we shall denote by $\widehat{\boldsymbol{x}}_{n,k|n}$ (we are removing the "loc" superscript). This local version is obtained by limiting the convex combination operation (1.47) to the neighborhood of each agent, and by replacing the matrix combination weights $\boldsymbol{\Gamma}_\ell$ by convex combination scalars $a_{\ell k}$. In this way, the computation (1.47) is replaced locally by

$$\widehat{\boldsymbol{x}}_{n,k|n} \leftarrow \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\psi}_{n,\ell}\,. \tag{1.48}$$

Comparing with the incremental listing (1.14), we see that the above calculation amounts to replacing the assignment $\widehat{\boldsymbol{x}}^{\text{loc}}_{n,k|n} \leftarrow \boldsymbol{\psi}_{n,k}$ (marked by $(\star)$ in (1.14)) by a local convex combination step. In view of this substitution, the error covariances of the local estimators $\widehat{\boldsymbol{x}}_{n,k|n}$ computed by (1.48) are not given anymore by the matrices $\boldsymbol{P}_{n,k|n}$ in the listings (1.49) and (1.52) below (where again we removed the superscript "loc"). In summary, we arrive at the listings (1.49) and (1.52) for the diffusion

Kalman filter in two equivalent forms: the measurement and time-update form and the information form.

More generally, and in view of (1.47), a more enhanced fusion of the local estimators $\{\boldsymbol{\psi}_{n,\ell}\}$ is possible by employing convex combination matrices in (1.48) rather than scalars; for example, these combination matrices could be defined in terms of the inverses $\boldsymbol{P}_{n,\ell|n}^{-1}$ as suggested by (1.46). This construction, however, would entail added complexity and would require sharing of additional information regarding these inverses. The implementation (1.49) from [12] employs scalar combination coefficients $\{a_{\ell k}\}$ in order to reduce the complexity of the resulting algorithm. Reference [35] studies the alternative fusion of the estimators $\{\boldsymbol{\psi}_{n,\ell}\}$ in the diffusion Kalman filter by exploiting information about the inverses $\boldsymbol{P}_{n,\ell|n}^{-1}$ and using covariance-intersection combinations over neighborhoods in a manner similar to (1.45).

Diffusion Kalman filter (measurement and time-update form)

start with $\widehat{\boldsymbol{x}}_{0,k|-1} = \boldsymbol{0}$, $\boldsymbol{P}_{0,k|-1} = \boldsymbol{\Pi}_0$.

repeat at every agent k for $n \geq 0$:

(measurement-updates)

set auxiliary variables:

$$\boldsymbol{\psi}_{n,k}^{(0)} \leftarrow \widehat{\boldsymbol{x}}_{n,k|n-1} \quad (N \times 1 \text{ vector})$$

$$\boldsymbol{P}_{n,k}^{(0)} \leftarrow \boldsymbol{P}_{n,k|n-1} \quad (N \times N \text{ matrix})$$

for each neighbor $\ell = 1, 2, \ldots, d_k$ do:

$$\begin{aligned}
&\boldsymbol{e}_{n,k_\ell} = \boldsymbol{z}_{n,k_\ell} - \boldsymbol{H}_{n,k_\ell} \boldsymbol{\psi}_{n,k}^{(\ell-1)} \\
&\boldsymbol{R}_{e,n,k_\ell} = \boldsymbol{R}_{n,k_\ell} + \boldsymbol{H}_{n,k_\ell} \boldsymbol{P}_{n,k}^{(\ell-1)} \boldsymbol{H}_{n,k_\ell}^{\mathrm{T}} \\
&\boldsymbol{\psi}_{n,k}^{(\ell)} = \boldsymbol{\psi}_{n,k}^{(\ell-1)} + \boldsymbol{P}_{n,k}^{(\ell-1)} \boldsymbol{H}_{n,k_\ell}^{\mathrm{T}} \boldsymbol{R}_{e,n,k_\ell}^{-1} \boldsymbol{e}_{n,k_\ell} \\
&\boldsymbol{P}_{n,k}^{(\ell)} = \boldsymbol{P}_{n,k}^{(\ell-1)} - \boldsymbol{P}_{n,k}^{(\ell-1)} \boldsymbol{H}_{n,k_\ell}^{\mathrm{T}} \boldsymbol{R}_{e,n,k_\ell}^{-1} \boldsymbol{H}_{n,k_\ell} \boldsymbol{P}_{n,k}^{(\ell-1)}
\end{aligned} \tag{1.49}$$

end

$$\boldsymbol{P}_{n,k|n} \leftarrow \boldsymbol{P}_{n,k}^{(d_k)}$$

$$\boldsymbol{\psi}_{n,k} \leftarrow \boldsymbol{\psi}_{n,k}^{(d_k)}$$

(combination step)

$$\widehat{\boldsymbol{x}}_{n,k|n} = \sum_{\ell \in \mathfrak{N}_k} a_{\ell k} \boldsymbol{\psi}_{\ell,n}$$

(time-update)

$$\widehat{\boldsymbol{x}}_{n+1,k|n} = \boldsymbol{F}_n \widehat{\boldsymbol{x}}_{n,k|n}$$

$$\boldsymbol{P}_{n+1,k|n} = \boldsymbol{F}_n \boldsymbol{P}_{k,n|n} \boldsymbol{F}_n^{\mathrm{T}} + \boldsymbol{G}_n \boldsymbol{Q}_n \boldsymbol{G}_n^{\mathrm{T}}$$

end

We also note that the combination step in (1.52) is similar to the update used in the consensus Kalman filter derived in [5] with one main difference. The consensus Kalman filter employs a specific combination construction at the local level of the

form:

$$\widehat{\boldsymbol{x}}_{n,k|n} = (1 + (1-d_k)\epsilon)\,\boldsymbol{\psi}_{n,k} + \sum_{\ell\in\mathfrak{N}_k\backslash\{k\}} \epsilon\,\boldsymbol{\psi}_{n,\ell}, \tag{1.50}$$

where we recall that $d_k = |\mathfrak{N}_k|$, and $\epsilon > 0$ is a small weight that is *equal* for all neighbors. In comparison, the diffusion implementation (1.52), which can be written as

$$\widehat{\boldsymbol{x}}_{n,k|n} = a_{kk}\,\boldsymbol{\psi}_{n,k} + \sum_{\ell\in\mathfrak{N}_k\backslash\{k\}} a_{\ell k}\,\boldsymbol{\psi}_{n,\ell}, \tag{1.51}$$

allows more freedom in assigning generally different weights $a_{\ell k}$ to different neighbors [10, 12].

Diffusion Kalman filter (information form)

start with $\widehat{\boldsymbol{x}}_{0,k|-1} = \boldsymbol{0}, \left(\boldsymbol{P}_{0,k|-1}\right)^{-1} = \boldsymbol{\Pi}_0^{-1}$.

repeat at every agent k for $n \geq 0$:

$$\begin{aligned}
\boldsymbol{S}_{n,k} &= \sum_{\ell\in\mathfrak{N}_k} \boldsymbol{H}_{n,\ell}^{\mathrm{T}} \boldsymbol{R}_{n,\ell}^{-1} \boldsymbol{H}_{n,\ell} \\
\boldsymbol{q}_{n,k} &= \sum_{\ell\in\mathfrak{N}_k} \boldsymbol{H}_{n,\ell}^{\mathrm{T}} \boldsymbol{R}_{n,\ell}^{-1} \boldsymbol{z}_{n,\ell} \\
(\boldsymbol{P}_{n,k|n})^{-1} &= (\boldsymbol{P}_{n,k|n-1})^{-1} + \boldsymbol{S}_{n,k} \\
\boldsymbol{\psi}_{n,k} &= \widehat{\boldsymbol{x}}_{n,k|n-1} + \boldsymbol{P}_{n,k|n}(\boldsymbol{q}_{n,k} - \boldsymbol{S}_{n,k}\widehat{\boldsymbol{x}}_{n,k|n-1}) \\
\widehat{\boldsymbol{x}}_{n,k|n} &= \sum_{\ell\in\mathfrak{N}_k} a_{\ell k}\boldsymbol{\psi}_{n,\ell} \\
\widehat{\boldsymbol{x}}_{n+1,k|n} &= \boldsymbol{F}_n \widehat{\boldsymbol{x}}_{n,k|n} \\
\boldsymbol{P}_{n+1,k|n} &= \boldsymbol{F}_n \boldsymbol{P}_{n,k|n} \boldsymbol{F}_n^{\mathrm{T}} + \boldsymbol{G}_n \boldsymbol{Q}_n \boldsymbol{G}_n^{\mathrm{T}}
\end{aligned} \tag{1.52}$$

end

1.2.8 PERFORMANCE ANALYSIS

In view of the approximations (1.44) and (1.48), it is important to examine the mean-square error performance of the resulting diffusion Kalman filter. In this section, we examine how close the local estimates $\{\widehat{\boldsymbol{x}}_{n,k|n}\}$ get to the state variable $\boldsymbol{x}_n$ by evaluating the size of the mean-square error in steady-state after the filter has had sufficient time to learn.

We start by collecting all state estimation errors from across the network into an extended $K \times 1$ block vector (whose individual entries are of size $N \times 1$ each):

$$\widetilde{\boldsymbol{\mathcal{X}}}_{n|n} \triangleq \begin{pmatrix} \widetilde{\boldsymbol{x}}_{1,n|n} \\ \widetilde{\boldsymbol{x}}_{n,2|n} \\ \vdots \\ \widetilde{\boldsymbol{x}}_{n,K|n} \end{pmatrix}, \quad \text{where } \widetilde{\boldsymbol{x}}_{n,k|n} \triangleq \boldsymbol{x}_n - \widehat{\boldsymbol{x}}_{n,k|n}, \tag{1.53}$$

and introduce supporting block-diagonal and Kronecker product matrices:

$$\boldsymbol{\mathcal{U}}_{n-1} \triangleq \mathbb{1}_K \otimes \boldsymbol{u}_{n-1}, \tag{1.54}$$

$$\boldsymbol{\mathcal{V}}_n \triangleq \text{blkcol}\{\boldsymbol{v}_{n,1|n}, \boldsymbol{v}_{2,n|n}, \ldots, \boldsymbol{v}_{n,K|n}\}, \tag{1.55}$$

$$\boldsymbol{\mathcal{P}}_{n|n} \triangleq \text{blkdiag}\{\boldsymbol{P}_{n,1|n}, \boldsymbol{P}_{n,2|n}, \ldots, \boldsymbol{P}_{n,K|n}\}, \tag{1.56}$$

$$\mathcal{H}_n \triangleq \text{blkdiag}\{\boldsymbol{H}_{n,1}, \boldsymbol{H}_{n,2}, \ldots, \boldsymbol{H}_{n,K}\}, \tag{1.57}$$

$$\mathcal{S}_n \triangleq \text{blkdiag}\{\boldsymbol{S}_{n,1}, \boldsymbol{S}_{n,2}, \ldots, \boldsymbol{S}_{n,K}\}, \tag{1.58}$$

$$\mathcal{R}_n \triangleq \text{blkdiag}\{\boldsymbol{R}_{n,1}, \boldsymbol{R}_{n,2}, \ldots, \boldsymbol{R}_{n,K}\}, \tag{1.59}$$

$$\mathcal{J} \triangleq \boldsymbol{J} \otimes \boldsymbol{I}_N, \tag{1.60}$$

$$\mathcal{A} \triangleq \boldsymbol{A} \otimes \boldsymbol{I}_N, \tag{1.61}$$

where $\otimes$ denotes the Kronecker product operation. Subtracting $\boldsymbol{x}_n$ from both sides of the expression for $\boldsymbol{\psi}_{n,k}$ in (1.52), and using $z_{n,\ell} = \boldsymbol{H}_{n,\ell}\boldsymbol{x}_n + \boldsymbol{v}_{n,\ell}$, gives

$$\begin{aligned}
\widetilde{\boldsymbol{\psi}}_{n,k} &= \widetilde{\boldsymbol{x}}_{n,k|n-1} - \boldsymbol{P}_{n,k|n}(\boldsymbol{q}_{n,k} - \boldsymbol{S}_{n,k}\widehat{\boldsymbol{x}}_{n,k|n-1}) \\
&= \widetilde{\boldsymbol{x}}_{n,k|n-1} - \boldsymbol{P}_{n,k|n}\sum_{\ell\in\mathfrak{N}_k} \boldsymbol{H}_{n,\ell}^{\mathrm{T}}\boldsymbol{R}_{n,\ell}^{-1}(z_{n,\ell} - \boldsymbol{H}_{n,\ell}\widehat{\boldsymbol{x}}_{n,k|n-1}) \\
&= (\boldsymbol{I}_N - \boldsymbol{P}_{n,k|n}\boldsymbol{S}_{n,k})\widetilde{\boldsymbol{x}}_{n,k|n-1} - \boldsymbol{P}_{n,k|n}\sum \boldsymbol{H}_{n,\ell}^{\mathrm{T}}\boldsymbol{R}_{n,\ell}^{-1}\boldsymbol{v}_{n,\ell},
\end{aligned} \tag{1.62}$$

where $\widetilde{\boldsymbol{\psi}}_{n,k} \triangleq \boldsymbol{x}_n - \boldsymbol{\psi}_{n,k}$. Using the combination step from (1.52) gives

$$\widetilde{\boldsymbol{x}}_{n,k|n} = \sum_{\ell\in\mathfrak{N}_k} a_{\ell k}\left((\boldsymbol{I}_N - \boldsymbol{P}_{n,\ell|n}\boldsymbol{S}_{n,\ell})\widetilde{\boldsymbol{x}}_{n,\ell|n-1} - \boldsymbol{P}_{n,\ell|n}\sum_{i\in\mathfrak{N}_\ell}\boldsymbol{H}_{n,i}^{\mathrm{T}}\boldsymbol{R}_{n,i}^{-1}\boldsymbol{v}_{n,i}\right). \tag{1.63}$$

Using $\widetilde{\boldsymbol{x}}_{n,\ell|n-1} = \boldsymbol{F}_{n-1}\widetilde{\boldsymbol{x}}_{n-1,\ell|n-1} + \boldsymbol{G}_{n-1}\boldsymbol{u}_{n-1}$, we arrive at the recursion

$$\begin{aligned}
\widetilde{\boldsymbol{x}}_{n,k|n} = \sum_{\ell\in\mathfrak{N}_k} a_{\ell k}\Bigg((\boldsymbol{I}_N - \boldsymbol{P}_{n,\ell|n}\boldsymbol{S}_{n,\ell})\boldsymbol{F}_{n-1}\widetilde{\boldsymbol{x}}_{n-1,\ell|n-1} \\
+ (\boldsymbol{I}_N - \boldsymbol{P}_{n,\ell|n}\boldsymbol{S}_{n,\ell})\boldsymbol{G}_{n-1}\boldsymbol{u}_{n-1} - \boldsymbol{P}_{n,\ell|n}\sum_{i\in\mathfrak{N}_\ell}\boldsymbol{H}_{n,i}^{\mathrm{T}}\boldsymbol{R}_{n,i}^{-1}\boldsymbol{v}_{n,i}\Bigg).
\end{aligned} \tag{1.64}$$

This relation shows that the network error vector evolves according to the following dynamics:

$$\begin{aligned}
\widetilde{\boldsymbol{\mathcal{X}}}_{n|n} &= \mathcal{A}^{\mathrm{T}}(\boldsymbol{I}_N - \boldsymbol{\mathcal{P}}_{n|n}\mathcal{S}_n)(\boldsymbol{I}_K \otimes \boldsymbol{F}_{n-1})\widetilde{\boldsymbol{\mathcal{X}}}_{n-1|n-1} \\
&\quad + \mathcal{A}^{\mathrm{T}}(\boldsymbol{I}_{KN} - \boldsymbol{\mathcal{P}}_{n|n}\mathcal{S}_n)(\boldsymbol{I}_K \otimes \boldsymbol{G}_{n-1})\boldsymbol{\mathcal{U}}_{n-1} - \mathcal{A}^{\mathrm{T}}\boldsymbol{\mathcal{P}}_{n|n}\mathcal{J}^{\mathrm{T}}\mathcal{H}_n^{\mathrm{T}}\mathcal{R}_n^{-1}\boldsymbol{\mathcal{V}}_n,
\end{aligned} \tag{1.65}$$

which we rewrite more compactly as

$$\widetilde{\boldsymbol{\mathcal{X}}}_{n|n} = \mathcal{F}_n\widetilde{\boldsymbol{\mathcal{X}}}_{n-1|n-1} + \mathcal{G}_n\boldsymbol{\mathcal{U}}_{n-1} - \mathcal{D}_n\boldsymbol{\mathcal{V}}_n, \tag{1.66}$$

where

$$\mathcal{F}_n \triangleq \mathcal{A}^{\mathrm{T}}(\boldsymbol{I}_{KN} - \mathcal{P}_{n|n}\mathcal{S}_n)(\boldsymbol{I}_K \otimes \boldsymbol{F}_{n-1}), \tag{1.67}$$

$$\mathcal{G}_n \triangleq \mathcal{A}^{\mathrm{T}}(\boldsymbol{I}_{KN} - \mathcal{P}_{n|n}\mathcal{S}_n)(\boldsymbol{I}_K \otimes \boldsymbol{G}_{n-1}), \tag{1.68}$$

$$\mathcal{D}_n \triangleq \mathcal{A}^{\mathrm{T}}\mathcal{P}_{n|n}\mathcal{J}^{\mathrm{T}}\mathcal{H}_n^{\mathrm{T}}\mathcal{R}_n^{-1}. \tag{1.69}$$

If we now let $\mathcal{P}_{\widetilde{\boldsymbol{\mathcal{X}}},n}$ denote the covariance matrix of the network error vector,

$$\mathcal{P}_{\widetilde{\boldsymbol{\mathcal{X}}},n} \triangleq \mathbb{E}\left[\widetilde{\boldsymbol{\mathcal{X}}}_{n|n}\widetilde{\boldsymbol{\mathcal{X}}}_{n|n}^{\mathrm{T}}\right], \tag{1.70}$$

it then follows from (1.66) that this matrix satisfies the Lyapunov recursion

$$\mathcal{P}_{\widetilde{\boldsymbol{\mathcal{X}}},n} = \mathcal{F}_n\mathcal{P}_{\widetilde{\boldsymbol{\mathcal{X}}},n-1}\mathcal{F}_n^{\mathrm{T}} + \mathcal{G}_n(\mathbb{1}_K\mathbb{1}_K^{\mathrm{T}} \otimes \boldsymbol{Q}_{n-1})\mathcal{G}_n^{\mathrm{T}} + \mathcal{D}_n\mathcal{R}_n\mathcal{D}_n^{\mathrm{T}}. \tag{1.71}$$

In order to analyze the stability and performance of the diffusion filter, we shall consider here the following conditions – see [28] for definitions of the notions of stabilizability and detectability.

Assumption. *It is assumed that the model parameters* $\{\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}_k, \boldsymbol{Q}, \boldsymbol{R}_k\}$ *do not depend on the time index, n. We further collect the measurement matrices in the neighborhood of agent k into the block column matrix:*

$$\boldsymbol{H}_k^{\mathrm{loc}} \triangleq \mathrm{blkcol}\{\boldsymbol{H}_{k_1}, \boldsymbol{H}_{k_2}, \ldots, \boldsymbol{H}_{k_{d_k}}\}, \tag{1.72}$$

and assume that the pair $(\boldsymbol{F}, \boldsymbol{G}\boldsymbol{Q}^{1/2})$ *is stabilizable and the pair* $(\boldsymbol{F}, \boldsymbol{H}_k^{\mathrm{loc}})$ *is detectable.* □

Under the stabilizability and detectability conditions, it is known from the convergence properties of the discrete Riccati recursions that each entry $\boldsymbol{P}_{n,k|n}$ of $\mathcal{P}_{n|n}$ converges to the quantity $\boldsymbol{P}_k^+$ defined by [28]:

$$(\boldsymbol{P}_k^+)^{-1} = (\boldsymbol{P}_k)^{-1} + (\boldsymbol{H}_k^{\mathrm{loc}})^{\mathrm{T}}(\boldsymbol{R}_k^{\mathrm{loc}})^{-1}\boldsymbol{H}_k^{\mathrm{loc}} = (\boldsymbol{P}_k)^{-1} + \boldsymbol{S}_k, \tag{1.73}$$

where

$$\boldsymbol{R}_k^{\mathrm{loc}} \triangleq \mathrm{blkdiag}\{\boldsymbol{R}_{k_1}, \boldsymbol{R}_{k_2}, \ldots, \boldsymbol{R}_{k_{d_k}}\}, \tag{1.74}$$

and $\boldsymbol{P}_k$ is the unique stabilizing solution of the following discrete algebraic Riccati equation (DARE):

$$\boldsymbol{P}_k = \boldsymbol{F}\boldsymbol{P}_k^+\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^{\mathrm{T}} = \boldsymbol{F}\boldsymbol{P}_k\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^{\mathrm{T}} - \boldsymbol{K}_{p,k}\boldsymbol{R}_{e,k}\boldsymbol{K}_{p,k}^{\mathrm{T}}, \tag{1.75}$$

where

$$\boldsymbol{K}_{p,k} \triangleq \boldsymbol{F}\boldsymbol{P}_k(\boldsymbol{H}_k^{\mathrm{loc}})^{\mathrm{T}}\boldsymbol{R}_{e,k}^{-1}, \qquad \boldsymbol{R}_{e,k} \triangleq \boldsymbol{R}_k^{\mathrm{loc}} + \boldsymbol{H}_k^{\mathrm{loc}}\boldsymbol{P}_k(\boldsymbol{H}_k^{\mathrm{loc}})^{\mathrm{T}}. \tag{1.76}$$

Accordingly, the quantities $\{\mathcal{P}_{n|n}, \mathcal{F}_n, \mathcal{G}_n, \mathcal{D}_n\}$ converge to steady-state values denoted

by [12]:

$$\mathcal{P}^{+} \triangleq \lim_{n\to\infty} \mathcal{P}_{n|n} \ \text{(block diagonal)}, \tag{1.77}$$

$$\mathcal{F} \triangleq \lim_{n\to\infty} \mathcal{F}_n = \mathcal{A}^{\mathrm{T}}(\boldsymbol{I}_{KN} - \mathcal{P}^{+}\mathcal{S})(\boldsymbol{I}_K \otimes \boldsymbol{F}), \tag{1.78}$$

$$\mathcal{G} \triangleq \lim_{n\to\infty} \mathcal{G}_n = \mathcal{A}^{\mathrm{T}}(\boldsymbol{I}_{KN} - \mathcal{P}^{+}\mathcal{S})(\boldsymbol{I}_K \otimes \boldsymbol{G}), \tag{1.79}$$

$$\mathcal{D} \triangleq \lim_{n\to\infty} \mathcal{D}_n = \mathcal{A}^{\mathrm{T}}\mathcal{P}^{+}\mathcal{J}^{\mathrm{T}}\mathcal{H}^{\mathrm{T}}\mathcal{R}^{-1}, \tag{1.80}$$

where $\mathcal{S}$, $\mathcal{H}$, and $\mathcal{R}$ are written without the time subscript n because their entries are now time-independent. Furthermore, the stabilizability and detectability conditions ensure that (see Lemma 1 in [12]):

$$\mathcal{X} \triangleq (\boldsymbol{I}_{KN} - \mathcal{P}^{+}\mathcal{S})(\boldsymbol{I}_K \otimes \boldsymbol{F}) \ \text{ is block diagonal and stable.} \tag{1.81}$$

Note that the limiting matrix $\mathcal{F}$ in (1.78) has the form

$$\mathcal{F} = \mathcal{A}^{\mathrm{T}}\mathcal{X}. \tag{1.82}$$

We will verify soon that, under some conditions, $\mathcal{F}$ is also stable for any left-stochastic matrix $\mathcal{A}$ and, in view of Lemma 1 in the Appendix (Section 1.5), the error covariance matrix $\mathcal{P}_{\widetilde{\mathcal{X}},n}$ in (1.71) converges to the unique solution of the Lyapunov equation:

$$\mathcal{P}_{\widetilde{\mathcal{X}}} = \mathcal{F}\mathcal{P}_{\widetilde{\mathcal{X}}}\mathcal{F}^{\mathrm{T}} + \mathcal{G}(\mathbb{1}_K\mathbb{1}_K^{\mathrm{T}} \otimes \boldsymbol{Q})\mathcal{G}^{\mathrm{T}} + \mathcal{D}\mathcal{R}\mathcal{D}^{\mathrm{T}}. \tag{1.83}$$

Using the vec notation (which stacks the columns of a matrix on top of each other), we can solve for $\mathcal{P}_{\widetilde{\mathcal{X}}}$ and write

$$\mathrm{vec}(\mathcal{P}_{\widetilde{\mathcal{X}}}) = (\boldsymbol{I}_{(KN)^2} - \mathcal{F}^{\mathrm{T}} \otimes \mathcal{F})^{-1}\,\mathrm{vec}(\mathcal{G}(\mathbb{1}_K\mathbb{1}_K^{\mathrm{T}} \otimes \boldsymbol{Q})\mathcal{G}^{\mathrm{T}} + \mathcal{D}\mathcal{R}\mathcal{D}^{\mathrm{T}}). \tag{1.84}$$

In this way, the steady-state mean-square error at any agent k will be given by

$$\lim_{n\to\infty} \mathbb{E}\left[\|\widetilde{\boldsymbol{x}}_{n,k|n}\|^2\right] = \mathrm{Tr}(\mathcal{P}_{\widetilde{\mathcal{X}}}\mathcal{I}_k), \tag{1.85}$$

where $\mathcal{I}_k$ is an $K \times K$ block diagonal matrix with blocks of size $N \times N$; it contains the identity matrix at block (k, k) and zeros everywhere else. That is, it holds that $\mathcal{I}_k = \boldsymbol{e}_k\boldsymbol{e}_k^{\mathrm{T}} \otimes \boldsymbol{I}_N$, where $\boldsymbol{e}_k$ is the basis column vector of size $K \times 1$ with unit entry at location k and zeros everywhere else. Consequently, the average mean-square error across the network is

$$\mathrm{MSE}_{\mathrm{av}} = \frac{1}{K}\mathrm{Tr}(\mathcal{P}_{\widetilde{\mathcal{X}}}). \tag{1.86}$$

We verify the above claims by showing that the matrix $\mathcal{F}$ is stable and that, in view of this fact, the Lyapunov recursion (1.71) converges to the unique solution of the Lyapunov equation (1.83). These facts follow from the results of two lemmas presented in the Appendix.

1.3 DISTRIBUTED PARTICLE FILTERING

In the second part of this chapter, we discuss distributed particle filtering. We will focus on consensus-based methods in Subsection 1.3.4 and on diffusion-based methods in Subsection 1.3.5. Before that, we present the underlying state-space model, which generalizes (1.2) and (1.3), and we discuss some basics of sequential Bayesian estimation and review the particle filter. We adopt a Bayesian framework for estimation, which means that both the state $\boldsymbol{x}_n$ and the measurements $\boldsymbol{z}_{n,k}$ are modeled as random vectors [36].

1.3.1 STATE-SPACE MODEL

Differently from Section 1.2, the state-space model is now allowed to be nonlinear. The temporal dynamics or evolution of the state $\boldsymbol{x}_n$ is characterized by the state-evolution model (a.k.a. state-transition model or system model)

$$\boldsymbol{x}_{n+1} = \mathbf{g}_n(\boldsymbol{x}_n, \boldsymbol{u}_n), \quad n = 0, 1, \ldots, \tag{1.87}$$

which extends (1.2). In (1.87), $\mathbf{g}_n(\cdot, \cdot)$ is a known, generally nonlinear function, and $\boldsymbol{u}_n$ is state noise that is statistically independent and identically distributed (iid) across time n and also statistically independent of the state sequence. The probability density function (pdf) of $\boldsymbol{u}_n$ is assumed known. Equation (1.87) then determines the *state-transition pdf* $f(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$ for $n = 1, 2, \ldots$.

Furthermore, the statistical dependence of the measurement $\boldsymbol{z}_{n,k}$ of agent $k \in \{1, 2, \ldots, K\}$ on the state $\boldsymbol{x}_n$ is characterized by a measurement model

$$\boldsymbol{z}_{n,k} = \mathbf{h}_{n,k}(\boldsymbol{x}_n, \boldsymbol{v}_{n,k}), \quad n = 1, 2, \ldots; \;\; k = 1, 2, \ldots, K, \tag{1.88}$$

which extends (1.3). In (1.88), $\mathbf{h}_{n,k}(\cdot, \cdot)$ is a known, generally nonlinear function, and $\boldsymbol{v}_{n,k}$ is observation noise that is statistically independent across time n and across the agents k, and also statistically independent of the state sequence and of the state noise. The pdf of $\boldsymbol{v}_{n,k}$ is assumed known. Equation (1.88) then determines the *local likelihood function* of agent k, $f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n)$, for $n = 1, 2, \ldots$. Because $\boldsymbol{v}_{n,k}$ is assumed independent of $\boldsymbol{v}_{n,k'}$ for $k' \neq k$, the measurements $\boldsymbol{z}_{n,k}$ at different agents k are conditionally independent given the state $\boldsymbol{x}_n$. Thus, the *global (all-agents) likelihood function* $f(\boldsymbol{z}_n|\boldsymbol{x}_n)$, with $\boldsymbol{z}_n \stackrel{\Delta}{=} (\boldsymbol{z}_{n,1}^{\mathrm{T}} \; \boldsymbol{z}_{n,2}^{\mathrm{T}} \cdots \boldsymbol{z}_{n,K}^{\mathrm{T}})^{\mathrm{T}}$, factorizes into the local likelihood functions, i.e.,

$$f(\boldsymbol{z}_n|\boldsymbol{x}_n) = \prod_{k=1}^{K} f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n). \tag{1.89}$$

The above independence assumptions can be relaxed; for example, the state noise process $\boldsymbol{u}_n$ and the observation noise processes $\boldsymbol{v}_{n,k}$ may be allowed to be dependent [37]. Further, the conditional independence of the observations can be removed too [38, 39]. Also, the noises in the state and observation equations do not have to be

independent across time [40, 41]. However, the independence assumptions stated above are required for the consensus-based DPF methods presented below. We also note that the case where the state-evolution function $\mathbf{g}_n(\boldsymbol{x}_n, \boldsymbol{u}_n)$ is linear in both $\boldsymbol{x}_n$ and $\boldsymbol{u}_n$ and the measurement function $\mathbf{h}_{n,k}(\boldsymbol{x}_n, \boldsymbol{v}_{n,k})$ is linear in both $\boldsymbol{x}_n$ and $\boldsymbol{v}_n$ was considered in Section 1.2 (see (1.2) and (1.3)). Furthermore, when the state noise $\boldsymbol{u}_n$ in (1.87) and the observation noise $\boldsymbol{v}_n$ in (1.88) have Gaussian distributions, these distributions are completely characterized by their first- and second-order moments. Finally, we point out that the network may be dynamic in that the set of neighboring nodes may vary with time (i.e., we may have $\mathfrak{N}_{n,k}$ instead of $\mathfrak{N}_k$). However, in the rest of the chapter, we will continue to consider the network to be static.

The above state-evolution and measurement models along with the associated statistical assumptions imply two further conditional independence properties. First, the current state $\boldsymbol{x}_n$ is conditionally independent of all the past measurements, $\boldsymbol{z}_{1:n-1,k} \stackrel{\Delta}{=} (\boldsymbol{z}_{1,k}^{\mathrm{T}}\ \boldsymbol{z}_{2,k}^{\mathrm{T}} \cdots \boldsymbol{z}_{n-1,k}^{\mathrm{T}})^{\mathrm{T}}$, given the previous state $\boldsymbol{x}_{n-1}$, i.e.,

$$f(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}, \boldsymbol{z}_{1:n-1,k}) = f(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}), \tag{1.90}$$

for all $k = 1, 2, \ldots, K$. Second, the current measurements $\boldsymbol{z}_{n,k}$ are conditionally independent of all the past measurements, $\boldsymbol{z}_{1:n-1,k'}$, given the current state $\boldsymbol{x}_n$, i.e.,

$$f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n, \boldsymbol{z}_{1:n-1,k'}) = f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n), \tag{1.91}$$

for any $k' \in \{1, 2, \ldots, K\}$, including in particular $k' = k$.

1.3.2 SEQUENTIAL BAYESIAN ESTIMATION

We address the problem of sequential estimation of the state $\boldsymbol{x}_n$ from the total (all-agents) measurement sequence $\boldsymbol{z}_{1:n} \stackrel{\Delta}{=} (\boldsymbol{z}_1^{\mathrm{T}}\ \boldsymbol{z}_2^{\mathrm{T}} \cdots \boldsymbol{z}_n^{\mathrm{T}})^{\mathrm{T}}$. In the Bayesian context, this essentially amounts to calculating the *posterior pdf* $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$, from which optimal Bayesian estimators and quantities characterizing their performance can be derived. In particular, the MMSE estimator [36] is given by the first moment of $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$, i.e.,

$$\widehat{\boldsymbol{x}}_n^{\mathrm{MMSE}} = \mathbb{E}\,[\boldsymbol{x}_n|\boldsymbol{z}_{1:n}] = \int_{\mathbb{R}^N} \boldsymbol{x}_n f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})\,\mathrm{d}\boldsymbol{x}_n, \quad n = 1, 2, \ldots. \tag{1.92}$$

In a centralized scenario, based on the conditional independence properties (1.90) and (1.91), the current posterior pdf $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$ can be calculated sequentially (recursively) from the previous posterior pdf $f(\boldsymbol{x}_{n-1}|\boldsymbol{z}_{1:n-1})$ and the current all-agents measurement vector $\boldsymbol{z}_n$. This recursion consists of a prediction step calculating $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1})$ from $f(\boldsymbol{x}_{n-1}|\boldsymbol{z}_{1:n-1})$ and a subsequent time-update or correction step calculating $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$ from $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1})$. The prediction step involves the state-transition pdf $f(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$, and the update step involves the global likelihood function $f(\boldsymbol{z}_n|\boldsymbol{x}_n)$ and, thus, the all-agents measurement vector $\boldsymbol{z}_n$. Unfortunately, the prediction and update steps as well as the calculation of the MMSE estimate in (1.92) are usually computationally infeasible. A prominent exception is the case of the linear-Gaussian

state-space model, where optimal sequential Bayesian state estimation reduces to the Kalman filter considered in Subsection 1.2.3 [19, 28, 29].

Several computationally feasible approximations to optimal sequential Bayesian state estimation have been proposed in the centralized case; these include the extended Kalman filter [19, 42, 36], the Gaussian sum filter [43], the unscented (sigma-point) Kalman filter [44, 45], the cubature Kalman filter [46], and the particle filter [47, 20, 48, 49]. In particular, particle filters are well suited to any nonlinear state-space model and any distributions of the state and observation noises, and they perform well in situations where Kalman filter-based methods perform poorly. This generally comes at the cost of an increased computational complexity. Below, we will discuss distributed methods for particle filtering in decentralized agent networks. We start with a review of the basic particle filter.

1.3.3 REVIEW OF THE PARTICLE FILTER

A particle filter (PF) performs an approximation of optimal sequential Bayesian state estimation that is based on Monte Carlo simulation and importance sampling [47, 20, 48, 49]. In a centralized scenario, the posterior pdf $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$ is represented in an approximative manner by M randomly drawn samples or *particles* $\boldsymbol{x}_n^{(m)}$ and corresponding weights $w_n^{(m)}$, where $m = 1, 2, \ldots, M$. Specifically, at each time step n, propagation of the posterior pdf (i.e., $f(\boldsymbol{x}_{n-1}|\boldsymbol{z}_{1:n-1}) \rightarrow f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$) is replaced by propagation of the particles and weights (i.e., $\{(\boldsymbol{x}_{n-1}^{(m)}, w_{n-1}^{(m)})\}_{m=1}^M \rightarrow \{(\boldsymbol{x}_n^{(m)}, w_n^{(m)})\}_{m=1}^M$). As in the case of optimal sequential Bayesian estimation considered in Subsection 1.3.2, this propagation consists of a prediction step and an update or correction step.

In the prediction step at time n, for each preceding particle $\boldsymbol{x}_{n-1}^{(m)}$, a new particle $\boldsymbol{x}_n^{(m)}$ is sampled from a suitably chosen proposal pdf (a.k.a. importance pdf) $q(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(m)})$. In the simplest case, $q(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(m)})$ is chosen as $f(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(m)})$, i.e., the state-transition pdf $f(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$ conditioned on $\boldsymbol{x}_{n-1} = \boldsymbol{x}_{n-1}^{(m)}$; the resulting PF algorithm is known as the sequential importance resampling (SIR) filter. However, more sophisticated "adapted" proposal pdfs that also involve the current measurement $\boldsymbol{z}_n$ can result in improved estimation performance [49, 50] (see Subsection 1.3.4.7).

In the update step at time n, for each particle $\boldsymbol{x}_n^{(m)}$, a nonnormalized weight is calculated according to

$$\widetilde{w}_n^{(m)} = w_{n-1}^{(m)} \frac{f(\boldsymbol{z}_n|\boldsymbol{x}_n^{(m)}) f(\boldsymbol{x}_n^{(m)}|\boldsymbol{x}_{n-1}^{(m)})}{q(\boldsymbol{x}_n^{(m)}|\boldsymbol{x}_{n-1}^{(m)})}, \quad m = 1, 2, \ldots, M. \tag{1.93}$$

For the SIR filter, this simplifies to

$$\widetilde{w}_n^{(m)} = w_{n-1}^{(m)} f(\boldsymbol{z}_n|\boldsymbol{x}_n^{(m)}), \quad m = 1, 2, \ldots, M. \tag{1.94}$$

Subsequently, the weights are normalized according to $w_n^{(m)} = \widetilde{w}_n^{(m)} / \sum_{m'=1}^M \widetilde{w}_n^{(m')}$. The set of particles and normalized weights $\{(\boldsymbol{x}_n^{(m)}, w_n^{(m)})\}_{m=1}^M$ constitutes a Monte Carlo representation of the posterior pdf $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$. From $\{(\boldsymbol{x}_n^{(m)}, w_n^{(m)})\}_{m=1}^M$, a corresponding

approximation of the MMSE state estimate $\widehat{\boldsymbol{x}}_n^{\mathrm{MMSE}}$ in (1.92) can be computed as the weighted sample mean, i.e.,

$$\widehat{\boldsymbol{x}}_n = \sum_{m=1}^{M} w_n^{(m)} \boldsymbol{x}_n^{(m)}. \tag{1.95}$$

Finally, if a suitable criterion is satisfied (as discussed in [51, 20]), the set $\{(\boldsymbol{x}_n^{(m)}, w_n^{(m)})\}_{m=1}^{M}$ is resampled to avoid an effect known as particle degeneracy. In the simplest case [51], the resampled particles are obtained by sampling with replacement from the set $\{(\boldsymbol{x}_n^{(m)}, w_n^{(m)})\}_{m=1}^{M}$, where $\boldsymbol{x}_n^{(m)}$ is sampled with probability $w_n^{(m)}$. This results in M resampled particles $\boldsymbol{x}_n^{(m)}$. The weights are redefined as $w_n^{(m)} = 1/M$.

This recursive algorithm is initialized at time $n=0$ by M particles $\boldsymbol{x}_0^{(m)}$, $m = 1, 2, \ldots, M$, which are drawn from a suitable prior pdf $f(\boldsymbol{x}_0)$. The initial weights are equal, i.e., $w_0^{(m)} = 1/M$ for all m.

In a distributed implementation based on a decentralized network of agents, each agent runs a local instance of a PF, hereafter briefly referred to as "local PF." The local PF at agent k observes only the local measurements $\boldsymbol{z}_{n,k}$ directly; however, it receives from the neighbor agents indirect information about the measurements of the other agents in the network and also provides to the neighboring agents indirect information about its own measurements. The type of information that is exchanged between neighboring agents depends on the specific method used for distributed particle filtering. A distributed PF (DPF) is based on the PF algorithm summarized above. However, it modifies that algorithm to account for the fact that each agent runs its own local PF, and it employs some networkwide distributed scheme (such as consensus, gossip, or diffusion) to disseminate and fuse local information provided by the agents. Several types of DPF methods have been proposed, including [17, 52, 53, 54, 55, 13, 56, 57, 58, 59, 60, 61, 62, 63, 64]. In what follows, we discuss two classes of DPF methods that rely on consensus and diffusion schemes for distributed information dissemination and fusion.

1.3.4 CONSENSUS-BASED METHODS

In a DPF that emulates the performance of the basic PF, the local PF at agent k attempts to track a particle representation $\{(\boldsymbol{x}_{n,k}^{(m)}, w_{n,k}^{(m)})\}_{m=1}^{M}$ of the global posterior pdf $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$. According to (1.93), this requires the *global* likelihood function (GLF) $f(\boldsymbol{z}_n|\boldsymbol{x}_n)$ (evaluated at the current local particles, i.e., $\boldsymbol{x}_n = \boldsymbol{x}_{n,k}^{(m)}$), rather than merely the *local* likelihood function (LLF) of agent k, $f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n)$. Because agent k by itself is only able to calculate the LLF, calculation of the GLF requires information provided by the other agents.

1.3.4.1 DPF Based on Likelihood Consensus

A recently proposed class of DPF algorithms performs a distributed calculation of the GLF using the *likelihood consensus* (LC) scheme [13, 56]. To develop that scheme,

we first take the logarithm of (1.89), which yields

$$\ln f(\boldsymbol{z}_n|\boldsymbol{x}_n) = \sum_{k=1}^{K} \ln f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n), \tag{1.96}$$

where ln denotes the natural logarithm. We can thus write the GLF as

$$f(\boldsymbol{z}_n|\boldsymbol{x}_n) = \exp(\ln f(\boldsymbol{z}_n|\boldsymbol{x}_n)) = \exp(K\lambda(\boldsymbol{x}_n;\boldsymbol{z}_n)), \tag{1.97}$$

with

$$\lambda(\boldsymbol{x}_n;\boldsymbol{z}_n) \triangleq \frac{1}{K}\ln f(\boldsymbol{z}_n|\boldsymbol{x}_n) = \frac{1}{K}\sum_{k=1}^{K} \ln f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n), \tag{1.98}$$

where (1.96) has been used in the last step. Hence, the GLF $f(\boldsymbol{z}_n|\boldsymbol{x}_n)$ has been rewritten in terms of the the *average* of the K log-LLFs $\ln f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n)$, $k = 1, 2, \ldots, K$. Because the log-LLFs $\ln f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n)$ are functions of the unknown state $\boldsymbol{x}_n \in \mathbb{R}^N$, rather than simply numbers, we cannot directly use a distributed averaging scheme such as the average consensus algorithm to calculate $\lambda(\boldsymbol{x}_n;\boldsymbol{z}_n)$.

The solution provided by the LC scheme is based on an approximate (finite-order) function expansion of the log-LLFs $\ln f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n)$. Let $\{\varphi_r(\boldsymbol{x}_n)\}_{r=1}^{R}$ denote a system of R functions that is known to all agents. Possible choices of these functions include monomials [13], orthogonal polynomials [65], Fourier basis functions [56], and spline functions [66]. We consider the approximations of the log-LLFs given by

$$\ln f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n) \approx \sum_{r=1}^{R} \alpha_{n,k,r}(\boldsymbol{z}_{n,k})\,\varphi_r(\boldsymbol{x}_n), \qquad k = 1, 2, \ldots, K, \tag{1.99}$$

with suitable expansion coefficients $\alpha_{n,k,r}(\boldsymbol{z}_{n,k})$, $r = 1, 2, \ldots, R$. Note that these expansion coefficients involve the local measurements $\boldsymbol{z}_{n,k}$, and thus they are different at different agents k. Inserting (1.99) into (1.98) and changing the order of summation yields

$$\lambda(\boldsymbol{x}_n;\boldsymbol{z}_n) \approx \widetilde{\lambda}(\boldsymbol{x}_n;\boldsymbol{z}_n) \triangleq \sum_{r=1}^{R} a_{n,r}(\boldsymbol{z}_n)\,\varphi_r(\boldsymbol{x}_n), \tag{1.100}$$

where

$$a_{n,r}(\boldsymbol{z}_n) \triangleq \frac{1}{K}\sum_{k=1}^{K} \alpha_{n,k,r}(\boldsymbol{z}_{n,k}). \tag{1.101}$$

Using in (1.97) the approximation $\widetilde{\lambda}(\boldsymbol{x}_n;\boldsymbol{z}_n)$ instead of $\lambda(\boldsymbol{x}_n;\boldsymbol{z}_n)$, we obtain a corresponding approximation of the GLF, namely,

$$f(\boldsymbol{z}_n|\boldsymbol{x}_n) \approx \widetilde{f}(\boldsymbol{z}_n|\boldsymbol{x}_n) \triangleq \exp(K\widetilde{\lambda}(\boldsymbol{x}_n;\boldsymbol{z}_n)). \tag{1.102}$$

1.3.4.2 Distributed Calculation of the GLF

Within the approximation given by (1.100) and (1.102), the distributed calculation of the GLF $f(\boldsymbol{z}_n|\boldsymbol{x}_n)$ at time n amounts to the distributed calculation of the R numbers $a_{n,r}(\boldsymbol{z}_n)$, $r = 1, 2, \ldots, R$, each of which is an average of the respective local expansion coefficients $\alpha_{n,k,r}(\boldsymbol{z}_{n,k})$, $k = 1, 2, \ldots, K$. Note that because the measurements $\boldsymbol{z}_{n,k}$ have been observed and thus are fixed, we now have to average numbers instead of functions. Thus, the average expansion coefficients $a_{n,r}(\boldsymbol{z}_n)$, $r = 1, 2, \ldots, R$ can be calculated in a distributed manner via R instances of an average consensus algorithm [24, 25]. In addition (see (1.102)), each agent also needs to know the total number of agents, K. Distributed algorithms for counting the number of agents are available (e.g., [67]).

In iteration i of the rth instance of the average consensus algorithm, which is used to calculate $a_{n,r}(\boldsymbol{z}_n)$, each agent k updates an "internal state" $\zeta_{k,r}$ according to

$$\zeta_{k,r}^{(i)} = \sum_{k' \in \mathfrak{N}_k} \omega_{k,k'}^{(i)} \zeta_{k',r}^{(i-1)}. \tag{1.103}$$

Here, $\{\omega_{k,k'}^{(i)}\}_{k' \in \mathfrak{N}_k}$ is a set of weights whose choice is discussed in [24, 68, 25]. Hence, $\zeta_{k,r}^{(i)}$ is a linear combination of the preceding (i.e., at iteration $i-1$) internal state of agent k and the preceding internal states of the neighbor agents k'. The iteration is initialized by choosing the initial internal states as $\zeta_{k,r}^{(0)} = \alpha_{n,k,r}(\boldsymbol{z}_{n,k})$. Then, for a suitable choice of the weights, and using our assumption that the agent network is connected, it can be shown [25] that the internal state $\zeta_{k,r}^{(i)}$ of each agent k converges to the desired average $a_{n,r}(\boldsymbol{z}_n)$, i.e.,

$$\lim_{i \to \infty} \zeta_{k,r}^{(i)} = a_{n,r}(\boldsymbol{z}_n) = \frac{1}{K} \sum_{k'=1}^{K} \alpha_{n,k',r}(\boldsymbol{z}_{n,k'}). \tag{1.104}$$

For a finite number $i_{\max}$ of iterations, complete convergence cannot be achieved; this means that the internal states $\zeta_{k,r}^{(i_{\max})}$ will be (slightly) different from $a_{n,r}(\boldsymbol{z}_n)$, and also (slightly) different across the agents k.

Several standard choices of the weights $\omega_{k,k'}^{(i)}$ are available [68, 24]. A popular choice is given by the Metropolis weights [24]

$$\omega_{k,k'} = \begin{cases} \dfrac{1}{\max\{|\mathfrak{N}_k|, |\mathfrak{N}_{k'}|\}}, & k' \neq k, \\ 1 - \sum_{k'' \in \mathfrak{N}_k \setminus \{k\}} \omega_{k,k''}, & k' = k. \end{cases} \tag{1.105}$$

Note that these weights do not depend on the iteration index i. Their calculation at agent k requires that agent k knows both $|\mathfrak{N}_k|$ and $|\mathfrak{N}_{k'}|$ for all $k' \in \mathfrak{N}_k$. Certain other choices of the weights require less knowledge [24].

In each iteration of the consensus scheme, agent k has to broadcast its internal states $\zeta_{k,r}^{(i)}$, $r = 1, 2, \ldots, R$ to its neighbors $k' \in \mathfrak{N}_k \setminus \{k\}$. Furthermore, agent k receives the internal states $\zeta_{k',r}^{(i)}$, $r = 1, 2, \ldots, R$ from its neighbors $k' \in \mathfrak{N}_k \setminus \{k\}$. Thus, during one time step n, each agent k has to broadcast a total of $i_{\max} R$ real numbers to its

neighbors.

1.3.4.3 Calculation of the Local Expansion Coefficients

The local expansion coefficients $\alpha_{n,k,r}(z_{n,k})$, $r = 1, 2, \ldots, R$ arising in the log-LLF approximation (1.99) are calculated *locally* at the respective agent k. This can be done by means of a least-squares (LS) fit between the log-LLF $\ln f(z_{n,k}|\boldsymbol{x}_n)$ and the approximating expansion $\sum_{r=1}^{R} \alpha_{n,k,r}(z_{n,k})\, \varphi_r(\boldsymbol{x}_n)$. In view of the way the local PF operates, the approximation does not need to be good for all possible states $\boldsymbol{x}_n$ but only in those regions of the state space where the current particles $\boldsymbol{x}_{n,k}^{(m)}$ are located. Therefore, the LS fit at agent k minimizes, with respect to the vector of expansion coefficients $\boldsymbol{\alpha}_{n,k} \overset{\Delta}{=} (\alpha_{n,k,1}(z_{n,k}) \cdots \alpha_{n,k,R}(z_{n,k}))^{\mathrm{T}}$, the LS approximation error of the expansion (1.99) evaluated at the particles $\boldsymbol{x}_{n,k}^{(m)}$, $m = 1, 2, \ldots, M$, i.e.,

$$\sum_{m=1}^{M} \Big(\ln f(z_{n,k}|\boldsymbol{x}_{n,k}^{(m)}) - \sum_{r=1}^{R} \alpha_{n,k,r}(z_{n,k})\, \varphi_r(\boldsymbol{x}_{n,k}^{(m)}) \Big)^2 = \|\boldsymbol{\xi}_{n,k} - \boldsymbol{\Phi}_{n,k}\boldsymbol{\alpha}_{n,k}\|^2 \to \min_{\boldsymbol{\alpha}_{n,k}} . \tag{1.106}$$

Here, $\boldsymbol{\xi}_{n,k} \overset{\Delta}{=} \big(\ln f(z_{n,k}|\boldsymbol{x}_{n,k}^{(1)}) \cdots \ln f(z_{n,k}|\boldsymbol{x}_{n,k}^{(M)}) \big)^{\mathrm{T}}$ and $\boldsymbol{\Phi}_{n,k}$ is an $M \times R$ matrix with columns $\boldsymbol{\phi}_{n,k,r} \overset{\Delta}{=} \big(\varphi_r(\boldsymbol{x}_{n,k}^{(1)}) \cdots \varphi_r(\boldsymbol{x}_{n,k}^{(M)}) \big)^{\mathrm{T}}$. Assuming that $M \geq R$ (i.e., there are at least as many particles as expansion coefficients) and that $\boldsymbol{\Phi}_{n,k}$ has full rank, the solution to the LS problem (1.106) is given by [69]

$$\widehat{\boldsymbol{\alpha}}_{n,k} = \boldsymbol{\Phi}_{n,k}^{\#}\, \boldsymbol{\xi}_{n,k} , \quad \text{with } \boldsymbol{\Phi}_{n,k}^{\#} \overset{\Delta}{=} (\boldsymbol{\Phi}_{n,k}^{\mathrm{T}} \boldsymbol{\Phi}_{n,k})^{-1} \boldsymbol{\Phi}_{n,k}^{\mathrm{T}} . \tag{1.107}$$

Numerical aspects of computing $\widehat{\boldsymbol{\alpha}}_{n,k}$ in (1.107) are discussed in [69, 70]. The elements $\widehat{\alpha}_{n,k,r}$ of $\widehat{\boldsymbol{\alpha}}_{n,k}$ are then used to initialize the LC according to $\zeta_{k,r}^{(0)} = \widehat{\alpha}_{n,k,r}$.

A summary of the LC-based DPF algorithm is provided in listing (1.108).

LC-based DPF

each agent starts with its own set of particles $\boldsymbol{x}_{0,k}^{(m)} \sim f(\boldsymbol{x}_0)$ and identical weights $w_{0,k}^{(m)} = 1/M$, where $m = 1, 2, \ldots, M$.

repeat at every agent k for $n \geq 1$:

(propose a new set of particles)

$$\boldsymbol{x}_{n,k}^{(m)} \sim q(\boldsymbol{x}_n | \boldsymbol{x}_{n-1,k}^{(m)}), \ m = 1, 2, \ldots, M$$

(calculate an approximation $\widetilde{f}_k(\boldsymbol{z}_n|\boldsymbol{x}_n)$ of the GLF $f(\boldsymbol{z}_n|\boldsymbol{x}_n)$)

- calculate the coefficient vector $\widehat{\boldsymbol{\alpha}}_{n,k}$ according to (1.107)
- for $r = 1, 2, \ldots, R$:
  - initialize the internal LC state: $\zeta_{k,r}^{(0)} = \widehat{\alpha}_{n,k,r}$
  - for $i = 1, 2, \ldots, i_{\max}$, update the internal LC state $\zeta_{k,r}^{(i)}$ according to (1.103)
  - calculate $\widetilde{f}_k(\boldsymbol{z}_n|\boldsymbol{x}_n) = \exp\Big(K \sum_{r=1}^{R} \zeta_{k,r}^{(i_{\max})} \varphi_r(\boldsymbol{x}_n)\Big)$ (cf. (1.102), (1.100)) (1.108)

(calculate the weights of the proposed particles)

- calculate nonnormalized weights:

$$\widetilde{w}_{n,k}^{(m)} = w_{n-1,k}^{(m)} \frac{\widetilde{f}_k(\boldsymbol{z}_n|\boldsymbol{x}_{n,k}^{(m)}) \, f(\boldsymbol{x}_{n,k}^{(m)}|\boldsymbol{x}_{n-1,k}^{(m)})}{q(\boldsymbol{x}_{n,k}^{(m)}|\boldsymbol{x}_{n-1,k}^{(m)})}, \ m = 1, 2, \ldots, M$$

- normalize weights:

$$w_n^{(m)} = \widetilde{w}_n^{(m)} / \textstyle\sum_{m'=1}^{M} \widetilde{w}_n^{(m')}$$

(compute the state estimate)

$$\widehat{\boldsymbol{x}}_{n,k} = \textstyle\sum_{m=1}^{M} w_{n,k}^{(m)} \boldsymbol{x}_{n,k}^{(m)}$$

(if necessary, perform resampling [51, 20])

end

1.3.4.4 Exponential Family

An alternative LC scheme is possible if the LLFs of all the agents k (viewed as conditional pdfs of $\boldsymbol{z}_{n,k}$) belong to the exponential family of distributions [71], i.e.,

$$f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n) = c_{n,k}(\boldsymbol{z}_{n,k}) \exp\big(\boldsymbol{b}_{n,k}^{\mathrm{T}}(\boldsymbol{x}_n) \boldsymbol{d}_{n,k}(\boldsymbol{z}_{n,k}) - g_{n,k}(\boldsymbol{x}_n)\big), \quad k = 1, 2, \ldots, K, \quad (1.109)$$

with some functions $c_{n,k}(\cdot) \in \mathbb{R}_+$, $\boldsymbol{b}_{n,k}(\cdot) \in \mathbb{R}^q$, $\boldsymbol{d}_{n,k}(\cdot) \in \mathbb{R}^q$, and $g_{n,k}(\cdot) \in \mathbb{R}$, where $q \geq N$. Inserting this expression into (1.89) yields for the GLF

$$f(\boldsymbol{z}_n|\boldsymbol{x}_n) = C_n(\boldsymbol{z}_n) \exp\big(K G_n(\boldsymbol{x}_n; \boldsymbol{z}_n)\big), \quad (1.110)$$

where $C_n(\boldsymbol{z}_n) \stackrel{\Delta}{=} \prod_{k=1}^{K} c_{n,k}(\boldsymbol{z}_{n,k})$ and

$$G_n(\boldsymbol{x}_n; \boldsymbol{z}_n) \stackrel{\Delta}{=} \frac{1}{K} \sum_{k=1}^{K} \big(\boldsymbol{b}_{n,k}^{\mathrm{T}}(\boldsymbol{x}_n) \boldsymbol{d}_{n,k}(\boldsymbol{z}_{n,k}) - g_{n,k}(\boldsymbol{x}_n)\big). \quad (1.111)$$

The normalization factor $C_n(\boldsymbol{z}_n)$ does not depend on the state $\boldsymbol{x}_n$ and is hence irrelevant due to weight normalization (see Subsection 1.3.3).

As an alternative to expanding the log-LLFs as in (1.99), we may use separate

finite-order function expansions of the state-dependent functions involved in (1.111) [13], i.e.,

$$\boldsymbol{b}_{n,k}(\boldsymbol{x}_n) \approx \sum_{r=1}^{R_b} \boldsymbol{\beta}_{n,k,r}\, \varphi_r(\boldsymbol{x}_n), \quad g_{n,k}(\boldsymbol{x}_n) \approx \sum_{r=1}^{R_g} \gamma_{n,k,r}\, \psi_r(\boldsymbol{x}_n), \quad k = 1, 2, \ldots, K. \tag{1.112}$$

Here, the coefficients $\boldsymbol{\beta}_{n,k,r}$ are vectors of the same dimension as $\boldsymbol{b}_{n,k}(\boldsymbol{x}_n)$. The coefficients $\boldsymbol{\beta}_{n,k,r}$ and $\gamma_{n,k,r}$ can again be calculated by means of LS fitting, similarly to Subsection 1.3.4.3; note that one separate LS fit is required for each component of $\boldsymbol{\beta}_{n,k,r}$. Inserting (1.112) into (1.111) and changing the order of summation yields

$$G_n(\boldsymbol{x}_n; \boldsymbol{z}_n) \approx \widetilde{G}_n(\boldsymbol{x}_n; \boldsymbol{z}_n) \stackrel{\Delta}{=} \sum_{r=1}^{R_b} B_{n,r}(\boldsymbol{z}_n)\varphi_r(\boldsymbol{x}_n) - \sum_{r=1}^{R_g} \Gamma_{n,r}\psi_r(\boldsymbol{x}_n), \tag{1.113}$$

with

$$B_{n,r}(\boldsymbol{z}_n) \stackrel{\Delta}{=} \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\beta}_{n,k,r}^{\mathrm{T}} \boldsymbol{d}_{n,k}(\boldsymbol{z}_{n,k}), \quad \Gamma_{n,r} \stackrel{\Delta}{=} \frac{1}{K}\sum_{k=1}^{K} \gamma_{n,k,r}. \tag{1.114}$$

Using in (1.110) the approximation $\widetilde{G}_n(\boldsymbol{x}_n; \boldsymbol{z}_n)$ instead of $G_n(\boldsymbol{x}_n; \boldsymbol{z}_n)$ yields an approximation of the GLF.

The R_b numbers $B_{n,r}(\boldsymbol{z}_k)$, $r = 1, 2, \ldots, R_b$ and the R_g numbers $\Gamma_{n,r}$, $r = 1, 2, \ldots, R_g$ are seen in (1.114) to be averages of the local quantities $a_{n,k,r}(\boldsymbol{z}_{n,k}) \stackrel{\Delta}{=} \boldsymbol{\beta}_{n,k,r}^{\mathrm{T}} \boldsymbol{d}_{n,k}(\boldsymbol{z}_{n,k})$ and $\gamma_{n,k,r}$, respectively. Hence, they can be calculated in a distributed manner via $R_b + R_g$ instances of an average consensus algorithm, similarly to Subsection 1.3.4.2. Note that this distributed calculation assumes that each agent k knows its own functions $\boldsymbol{b}_{n,k}(\cdot)$, $\boldsymbol{d}_{n,k}(\cdot)$, and $g_{n,k}(\cdot)$, but not the respective functions of the other agents $k' \neq k$.

This distributed calculation of (an approximation of) the GLF $f(\boldsymbol{z}_n|\boldsymbol{x}_n)$ may be preferable over the general scheme presented in Subsections 1.3.4.1–1.3.4.3 if it is easier to approximate the functions $\boldsymbol{b}_{n,k}(\boldsymbol{x}_n)$ and $g_{n,k}(\boldsymbol{x}_n)$ than the LLFs $\ln f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n)$. In particular, there is a reduction in the number of real numbers each agent has to broadcast to its neighbors if $R_b + R_g < R$, where R is the order of function expansion used in the general scheme.

1.3.4.5 Gaussian Observation Noise

An important special case of an exponential-family LLF arises with additive Gaussian observation noise [13]. Here, the measurement model of agent k in (1.88) is specialized according to

$$\boldsymbol{z}_{n,k} = \boldsymbol{m}_{n,k}(\boldsymbol{x}_n) + \boldsymbol{v}_{n,k}, \quad n = 1, 2, \ldots; \quad k = 1, 2, \ldots, K, \tag{1.115}$$

where $\boldsymbol{m}_{n,k}(\boldsymbol{x}_n)$ is a generally nonlinear function and the additive observation noise $\boldsymbol{v}_{n,k}$ satisfies the independence properties formulated in Section 1.3.1 and, in addition, is Gaussian with zero mean and covariance matrix $\boldsymbol{C}_{n,k}$. It follows that the LLF of

agent k is given, up to a normalization factor, by

$$f(z_{n,k}|\boldsymbol{x}_n) \propto \exp\Big(-\frac{1}{2}(z_{n,k}-\boldsymbol{m}_{n,k}(\boldsymbol{x}_n))^{\mathrm{T}}\boldsymbol{C}_{n,k}^{-1}(z_{n,k}-\boldsymbol{m}_{n,k}(\boldsymbol{x}_n))\Big). \tag{1.116}$$

This is easily verified to be an exponential-family LLF (1.109) with

$$\boldsymbol{b}_{n,k}(\boldsymbol{x}_n) = \boldsymbol{m}_{n,k}(\boldsymbol{x}_n), \tag{1.117}$$

$$\boldsymbol{d}_{n,k}(z_{n,k}) = \boldsymbol{C}_{n,k}^{-1} z_{n,k}, \tag{1.118}$$

$$g_{n,k}(\boldsymbol{x}_n) = \frac{1}{2}\boldsymbol{m}_{n,k}^{\mathrm{T}}(\boldsymbol{x}_n)\boldsymbol{C}_{n,k}^{-1}\boldsymbol{m}_{n,k}(\boldsymbol{x}_n). \tag{1.119}$$

As in Subsection 1.3.4.4, we may use separate function expansion approximations of $\boldsymbol{b}_{n,k}(\boldsymbol{x}_n) = \boldsymbol{m}_{n,k}(\boldsymbol{x}_n)$ and $g_{n,k}(\boldsymbol{x}_n)$. However, an alternative approach is possible because according to (1.119), $g_{n,k}(\boldsymbol{x}_n)$ is a function of $\boldsymbol{m}_{n,k}(\boldsymbol{x}_n)$ [13]. Indeed, we may insert a function expansion approximation of $\boldsymbol{b}_{n,k}(\boldsymbol{x}_n) = \boldsymbol{m}_{n,k}(\boldsymbol{x}_n)$, i.e.,

$$\boldsymbol{m}_{n,k}(\boldsymbol{x}_n) \approx \widetilde{\boldsymbol{m}}_{n,k}(\boldsymbol{x}_n) \stackrel{\Delta}{=} \sum_{r=1}^{R_b} \boldsymbol{\beta}_{n,k,r}\,\varphi_r(\boldsymbol{x}_n), \tag{1.120}$$

into (1.119) to obtain an "induced" function expansion approximation of $g_{n,k}(\boldsymbol{x}_n)$,

$$\begin{aligned} g_{n,k}(\boldsymbol{x}_n) \approx \widetilde{g}_{n,k}(\boldsymbol{x}_n) &\stackrel{\Delta}{=} \frac{1}{2}\widetilde{\boldsymbol{m}}_{n,k}^{\mathrm{T}}(\boldsymbol{x}_n)\boldsymbol{C}_{n,k}^{-1}\widetilde{\boldsymbol{m}}_{n,k}(\boldsymbol{x}_n) \\ &= \frac{1}{2}\sum_{r_1=1}^{R_b}\sum_{r_2=1}^{R_b}\boldsymbol{\beta}_{n,k,r_1}^{\mathrm{T}}\boldsymbol{C}_{n,k}^{-1}\boldsymbol{\beta}_{n,k,r_2}\,\varphi_{r_1}(\boldsymbol{x}_n)\varphi_{r_2}(\boldsymbol{x}_n). \end{aligned} \tag{1.121}$$

Using any one-to-one mapping of the double index $(r_1, r_2) \in \{1, 2, \ldots, R_b\} \times \{1, 2, \ldots, R_b\}$ to a single index $r \in \{1, 2, \ldots, R_g\}$, with $R_g = R_b^2$, we can rewrite (1.121) as (cf. (1.112))

$$\widetilde{g}_{n,k}(\boldsymbol{x}_n) = \sum_{r=1}^{R_g}\gamma_{n,k,r}\,\psi_r(\boldsymbol{x}_n), \tag{1.122}$$

where $\gamma_{n,k,r} = \frac{1}{2}\boldsymbol{\beta}_{n,k,r_1}^{\mathrm{T}}\boldsymbol{C}_{n,k}^{-1}\boldsymbol{\beta}_{n,k,r_2}$ and $\psi_r(\boldsymbol{x}_n) = \varphi_{r_1}(\boldsymbol{x}_n)\varphi_{r_2}(\boldsymbol{x}_n)$. It can be easily verified that the resulting approximate GLF (cf. (1.89)) can be written as $\widetilde{f}(z_k|\boldsymbol{x}_k) \propto \exp\big(-\frac{1}{2}\widetilde{Q}_n(z_n, \boldsymbol{x}_n)\big)$ with $\widetilde{Q}_n(z_n, \boldsymbol{x}_n) \stackrel{\Delta}{=} \sum_{k=1}^{K}(z_{n,k} - \widetilde{\boldsymbol{m}}_{n,k}(\boldsymbol{x}_n))^{\mathrm{T}}\boldsymbol{C}_{n,k}^{-1}(z_{n,k} - \widetilde{\boldsymbol{m}}_{n,k}(\boldsymbol{x}_n))$. This equals the true GLF $f(z_k|\boldsymbol{x}_k)$, except that the means $\boldsymbol{m}_{n,k}(\boldsymbol{x}_n)$ are replaced by their approximations $\widetilde{\boldsymbol{m}}_{n,k}(\boldsymbol{x}_n)$.

1.3.4.6 Distributed Gaussian Particle Filter

The Gaussian PF (GPF) [72] uses a Gaussian approximation of the posterior pdf $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$. This Gaussian approximation is derived from a weighted particle set, which is calculated in a similar way as in the PF except that no resampling is required. Note that the measurement model is still allowed to be non-Gaussian, as in (1.88).

In a distributed GPF, the local GPF at agent k propagates a Gaussian approximation $\mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_{n,k}, \boldsymbol{\Sigma}_{n,k})$ of the global posterior pdf $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$. At time n, particles $\boldsymbol{x}_{n,k}^{(m)}$, $m = 1, 2, \ldots, M$ are drawn from the preceding Gaussian approximation $\mathcal{N}(\boldsymbol{x}_{n-1}; \boldsymbol{\mu}_{n-1,k}, \boldsymbol{\Sigma}_{n-1,k})$. Then, the prediction and update steps are performed as described in Section 1.3.3. From the resulting weighted particles $\{(\boldsymbol{x}_{n,k}^{(m)}, w_{n,k}^{(m)})\}_{m=1}^{M}$, a new Gaussian approximation $\mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_{n,k}, \boldsymbol{\Sigma}_{n,k})$ is determined according to

$$\boldsymbol{\mu}_{n,k} = \sum_{m=1}^{m} w_{n,k}^{(m)} \boldsymbol{x}_{n,k}^{(m)} \quad \text{and} \quad \boldsymbol{\Sigma}_{n,k} = \sum_{m=1}^{M} w_{n,k}^{(m)} \boldsymbol{x}_{n,k}^{(m)} \boldsymbol{x}_{n,k}^{(m)\mathrm{T}} - \boldsymbol{\mu}_{n,k} \boldsymbol{\mu}_{n,k}^{\mathrm{T}}. \tag{1.123}$$

The weighted sample mean $\boldsymbol{\mu}_{n,k}$ also constitutes the local approximation $\widehat{\boldsymbol{x}}_{n,k}$ of the MMSE state estimate $\widehat{\boldsymbol{x}}_n^{\mathrm{MMSE}}$ in (1.92), i.e., $\widehat{\boldsymbol{x}}_{n,k} = \boldsymbol{\mu}_{n,k}$.

Just as in the conventional DPF (cf. (1.93)), the update step of the local GPFs requires the GLF evaluated at the current local particles, i.e., $f(\boldsymbol{z}_n|\boldsymbol{x}_n)$ for $\boldsymbol{x}_n = \boldsymbol{x}_{n,k}^{(m)}$, $m = 1, 2, \ldots, M$. An approximation of $f(\boldsymbol{z}_n|\boldsymbol{x}_{n,k}^{(m)})$ can be calculated in a distributed manner by the LC as explained earlier. However, the following variation yields a significant reduction of computational complexity at the cost of some increase in local communications [13]. Inspired by the parallel GPF implementation proposed in [73], the idea is to "distribute" the M particles over the K agents, such that each local GPF uses a significantly reduced set of particles, and to combine the results of the local GPFs via additional average consensus operations. The local GPF at agent k uses only $M_k < M$ particles, where $\sum_{k=1}^{K} M_k = M$. In the local weight update step at agent k, for each particle $\boldsymbol{x}_{n,k}^{(m)}$ from the reduced local particle set, $m \in \{1, 2, \ldots, M_k\}$, a nonnormalized weight $\widetilde{w}_{n,k}^{(m)}$ is calculated according to (1.93). This requires the GLF evaluated at the M_k particles, i.e., $f(\boldsymbol{z}_n|\boldsymbol{x}_{n,k}^{(m)})$, $m = 1, 2, \ldots, M_k$, which was previously calculated (approximately) by the LC. Furthermore, the sum of the resulting nonnormalized weights is calculated, i.e., $\widetilde{W}_{n,k} = \sum_{m=1}^{M_k} \widetilde{w}_{n,k}^{(m)}$.

Next, from the weighted particles $\{(\boldsymbol{x}_{n,k}^{(m)}, \widetilde{w}_{n,k}^{(m)})\}_{m=1}^{M_k}$, a local nonnormalized mean vector and a local nonnormalized correlation matrix are calculated at each agent k as

$$\boldsymbol{\mu}'_{n,k} = \sum_{m=1}^{M_k} \widetilde{w}_{n,k}^{(m)} \boldsymbol{x}_{n,k}^{(m)}, \qquad \mathbf{R}'_{n,k} = \sum_{m=1}^{M_k} \widetilde{w}_{n,k}^{(m)} \boldsymbol{x}_{n,k}^{(m)} \boldsymbol{x}_{n,k}^{(m)\mathrm{T}}. \tag{1.124}$$

Finally, the local means and correlations from all the agents are combined into a global mean and covariance:

$$\bar{\boldsymbol{\mu}}_{n,k} = \frac{1}{W_{n,k}} \sum_{k=1}^{K} \boldsymbol{\mu}'_{n,k}, \qquad \bar{\boldsymbol{\Sigma}}_{n,k} = \frac{1}{W_{n,k}} \sum_{k=1}^{K} \mathbf{R}'_{n,k} - \bar{\boldsymbol{\mu}}_{n,k} \bar{\boldsymbol{\mu}}_{n,k}^{\mathrm{T}}, \tag{1.125}$$

where $W_{n,k} \triangleq \sum_{k=1}^{K} \widetilde{W}_{n,k}$. Approximations of the sums $\sum_{k=1}^{K} \boldsymbol{\mu}'_{n,k}$, $\sum_{k=1}^{K} \mathbf{R}'_{n,k}$, and $\sum_{k=1}^{K} \widetilde{W}_{n,k}$ can be calculated in a distributed manner by means of an average consensus algorithm (again assuming that the number of agents, K, is known to each agent). Subsequently, the division by $W_{n,k}$ and the subtraction of $\bar{\boldsymbol{\mu}}_{n,k} \bar{\boldsymbol{\mu}}_{n,k}^{\mathrm{T}}$ in (1.125)

are performed locally at each agent. This results in local approximations $\boldsymbol{\mu}_{n,k}$ and $\boldsymbol{\Sigma}_{n,k}$ of, respectively, $\bar{\boldsymbol{\mu}}_{n,k}$ and $\bar{\boldsymbol{\Sigma}}_{n,k}$ at each agent k. The state estimate at agent k, $\widehat{\mathbf{x}}_{n,k}$, is taken to be $\boldsymbol{\mu}_{n,k}$.

1.3.4.7 Distributed Proposal Adaptation

The choice of the proposal pdf $q(\boldsymbol{x}_n; \boldsymbol{x}_{n-1}^{(m)})$ used in the weight update step (1.93) strongly affects the performance of a PF. The proposal pdf should be similar to the posterior pdf $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$ [20]. Using the state-transition pdf $f(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(m)})$ as proposal pdf, as is done in the SIR filter, does not cater to this goal; in particular, it does not take into account the current measurement $\boldsymbol{z}_n$. The design of a proposal pdf taking into account the measurements is known as *proposal adaptation* [49, 50]. In a DPF, the proposal pdfs used by the local PFs should be adapted to the total (all-agents) measurement $\boldsymbol{z}_n = (\boldsymbol{z}_{n,1}^{\mathrm{T}}\, \boldsymbol{z}_{n,2}^{\mathrm{T}} \cdots \boldsymbol{z}_{n,K}^{\mathrm{T}})^{\mathrm{T}}$.

A distributed method for this type of "global" proposal adaptation can again be based on the average consensus principle [56]. In this method, each agent first calculates a "predistorted" local posterior pdf. A Gaussian approximation of the global posterior pdf is then obtained by fusing all the predistorted local posterior pdfs; this approximate global posterior pdf is used by the local PFs as a proposal pdf. The method is inspired by [52] but employs a different predistortion that enables the use of a Gaussian filter for calculating the predistorted local posterior pdfs. To derive the method, we note that the global posterior pdf can be developed as $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n}) = f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1}, \boldsymbol{z}_n) \propto f(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{z}_{1:n-1}) f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1}) = f(\boldsymbol{z}_n|\boldsymbol{x}_n) f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1})$, where Bayes' rule and (1.91) have been used. Inserting (1.89), we further obtain

$$f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n}) \propto \left(\prod_{k=1}^{K} f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n) \right) f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1}). \tag{1.126}$$

We now define a predistorted, nonnormalized "local pseudoposterior pdf" at agent k as

$$\widetilde{f}(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1}, \boldsymbol{z}_{n,k}) \stackrel{\Delta}{=} f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n) \left(f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1})\right)^{1/K}. \tag{1.127}$$

Furthermore, we consider the network-wide product of all the local pseudoposterior pdfs,

$$\prod_{k=1}^{K} \widetilde{f}(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1}, \boldsymbol{z}_{n,k}) = \left(\prod_{k=1}^{K} f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_n) \right) f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1}) \propto f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n}), \tag{1.128}$$

where (1.126) has been used in the last step. According to (1.128), the product of all the local pseudoposterior pdfs equals the global posterior pdf up to a factor. This fact can be used to calculate a Gaussian approximation of the global posterior pdf $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$—note that $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n})$ involves all the measurements and in fact would be the optimal proposal pdf—by a distributed evaluation of the leftmost side of (1.128). To make this possible, we use Gaussian approximations of the local pseudoposterior pdfs, i.e.,

$$\widetilde{f}(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1}, z_{n,k}) \approx \mathcal{N}(\boldsymbol{x}_n; \widetilde{\boldsymbol{\mu}}_{n,k}, \widetilde{\boldsymbol{\Sigma}}_{n,k}), \tag{1.129}$$

and of the global posterior pdf, i.e.,

$$f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n}) \approx q(\boldsymbol{x}_n; z_k) \stackrel{\Delta}{=} \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n). \tag{1.130}$$

Note that (1.130) also defines the global proposal pdf $q(\boldsymbol{x}_n; z_n)$. Inserting the approximations (1.129) and (1.130) into the factorization $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n}) \propto \prod_{k=1}^{K} \widetilde{f}(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1}, z_{n,k})$ (see (1.128)) yields

$$\mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n) \propto \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}_n; \widetilde{\boldsymbol{\mu}}_{n,k}, \widetilde{\boldsymbol{\Sigma}}_{n,k}). \tag{1.131}$$

Using this relation and the rules for a product of Gaussian densities [52, 74], the global mean $\boldsymbol{\mu}_n$ and global covariance $\boldsymbol{\Sigma}_n$—which determine $q(\boldsymbol{x}_n; z_n) = \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$—can be calculated from the local means $\widetilde{\boldsymbol{\mu}}_{n,k}$ and local covariances $\widetilde{\boldsymbol{\Sigma}}_{n,k}$ according to

$$\boldsymbol{\mu}_n = \boldsymbol{\Sigma}_n \sum_{k=1}^{K} (\widetilde{\boldsymbol{\Sigma}}_{n,k})^{-1} \widetilde{\boldsymbol{\mu}}_{n,k}\,, \quad \boldsymbol{\Sigma}_n = \left(\sum_{k=1}^{K} (\widetilde{\boldsymbol{\Sigma}}_{n,k})^{-1} \right)^{-1}. \tag{1.132}$$

Here, the sums $\sum_{k=1}^{K} (\widetilde{\boldsymbol{\Sigma}}_{n,k})^{-1} \widetilde{\boldsymbol{\mu}}_{n,k}$ and $\sum_{k=1}^{K} (\widetilde{\boldsymbol{\Sigma}}_{n,k})^{-1}$ can be computed in a distributed manner by means of an average consensus algorithm.

The calculation of the local mean $\widetilde{\boldsymbol{\mu}}_{n,k}$ and local covariance $\boldsymbol{\Sigma}_{n,k}$, as defined in (1.129), at agent k is based on the observation that (1.127) can be interpreted as the update step of a Bayesian filter using the predistorted predicted posterior pdf $\left(f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1})\right)^{1/K}$ instead of the true predicted posterior pdf $f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1})$. Each agent first calculates a Gaussian approximation of the predicted posterior pdf,

$$f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1}) \approx \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}'_{n,k}, \boldsymbol{\Sigma}'_{n,k}), \tag{1.133}$$

where $\boldsymbol{\mu}'_{n,k} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{x}_{n,k}^{(m)\prime}$ and $\boldsymbol{\Sigma}'_{n,k} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{x}_{n,k}^{(m)\prime} (\boldsymbol{x}_{n,k}^{(m)\prime})^{\mathrm{T}} - \boldsymbol{\mu}'_{n,k} (\boldsymbol{\mu}'_{n,k})^{\mathrm{T}}$. Here, $\{\boldsymbol{x}_{n,k}^{(m)\prime}\}_{m=1}^{M}$ is a set of temporary particles that are randomly drawn from $f(\boldsymbol{x}_n|\boldsymbol{x}_{n-1,k}^{(m)})$, where $\{\boldsymbol{x}_{n-1,k}^{(m)}\}_{m=1}^{M}$ is the set of particles resulting from the preceding filtering step at time $n-1$. The approximation (1.133) implies

$$\left(f(\boldsymbol{x}_n|\boldsymbol{z}_{1:n-1})\right)^{1/K} \approx \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}'_{n,k}, K\boldsymbol{\Sigma}'_{n,k}). \tag{1.134}$$

Using the Gaussian approximations (1.129) and (1.134) in (1.127) gives

$$\mathcal{N}(\boldsymbol{x}_k; \widetilde{\boldsymbol{\mu}}_{n,k}, \widetilde{\boldsymbol{\Sigma}}_{n,k}) = f(\boldsymbol{z}_{n,k}|\boldsymbol{x}_k)\, \mathcal{N}(\boldsymbol{x}_k; \boldsymbol{\mu}'_{n,k}, K\boldsymbol{\Sigma}'_{n,k}). \tag{1.135}$$

This can be calculated by the update step of a Gaussian filter with input mean $\boldsymbol{\mu}'_{n,k}$, input covariance $K\boldsymbol{\Sigma}'_{n,k}$, and measurement $\boldsymbol{z}_{n,k}$. This Gaussian filter update step is performed locally at each agent and produces $\widetilde{\boldsymbol{\mu}}_{n,k}$ and $\widetilde{\boldsymbol{\Sigma}}_{n,k}$. Examples of a Gaussian filter include the extended Kalman filter [19, 42, 36], the unscented Kalman filter [44, 45], the cubature Kalman filter [46], and the filters described in [75].

1.3.5 DIFFUSION-BASED METHODS

In this subsection, we address diffusion cooperation among agents as in Subsection 1.2.7 but implemented with particle filtering [60]. However, now the considered state-space models are nonlinear and the pdfs of $\boldsymbol{u}_n$ in (1.87) and $\boldsymbol{v}_{n,k}$ in (1.88) are assumed to be known up to proportionality constants. We reiterate that these pdfs can be of any form.

The diffusion-based DPF presented here follows the spirit of cooperation from Subsection 1.2.7 in the sense that the agents share moments (or parameters of approximating distributions) of the unknown state. We assume that the agents at time $n = 0$ start with the generation of their own particles $\boldsymbol{x}_{0,k}^{(m)}$, $m = 1, 2, \ldots, M$, which are drawn from a prior pdf $f(\boldsymbol{x}_0)$, i.e.,

$$\boldsymbol{x}_{0,k}^{(m)} \sim f(\boldsymbol{x}_0), \quad m = 1, 2, \ldots, M; \; k = 1, 2, \ldots, K. \tag{1.136}$$

Thus, before the tracking of the state starts, each agent has a representation of the prior pdf of $\boldsymbol{x}_0$ given by $\left\{\boldsymbol{x}_{0,k}^{(m)}, w_{0,k}^{(m)} = \frac{1}{M}\right\}_{m=1}^{M}$. Similarly, before processing the measurement vector $\boldsymbol{z}_{n,k}$, the local PF of agent k has a representation of the posterior pdf of $\boldsymbol{x}_{n-1,k}$ given by $\left\{\boldsymbol{x}_{n-1,k}^{(m)}, w_{n-1,k}^{(m)} = \frac{1}{M}\right\}_{m=1}^{M}$, which serves as a prior for the state $\boldsymbol{x}_n$.

Next, we explain how we obtain $\left\{\boldsymbol{x}_{n,k}^{(m)}, w_{n,k}^{(m)} = \frac{1}{M}\right\}_{m=1}^{M}$ from $\left\{\boldsymbol{x}_{n-1,k}^{(m)}, w_{n-1,k}^{(m)} = \frac{1}{M}\right\}_{m=1}^{M}$. The PF of each agent k propagates the particles $\boldsymbol{x}_{n-1,k}^{(m)}$ to particles that represent possible values of the state at time n. The propagation is carried out by drawing samples from the importance pdf $q(\boldsymbol{x}_n|\boldsymbol{x}_{n-1,k}^{(m)})$, i.e.,

$$\widetilde{\boldsymbol{x}}_{n,k}^{(m)} \sim q(\boldsymbol{x}_n|\boldsymbol{x}_{n-1,k}^{(m)}), \quad m = 1, 2, \ldots, M; \; k = 1, 2, \ldots, K. \tag{1.137}$$

As described in Subsection 1.3.3, the PF subsequently computes the weights of these particles. This is accomplished by

$$\widetilde{w}_{n,k}^{(m)} \propto \frac{f(\boldsymbol{z}_{n,k}|\widetilde{\boldsymbol{x}}_{n,k}^{(m)}) f(\widetilde{\boldsymbol{x}}_{n,k}^{(m)}|\boldsymbol{x}_{n-1,k}^{(m)})}{q(\widetilde{\boldsymbol{x}}_{n,k}^{(m)}|\boldsymbol{x}_{n-1,k}^{(m)})}, \tag{1.138}$$

where $\sum_{m=1}^{M} \widetilde{w}_{n,k}^{(m)} = 1$. With the particles $\widetilde{\boldsymbol{x}}_{n,k}^{(m)}$ and their weights $\widetilde{w}_{n,k}^{(m)}$, the agent k has a local approximation of the posterior pdf of $\boldsymbol{x}_n$ given by $\left\{\widetilde{\boldsymbol{x}}_{n,k}^{(m)}, \widetilde{w}_{n,k}^{(m)}\right\}_{m=1}^{M}$. The cooperation among the agents requires that they exchange these approximating local posterior pdfs and fuse them to form new local posteriors. The best approach in terms of accuracy would be that each agent broadcasts all the particles and weights to its neighbors, but that would be quite costly in terms of communication load because the number of particles and weights is usually large. An alternative is to use the particles and weights to construct a parametric distribution, that is, estimate the parameters of an approximating distribution of the weighted particles. Then, the agents would only exchange the parameters of the approximating distribution.

In the sequel, the approximating distributions are multivariate Gaussians. Thus, the agents proceed by computing the means and covariance matrices of the approxi-

mating Gaussians. First, the agents obtain the means by

$$\widetilde{\boldsymbol{\mu}}_{n,k} = \sum_{m=1}^{M} \widetilde{w}_{n,k}^{(m)} \widetilde{\boldsymbol{x}}_{n,k}^{(m)}, \quad k = 1, 2, \ldots, K, \tag{1.139}$$

and then the covariance matrices by

$$\widetilde{\boldsymbol{\Sigma}}_{n,k} = \sum_{m=1}^{M} \widetilde{w}_{n,k}^{(m)} \left(\widetilde{\boldsymbol{x}}_{n,k}^{(m)} - \widetilde{\boldsymbol{\mu}}_{n,k}\right)\left(\widetilde{\boldsymbol{x}}_{n,k}^{(m)} - \widetilde{\boldsymbol{\mu}}_{n,k}\right)^{\mathrm{T}}, \quad k = 1, 2, \ldots, K. \tag{1.140}$$

In the next step, the agents exchange the means and covariance matrices with their neighbors and follow up with merging all the Gaussian posteriors into one Gaussian. To that end, the agents use coefficients $a_{\ell k}$ that quantify how much they trust their neighbors (defined earlier in Subsection 1.2.1, and here with $a_{\ell k}$ representing how much agent k trusts agent ℓ). We note that $\sum_{\ell \in \mathfrak{N}_k} a_{\ell k} = 1$. In the chapter on Bayesian Approach to Inference from this book (see its Subsection 1.2.3), it is explained that the Gaussian after merging has a covariance matrix and mean that can be expressed in terms of the individual covariance matrices and means of the agents by

$$\boldsymbol{\Sigma}_{n,k} = \left(\sum_{\ell \in \mathfrak{N}_k} a_{\ell k} \left(\widetilde{\boldsymbol{\Sigma}}_{n,k}\right)^{-1}\right)^{-1} \tag{1.141}$$

and

$$\boldsymbol{\mu}_{n,k} = \boldsymbol{\Sigma}_{n,k} \left(\sum_{\ell \in \mathfrak{N}_k} a_{\ell k} \left(\widetilde{\boldsymbol{\Sigma}}_{n,k}\right)^{-1} \widetilde{\boldsymbol{\mu}}_{n,k}\right), \tag{1.142}$$

respectively. More specifically, this choice of $\boldsymbol{\mu}_{n,k}$ and $\boldsymbol{\Sigma}_{n,k}$ minimizes the weighted sum of Kullback-Leibler distances $\sum_{\ell \in \mathfrak{N}_k} a_{\ell k} \mathcal{D}(\mathcal{N}_{n,k} \| \widetilde{\mathcal{N}}_{n,\ell})$, where $\mathcal{N}_{n,k}$ is the resulting Gaussian after merging, and $\widetilde{\mathcal{N}}_{n,\ell}$, $\ell \in \mathfrak{N}_k$ are the Gaussians of agent k and its neighbors.

At this point, the agents have more than one way of proceeding. A simple way is that the agents use the newly obtained Gaussian and that each draws a set of particles that represent it, i.e.,

$$\boldsymbol{x}_{n,k}^{(m)} \sim \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_{n,k}, \boldsymbol{\Sigma}_{n,k}), \quad m = 1, 2, \ldots, M; \; k = 1, 2, \ldots, K. \tag{1.143}$$

After generation of these particles, the agents have a discrete representation, $\left\{\boldsymbol{x}_{n,k}^{(m)}, w_{n,k}^{(m)} = \frac{1}{M}\right\}_{m=1}^{M}$, of the posterior pdf at time n that serves as a prior pdf of $\boldsymbol{x}_{n+1}$.

Another approach would be to avoid fusion of the local posteriors by (1.141) and (1.142) and instead draw particles directly from the mixture of Gaussians with components $\widetilde{\mathcal{N}}(\boldsymbol{x}_n; \widetilde{\boldsymbol{\mu}}_{n,\ell}, \widetilde{\boldsymbol{\Sigma}}_{n,\ell})$ and weights $a_{\ell k}$, i.e.,

$$\boldsymbol{x}_{n,k}^{(m)} \sim \sum_{\ell \in \mathfrak{N}_k} a_{\ell k} \widetilde{\mathcal{N}}(\boldsymbol{x}_n; \widetilde{\boldsymbol{\mu}}_{n,\ell}, \widetilde{\boldsymbol{\Sigma}}_{n,\ell}), \quad m = 1, 2, \ldots, M; \; k = 1, 2, \ldots, K. \tag{1.144}$$

Once done, the posterior pdf is again represented by $\left\{\boldsymbol{x}_{n,k}^{(m)}, w_{n,k}^{(m)} = \frac{1}{M}\right\}_{m=1}^{M}$.

An alternative direction is to first resample the original local particles $\widetilde{\boldsymbol{x}}_{n,k}^{(m)}$ according to their weights $\widetilde{w}_{n,k}^{(m)}$. Let the resampled particles be denoted by $\bar{\boldsymbol{x}}_{n,k}^{(m)}$. In the next step, these particles are rescaled and shifted so that they correspond to particles of a Gaussian distribution with mean $\boldsymbol{\mu}_{n,k}$ in (1.142) and covariance $\boldsymbol{\Sigma}_{n,k}$ in (1.141). It is not difficult to show that this can be achieved by

$$\boldsymbol{x}_{n,k}^{(m)} = \boldsymbol{Q}_{n,k}^{\mathrm{T}} \widetilde{\boldsymbol{Q}}_{n,k}^{-T} \left(\bar{\boldsymbol{x}}_{n,k}^{(m)} - \widetilde{\boldsymbol{\mu}}_{n,k}\right) + \boldsymbol{\mu}_{n,k}, \tag{1.145}$$

where $\boldsymbol{Q}_{n,k}$ is defined by $\boldsymbol{\Sigma}_{n,k} = \boldsymbol{Q}_{n,k}^{\mathrm{T}} \boldsymbol{Q}_{n,k}$ and $\widetilde{\boldsymbol{Q}}_{n,k}$ by $\widetilde{\boldsymbol{\Sigma}}_{n,k} = \widetilde{\boldsymbol{Q}}_{n,k}^{\mathrm{T}} \widetilde{\boldsymbol{Q}}_{n,k}$. The particles obtained by (1.145) have all equal weights, and thus they represent the prior pdf for the state $\boldsymbol{x}_{n+1}$. This diffusion-based approach is described in the listing (1.146). Note that the particles generated by (1.143) and (1.144) are all different whereas many of the ones obtained by (1.145) may be replicated.

Diffusion particle filter (adapt and then combine form)

each agent starts with its own set of particles $\boldsymbol{x}_{0,k}^{(m)} \sim f(\boldsymbol{x}_0)$, $m = 1, 2, \ldots, M$.

repeat at every agent k for $n \geq 0$:

(propose a new set of particles)

$$\widetilde{\boldsymbol{x}}_{n,k}^{(m)} \sim q(\boldsymbol{x}_n | \boldsymbol{x}_{n-1,k}^{(m)}), \quad m = 1, 2, \ldots, M$$

(compute the weights of the proposed particles)

$$\widetilde{w}_{n,k}^{(m)} \propto \frac{f(\boldsymbol{z}_{n,k} | \widetilde{\boldsymbol{x}}_{n,k}^{(m)}) f(\widetilde{\boldsymbol{x}}_{n,k}^{(m)} | \boldsymbol{x}_{n-1,k}^{(m)})}{q(\widetilde{\boldsymbol{x}}_{n,k}^{(m)} | \boldsymbol{x}_{n-1,k}^{(m)})}$$

(find the parameters of a Gaussian distribution that approximates the posterior)

$$\widetilde{\boldsymbol{\mu}}_{n,k} = \sum_{m=1}^{M} \widetilde{w}_{n,k}^{(m)} \widetilde{\boldsymbol{x}}_{n,k}^{(m)}$$

$$\widetilde{\boldsymbol{\Sigma}}_{n,k} = \sum_{m=1}^{M} \widetilde{w}_{n,k}^{(m)} \left(\widetilde{\boldsymbol{x}}_{n,k}^{(m)} - \widetilde{\boldsymbol{\mu}}_{n,k}\right)\left(\widetilde{\boldsymbol{x}}_{n,k}^{(m)} - \widetilde{\boldsymbol{\mu}}_{n,k}\right)^{\mathrm{T}} \tag{1.146}$$

(combine own posterior with the posteriors of the neighbors)

$$\boldsymbol{\Sigma}_{n,k} = \left(\sum_{\ell \in \mathfrak{N}_k} a_{\ell k} \left(\widetilde{\boldsymbol{\Sigma}}_{n,k}\right)^{-1}\right)^{-1}$$

$$\boldsymbol{\mu}_{n,k} = \boldsymbol{\Sigma}_{n,k} \left(\sum_{\ell \in \mathfrak{N}_k} a_{\ell k} \left(\widetilde{\boldsymbol{\Sigma}}_{n,k}\right)^{-1} \widetilde{\boldsymbol{\mu}}_{n,k}\right)$$

(resample the local particles according to their weights)

$$\left\{\left(\widetilde{\boldsymbol{x}}_{n,k}^{(m)}, \widetilde{w}_{n,k}^{(m)}\right)\right\}_{m=1}^{M} \xrightarrow{\text{resampling}} \left\{\left(\bar{\boldsymbol{x}}_{n,k}^{(m)}, w_{n,k}^{(m)} = \frac{1}{M}\right)\right\}_{m=1}^{M}$$

(rescale and shift the resampled local particles)

$$\boldsymbol{x}_{n,k}^{(m)} = \boldsymbol{Q}_{n,k}^{\mathrm{T}} \widetilde{\boldsymbol{Q}}_{n,k}^{-\mathrm{T}} \left(\bar{\boldsymbol{x}}_{n,k}^{(m)} - \widetilde{\boldsymbol{\mu}}_{n,k}\right) + \boldsymbol{\mu}_{n,k}, \text{ where } \boldsymbol{Q}_{n,k} \text{ and } \widetilde{\boldsymbol{Q}}_{n,k} \text{ are defined by}$$

$$\boldsymbol{\Sigma}_{n,k} = \boldsymbol{Q}_{n,k}^{\mathrm{T}} \boldsymbol{Q}_{n,k} \text{ and } \widetilde{\boldsymbol{\Sigma}}_{n,k} = \widetilde{\boldsymbol{Q}}_{n,k}^{\mathrm{T}} \widetilde{\boldsymbol{Q}}_{n,k}, \text{ respectively}$$

end

The performance of the diffusion cooperation can be improved if the calculation of the particle weights of each agent involves the measurements of the respective neighbors. This requires that the agents engage in two rounds of communication, one when they exchange their measurements with their neighbors, and the other when they exchange the parameters of their posterior pdfs. In this case, the calculation of

the weights in (1.138) is replaced by

$$\widetilde{w}_{n,k}^{(m)} \propto \frac{\left(\prod_{\ell\in\mathcal{N}_k} f(\boldsymbol{z}_{n,\ell}|\widetilde{\boldsymbol{x}}_{n,k}^{(m)})\right) f(\widetilde{\boldsymbol{x}}_{n,k}^{(m)}|\boldsymbol{x}_{n-1,k}^{(m)})}{q(\widetilde{\boldsymbol{x}}_{n,k}^{(m)}|\boldsymbol{x}_{n-1,k}^{(m)})}. \tag{1.147}$$

Everything else in the described process remains the same.

In summary, the diffusion-based particle filtering algorithm described in this subsection relies on approximating the local posterior pdfs by Gaussians whose parameters are then exchanged with neighbors. Upon receiving the Gaussian parameters from its neighbors, each agent either uses them to create a new Gaussian that serves as the final posterior pdf or simply exploits them directly as in (1.143) or (1.144) to generate particles that represent the support of the final posterior pdf and the prior pdf of $\boldsymbol{x}_{n+1}$. However, we reiterate that the underlying approximating distributions of $\left\{\widetilde{\boldsymbol{x}}_{n,k}^{(m)}, \widetilde{w}_{n,k}^{(m)}\right\}_{m=1}^{M}$ used by the agents do not have to be Gaussians.

1.4 CONCLUSIONS

In this chapter, we discussed the problem of distributed filtering in a network of agents. The filtering amounts to estimating a hidden state process based on measurements that are acquired by the agents and contain information about the state process. The agents cooperate by communicating relevant information with their neighbors. Two main types of filtering methods were addressed. The first is distributed Kalman filtering where the state-space models are linear and the state and observation noise processes are zero-mean and white vector processes with known covariance matrices. The second type is distributed particle filtering where the models are nonlinear and the pdfs characterizing the state and observation noise processes are known up to proportionality constants. The distributed Kalman filtering method was based on the diffusion strategy, with a discussion on the relation and differences relative to a consensus-type implementation. The distributed particle filtering methods were addressed by both consensus and diffusion strategies.

ACKNOWLEDGMENTS

The work of A. H. Sayed was supported by the NSF under grants ECCS-1407712 and CCF-1524250. The author is grateful to the IEEE for allowing reproduction of material from [12] in this book chapter. The work of P. M. Djurić was supported by the NSF under grant CCF-1618999. The work of F. Hawatsch was supported by the Austrian Science Fund (FWF) under grant P27370-N30 and by the Czech Science Foundation (GAČR) under grant 17-19638S.

1.5 APPENDIX

Lemma 1 (Convergence of Lyapunov recursion). *Consider a general Lyapunov recursion of the form*

$$\boldsymbol{Z}_{n+1} = \boldsymbol{C}_n \boldsymbol{Z}_n \boldsymbol{C}_n^T + \boldsymbol{B}_n \tag{1.148}$$

where the matrix sequences $\{\boldsymbol{B}_n, \boldsymbol{C}_n\}$ *converge uniformly to* $\{\boldsymbol{B}, \boldsymbol{C}\}$ *as* $n \to \infty$ *with* $\boldsymbol{C}$ *being a stable matrix (all its eigenvalues lie strictly inside the unit circle). Then, the sequence* $\boldsymbol{Z}_n$ *converges to the unique solution* $\boldsymbol{Z}$ *of the Lyapunov equation*

$$\boldsymbol{Z} = \boldsymbol{C}\boldsymbol{Z}\boldsymbol{C}^T + \boldsymbol{B}. \tag{1.149}$$

Proof: Using the vec notation we have

$$\text{vec}(\boldsymbol{Z}_{n+1}) = (\boldsymbol{C}_n \otimes \boldsymbol{C}_n)\text{vec}(\boldsymbol{Z}_n) + \text{vec}(\boldsymbol{B}_n) \triangleq \boldsymbol{T}_n \text{vec}(\boldsymbol{Z}_n) + \boldsymbol{d}_n, \tag{1.150}$$

where we introduced the quantities $\boldsymbol{T}_n = (\boldsymbol{C}_n \otimes \boldsymbol{C}_n)$ and $\boldsymbol{d}_n = \text{vec}(\boldsymbol{B}_n)$. We know from the assumptions in the lemma that $\boldsymbol{d}_n$ converges to $\boldsymbol{d} = \text{vec}(\boldsymbol{B})$ and $\boldsymbol{T}_n$ converges to $\boldsymbol{T} = (\boldsymbol{C} \otimes \boldsymbol{C})$, which is a stable matrix since $\boldsymbol{C}$ is stable. It follows that $\text{vec}(\boldsymbol{Z}_n)$ converges to $\text{vec}(\boldsymbol{Z}) = (\boldsymbol{I} - \boldsymbol{C} \otimes \boldsymbol{C})^{-1}\boldsymbol{d}$, which establishes that the limiting $\boldsymbol{Z}$ is the solution to the Lyapunov equation (1.149). We can establish this convergence result more formally as follows by adjusting the proof given for Theorem 1 in App. E of [12]. Let $z_n = \text{vec}(\boldsymbol{Z}_n)$, $z = \text{vec}(\boldsymbol{Z})$, $\boldsymbol{T}_n - \boldsymbol{T} = \epsilon\boldsymbol{\Delta}_n$, and $\boldsymbol{d}_n - \boldsymbol{d} = \epsilon\boldsymbol{\delta}_n$, for some arbitrary scalar $\epsilon > 0$ and quantities $\{\boldsymbol{\delta}_n, \boldsymbol{\Delta}_n\}$. Using $z = \boldsymbol{T}z + \boldsymbol{d}$ and $z_{n+1} = \boldsymbol{T}_n z_n + \boldsymbol{d}_n$, we get

$$z_{n+1} - z = \boldsymbol{T}(z_n - z) + \epsilon\boldsymbol{\Delta}_n(z_n - z) + \epsilon\boldsymbol{\Delta}_n z + \epsilon\boldsymbol{\delta}_n. \tag{1.151}$$

Now, since $\boldsymbol{T}$ is a stable matrix, there exists a sub-multiplicative matrix norm, denoted by $\|\cdot\|_\rho$, such that $\|\boldsymbol{T}\|_\rho = c < 1$ [76]. Using this norm, and the triangle inequality, we have

$$\|z_{n+1} - z\|_\rho \le \|\boldsymbol{T}\|_\rho \|z_n - z\|_\rho + \epsilon\|\boldsymbol{\Delta}_n\|_\rho \|z_n - z\|_\rho + \epsilon\|\boldsymbol{\Delta}_n\|_\rho \|z\|_\rho + \epsilon\|\boldsymbol{\delta}_n\|_\rho. \tag{1.152}$$

In the limit, as $n \to \infty$, we can choose $\|\boldsymbol{\Delta}_n\|_\rho \le 1$ and $\|\boldsymbol{\delta}_n\|_\rho \le 1$, which implies that

$$\|z_{n+1} - z\|_\rho \le (\|\boldsymbol{T}\|_\rho + \epsilon)\|z_n - z\|_\rho + \epsilon(\|z\|_\rho + 1), \text{ as } n \to \infty. \tag{1.153}$$

But since $\|\boldsymbol{T}\|_\rho < 1$, we can select ϵ small enough such that $\|\boldsymbol{T}\|_\rho + \epsilon < 1$ and, hence,

$$\lim_{n\to\infty} \|z_{n+1} - z\|_\rho = \epsilon(\|z\|_\rho + 1)/(1 - \|\boldsymbol{T}\|_\rho - \epsilon). \tag{1.154}$$

Since ϵ can be chosen arbitrarily small, it follows that $\|z_{n+1} - z\| \to 0$ as $n \to \infty$.

■

Lemma 2 (Stability of $\mathcal{F}$). *For any left-stochastic matrix* $\mathcal{A}$ *and block diagonal matrix* $\mathcal{D}$ *with the* $2-$*induced norms of its blocks strictly bounded by one, it holds that the matrix product* $\mathcal{B} = \mathcal{A}^T\mathcal{D}$ *is stable. Consequently, when* $\|(\boldsymbol{I}_N - \boldsymbol{P}_k^+\boldsymbol{S}_k)\boldsymbol{F}\|_2 < 1$,

the matrix $\mathcal{F}$ defined below is stable:

$$\mathcal{F} \triangleq \mathcal{A}^T\mathcal{X}, \quad \mathcal{X} \triangleq (\boldsymbol{I}_{KN} - \mathcal{P}^+\mathcal{S})(\boldsymbol{I}_K \otimes \boldsymbol{F}). \tag{1.155}$$

Proof: This argument adjusts the proof given for Lemma 2 in [12], where the $\rho-$norm should be replaced by the block-maximum norm defined as follows [77]. Let $\boldsymbol{x} = \text{col}\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K\}$ denote a $K \times 1$ *block* column vector whose individual entries are themselves vectors of size $N \times 1$ each. The block maximum norm of $\boldsymbol{x}$ is denoted by $\|\boldsymbol{x}\|_{b,\infty}$ and is defined as $\|\boldsymbol{x}\|_{b,\infty} = \max_{1\le k\le K} \|\boldsymbol{x}_k\|$. This vector norm induces a block-maximum matrix norm. Let $\mathcal{A}$ denote an arbitrary $K \times K$ block matrix with individual block entries of size $N \times N$ each. Then, the block-maximum norm of $\mathcal{A}$ is defined as

$$\|\mathcal{A}\|_{b,\infty} \triangleq \max_{\boldsymbol{x}\neq 0} \frac{\|\mathcal{A}\boldsymbol{x}\|_{b,\infty}}{\|\boldsymbol{x}\|_{b,\infty}}. \tag{1.156}$$

The block-maximum norm has several useful properties — see [77]. In particular, when $\boldsymbol{A}$ is $K \times K$ left-stochastic and $\mathcal{A} = \boldsymbol{A} \otimes \boldsymbol{I}_N$, then it can be verified that $\|\mathcal{A}^{\mathrm{T}}\|_{b,\infty} = 1$. Likewise, when $\mathcal{D}$ is block diagonal with the $2-$induced norm of its diagonal blocks bounded by one, it is easy to check that $\|\mathcal{D}\|_{b,\infty} < 1$. Consequently, for $\mathcal{B} = \mathcal{A}^{\mathrm{T}}\mathcal{D}$ we get

$$\rho(\mathcal{B}) \le \|\mathcal{B}\|_{b,\infty} \le \|\mathcal{A}^{\mathrm{T}}\|_{b,\infty}\|\mathcal{D}\|_{b,\infty} < 1, \tag{1.157}$$

and we conclude that $\mathcal{B}$ is stable. Alternatively, we note that

$$\|\mathcal{B}^n\|_{b,\infty} \le \left(\|\mathcal{A}^{\mathrm{T}}\|_{b,\infty}\right)^n (\|\mathcal{D}\|_{b,\infty})^n \longrightarrow 0, \text{ as } n \to \infty. \tag{1.158}$$

The matrix $\mathcal{F}$ in (1.155) has a form that fits into this formulation with $\mathcal{D} = (\boldsymbol{I}_{KN} - \mathcal{P}^+\mathcal{S})(\boldsymbol{I}_K \otimes \boldsymbol{F}) = \mathcal{X}$, which is block diagonal. Moreover, since by assumption $\|(\boldsymbol{I}_{KN} - \boldsymbol{P}_k^+\boldsymbol{S}_k)\boldsymbol{F}\|_2 < 1$, it holds that $\|\mathcal{D}\|_{b,\infty} = \|\mathcal{X}\|_{b,\infty} < 1$.

■

REFERENCES

1. Zhu Y, You Z, Zhao J, Zhang K, Li XR, The optimality for the distributed Kalman filtering fusion with feedback. Automatica 2001; 37(9):1489–1493.
2. Coates M, Distributed particle filters for sensor networks. In: Proceedings of the 3rd ACM/IEEE International Symposium on Information Processing in Sensor Networks, 2004, pp. 99–107.
3. Spanos DP, Olfati-Saber R, Murray RM, Approximate distributed Kalman filtering in sensor networks with quantifiable performance. In: Proceedings of the 4th ACM/IEEE International Symposium on Information Processing in Sensor Networks, 2005, pp. 133–139.
4. Ribeiro A, Giannakis GB, Roumeliotis SI, SOI-KF: Distributed Kalman filtering with low-cost communications using the sign of innovations. IEEE Transactions on Signal Processing 2006; 54(12):4782–4795.

5. Olfati-Saber R, Distributed Kalman filtering for sensor networks. In: 46th IEEE Conference on Decision and Control, 2007, pp. 5492–5498.
6. Khan UA, Moura JM, Distributing the Kalman filter for large-scale systems. IEEE Transactions on Signal Processing 2008; 56(10):4919–4935.
7. Cattivelli FS, Lopes CG, Sayed AH, Diffusion strategies for distributed Kalman filtering: Formulation and performance analysis. In: Proceedings of the IEEE IAPR Workshop on Cognitive Information Processing, 2008, pp. 36–41.
8. Cattivelli FS, Sayed AH, Diffusion mechanisms for fixed-point distributed Kalman smoothing. In: 16th European Signal Processing Conference, 2008, pp. 1–5.
9. Carli R, Chiuso A, Schenato L, Zampieri S, Distributed Kalman filtering based on consensus strategies. IEEE Journal on Selected Areas in Communications 2008; 26(4):622–633.
10. Cattivelli F, Sayed AH, Diffusion distributed Kalman filtering with adaptive weights. In: Proceedings of Asilomar Conference on Signals, Systems and Computers, 2009, pp. 908–912.
11. Olfati-Saber R, Kalman-consensus filter: Optimality, stability, and performance. In: Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference, 2009, pp. 7036–7042.
12. Cattivelli FS, Sayed AH, Diffusion strategies for distributed Kalman filtering and smoothing. IEEE Transactions on Automatic Control 2010; 55(9):2069–2084.
13. Hlinka O, Slučiak O, Hlawatsch F, Djurić PM, Rupp M, Likelihood consensus and its application to distributed particle filtering. IEEE Transactions on Signal Processing 2012; 60(8):4334–4349.
14. Djurić PM, Wang Y, Distributed Bayesian learning in multiagent systems: Improving our understanding of its capabilities and limitations. IEEE Signal Processing Magazine 2012; 29(2):65–76.
15. Sayed AH, Adaptation, learning, and optimization over networks. Foundations and Trends® in Machine Learning 2014; 7(4-5):311–801.
16. Zhao F, Guibas LJ, Wireless Sensor Networks: An Information Processing Approach. Amsterdam, The Netherlands: Morgan Kaufmann, 2004.
17. Hlinka O, Hlawatsch F, Djurić PM, Distributed particle filtering in agent networks: A survey, classification, and comparison. IEEE Signal Processing Magazine 2013; 30(1):61–81.
18. Sayed AH, Adaptive networks. Proceedings of the IEEE 2014; 102(4):460–497.
19. Anderson BDO, Moore JB, Optimal Filtering. Englewood Cliffs, NJ: Prentice Hall, 1979.
20. Arulampalam MS, Maskell S, Gordon N, Clapp T, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing 2002; 50(2):174–188.
21. Li XR, Jilkov VP, Survey of maneuvering target tracking. Part I. Dynamic models. IEEE Transactions on Aerospace and Electronic Systems 2003; 39(4):1333–1364.
22. Tu SY, Sayed AH, Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks. IEEE Transactions on Signal Processing 2012; 60(12):6217–6234.
23. Towfic ZJ, Chen J, Sayed AH, Excess-risk of distributed stochastic learners. IEEE Transactions on Information Theory 2016; 62(10):5753–5785.
24. Xiao L, Boyd S, Lall S, A scheme for robust distributed sensor fusion based on average consensus. In: Proceedings of the 4th ACM/IEEE International Symposium on Information Processing in Sensor Networks, Los Angeles, CA, 2005, pp. 63–70.
25. Olfati-Saber R, Fax JA, Murray RM, Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE 2007; 95(1):215–233.
26. Aysal TC, Yildiz ME, Sarwate AD, Scaglione A, Broadcast gossip algorithms for consensus.

IEEE Transactions on Signal Processing 2009; 57(7):2748–2761.
27. Dimakis AG, Kar S, Moura JMF, Rabbat MG, Scaglione A, Gossip algorithms for distributed signal processing. Proceedings of the IEEE 2010; 98(11):1847–1864.
28. Kailath T, Sayed AH, Hassibi B, Linear Estimation. Prentice Hall Upper Saddle River, NJ, 2000.
29. Sayed AH, Adaptive Filters. Wiley, 2008.
30. Chang KC, Saha RK, Bar-Shalom Y, On optimal track-to-track fusion. IEEE Transactions on Aerospace and Electronic Systems 1997; 33(4):1271–1276.
31. Atherton DP, Bather JA, Briggs AJ, Data fusion for several Kalman filters tracking a single target. IEE Proceedings – Radar, Sonar and Navigation 2005; 152(5):372–376.
32. Mitchell HB, Multi-sensor Data Fusion: An Introduction. Springer, 2007.
33. Julier SJ, Uhlmann JK, A non-divergent estimation algorithm in the presence of unknown correlations. In: Proceedings of the American Control Conference, vol. 4, vol. 4, 1997, pp. 2369–2373.
34. Uhlmann JK, Covariance consistency methods for fault-tolerant distributed data fusion. Information Fusion 2003; 4(3):201–215.
35. Hu J, Xie L, Zhang C, Diffusion Kalman filtering based on covariance intersection. IEEE Transactions on Signal Processing 2012; 60(2):891–902.
36. Kay SM, Fundamentals of Statistical Signal Processing: Estimation Theory. Upper Saddle River, NJ: Prentice Hall, 1993.
37. Djurić PM, Khan M, Johnston DE, Particle filtering of stochastic volatility modeled with leverage. IEEE Journal of Selected Topics in Signal Processing 2012; 6(4):327–336.
38. Hlinka O, Hlawatsch F, Distributed particle filtering in the presence of mutually correlated sensor noises. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6269–6273.
39. Moldaschl M, Gansterer WN, Hlinka O, Meyer F, Hlawatsch F, Distributed decorrelation in sensor networks with application to distributed particle filtering. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2014, pp. 6117–6121.
40. Urteaga I, Djurić PM, Sequential estimation of hidden ARMA processes by particle filtering: Part I. IEEE Transactions on Signal Processing 2017; 65(2):482–493.
41. Urteaga I, Djurić PM, Sequential estimation of hidden ARMA processes by particle filtering: Part II. IEEE Transactions on Signal Processing 2017; 65(2):494–504.
42. Tanizaki H, Nonlinear Filters: Estimation and Applications. Berlin, Germany: Springer, 1996.
43. Alspach D, Sorenson H, Nonlinear Bayesian estimation using Gaussian sum approximations. IEEE Transactions on Automatic Control 1972; 17(4):439–448.
44. Julier SJ, Uhlmann JK, Unscented filtering and nonlinear estimation. Proceedings of the IEEE 2004; 92(3):401–422.
45. van der Merwe R, Sigma-point Kalman filters for probabilistic inference in dynamic state-space models. Ph.D. thesis, OGI School of Science and Engineering, Oregon Health and Science University, Hillsboro, OR, 2004.
46. Arasaratnam I, Haykin S, Cubature Kalman filters. IEEE Transactions on Automatic Control 2009; 54(6):1254–1269.
47. Gordon NJ, Salmond DJ, Smith AFM, Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings F (Radar and Signal Processing) 1993; 140(2):107–113.
48. Djurić PM, Kotecha JH, Zhang J, Huang Y, Ghirmai T, Bugallo MF, et al., Particle filtering. IEEE Signal Processing Magazine 2003; 20(5):19–38.

49. Cappé O, Godsill SJ, Moulines E, An overview of existing methods and recent advances in sequential Monte Carlo. Proceedings of the IEEE 2007; 95(5):899–924.
50. Bugallo MF, Elvira V, Martino L, Luengo D, Miguez J, Djurić PM, Adaptive importance sampling: The past, the present, and the future. IEEE Signal Processing Magazine 2017; 34(4):60–79.
51. Li T, Bolic M, Djurić PM, Resampling methods for particle filtering: Classification, implementation, and strategies. IEEE Signal Processing Magazine 2015; 32(3):70–86.
52. Oreshkin BN, Coates MJ, Asynchronous distributed particle filter via decentralized evaluation of Gaussian products. In: Proceedings of FUSION, 2010.
53. Farahmand S, Roumeliotis SI, Giannakis GB, Set-membership constrained particle filter: Distributed adaptation for sensor networks. IEEE Transactions on Signal Processing 2011; 59(9):4122–4138.
54. Üstebay D, Coates M, Rabbat M, Distributed auxiliary particle filters using selective gossip. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2011, pp. 3296–3299.
55. Mohammadi A, Asif A, Consensus-based distributed unscented particle filter. In: Proceedings of IEEE Statistical Signal Processing Workshop, 2011, pp. 237–240.
56. Hlinka O, Hlawatsch F, Djurić PM, Consensus-based distributed particle filtering with distributed proposal adaptation. IEEE Transactions on Signal Processing 2014; 62(12):3029–3041.
57. Bordin CJ, Bruno MGS, Consensus-based distributed particle filtering algorithms for cooperative blind equalization in receiver networks. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2011, pp. 3968–3971.
58. Bandyopadhyay S, Chung SJ, Distributed estimation using Bayesian consensus filtering. In: Proceedings of American Control Conference, 2014, pp. 634–641.
59. Mohammadi A, Asif A, Distributed consensus + innovation particle filtering for bearing/range tracking with communication constraints. IEEE Transactions on Signal Processing 2015; 63(3):620–635.
60. Wang H, Djurić PM, Diffusion in networks by cooperative particle filtering. In: Proceedings of the IEEE Workshop on Computational Advances in Multi-sensor Adaptive Processing, 2017.
61. Yu JY, Coates MJ, Rabbat MG, Blouin S, A distributed particle filter for bearings-only tracking on spherical surfaces. IEEE Signal Processing Letters 2016; 23(3):326–330.
62. Li J, Nehorai A, Distributed particle filtering via optimal fusion of Gaussian mixtures. IEEE Transactions on Signal and Information Processing over Networks 2017; To be published.
63. Vázquez MA, Míguez J, A robust scheme for distributed particle filtering in wireless sensors networks. Signal Processing 2017; 131:190–201.
64. Savic V, Wymeersch H, Zazo S, Belief consensus algorithms for fast distributed target tracking in wireless sensor networks. Signal Processing 2014; 95:149–160.
65. Gautschi W, Orthogonal Polynomials: Computation and Approximation. Oxford, U.K.: Oxford University Press, 2004.
66. Unser M, Splines: A perfect fit for signal and image processing. IEEE Signal Processing Magazine 1999; 16(6):22–38.
67. Mosk-Aoyama D, Shah D, Fast distributed algorithms for computing separable functions. IEEE Transactions on Information Theory 2008; 54(7):2997–3007.
68. Xiao L, Boyd S, Fast linear iterations for distributed averaging. Systems & Control Letters 2004; 53(1):65–78.
69. Björck Å, Numerical Methods for Least Squares Problems. Philadelphia, PA: SIAM, 1996.

70. Lawson CL, Hanson RJ, Solving Least Squares Problems. Philadelphia, PA: SIAM, 1995.
71. Bishop CM, Pattern Recognition and Machine Learning. New York, NY: Springer, 2006.
72. Kotecha JH, Djurić PM, Gaussian particle filtering. IEEE Transactions on Signal Processing 2003; 51(10):2592–2601.
73. Bolić M, Athalye A, Hong S, Djurić PM, Study of algorithmic and architectural characteristics of Gaussian particle filters. Journal of Signal Processing Systems 2010; 61(2):205–218.
74. Gales MJF, Airey SS, Product of Gaussians for speech recognition. Computer Speech & Language 2006; 20(1):22–40.
75. Ito K, Xiong K, Gaussian filters for nonlinear filtering problems. IEEE Transactions on Automatic Control 2000; 45(5):910–927.
76. Horn RA, Johnson CR, Matrix Analysis. Cambridge University Press, 1990.
77. Sayed AH, Diffusion adaptation over networks. In: Chellapa R, Theodoridis S, editors, Academic Press Library in Signal Processing, vol. 3, vol. 3, Academic Press, Elsevier, 2014; pp. 323–454.