DISTRIBUTED AND ADAPTIVE TARGET TRACKING WITH A SENSOR NETWORK

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Michael A. Jacobs

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Daniel A. DeLaurentis, Chair

School of Aeronautics and Astronautics

Dr. Arthur E. Frazho

School of Aeronautics and Astronautics

Dr. Inseok Hwang

School of Aeronautics and Astronautics

Dr. Shreyas Sundaram

School of Electrical and Computer Engineering

Approved by:

Dr. Weinong Chen

Head of the School Graduate Program

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

SYMBOLS

State-Space Model

$\mathbf{y}$	Measurement
$\boldsymbol{\theta}$	Parameter
$\mathbf{x}$	State vector
$\mathbf{A}$	Process matrix
$\mathbf{q}$	Process noise
$\mathbf{Q}$	Process noise covariance
$\mathbf{H}$	Measurement matrix
$\mathbf{r}$	Measurement noise
$\mathbf{R}$	Measurement noise covariance

Estimates

$\hat{\mathbf{y}}$	Pseudo measurement
$\hat{\boldsymbol{\theta}}_{\text{MAP}}$	Maximum a posteriori parameter estimate
$\varphi(\boldsymbol{\theta})$	Energy function for the parameter as a design variable
$\frac{\partial \varphi(\boldsymbol{\theta})}{\partial \theta_j}$	Sensitivity equation for the j^{th} parameter as a design variable
$\bar{\mathbf{x}}$	*A priori* state estimate
$\bar{\mathbf{P}}$	*A priori* covariance estimate
$\widetilde{\mathbf{x}}$	*A posteriori* state estimate
$\widetilde{\mathbf{P}}$	*A posteriori* covariance estimate
$\widehat{\mathbf{x}}$	Smoothed state estimate
$\widehat{\mathbf{P}}$	Smoothed covariance estimate

Dimensions and Indices

n	Length of the state vector
m	Length of the measurement vector
N	Number of nodes or sensors in the network
i	Node or sensor $i \in [\,1\,,\,\cdots\,,\,N\,]$
D	Number of function calls in the parameter estimation
d	Time instance for the parameter estimation: $d \in [\,0\,,\,\cdots\,,\,D-1\,]$
T	Number of measurements
k	Time instance for the state estimation: $k \in [\,0\,,\,\cdots\,,\,T\,]$
L	Number of iterations in the consensus estimation
ℓ	Time instance for the consensus estimation: $\ell \in [\,0\,,\,\cdots\,,\,L\,]$

Graph Theory

$\mathcal{G}$	Graph
$\boldsymbol{\mathcal{V}}$	Set of nodes
$\mathcal{E}$	Set of directed edges
$\boldsymbol{\mathcal{D}}$	Degree matrix
$\boldsymbol{\mathcal{A}}$	Adjacency matrix
$\boldsymbol{\mathcal{L}}$	Laplacian matrix
$\boldsymbol{\mathcal{P}}$	Perron matrix
$\mathcal{N}(i)$	Neighbors of node i
ϵ	Consensus gain

Spaces

$\mathbb{R}^{n\times m}$	Real $n \times m$ matrix
$\mathbb{S}^{n\times m}$	Symmetric $n \times m$ matrix
$\mathbb{S}_{+}$	Symmetric and positive semidefinite matrix
$\mathbb{S}_{++}$	Symmetric and positive definite matrix

ABBREVIATIONS

Algorithms

ACKF	Adaptive Centralized Kalman Filter
ADKF	Adaptive Distributed Kalman Filter
AKF	Adaptive Kalman Filter
AGPR	Adaptive Gaussian Process Regression
CF	Consensus Filter
DKF	Distributed Kalman Filter
KCF	Kalman Consensus Filter
KF	Kalman Filter
GKCF	Generalized Kalman Consensus Filter
GP	Gaussian Process
GPR	Gaussian Process Regression
ICF	Information Consensus Filter
IF	Information Filter
IWCF	Information Weighted Consensus Filter
RTSS	Rauch-Tung-Striebel Smoother

Models

DSSM	Distributed state-space model
NCAM	Nearly Constant Acceleration Model
NCVM	Nearly Constant Velocity Model
PDSSM	Parameterized distributed state-space model
PSSM	Parameterized state-space model
SAM	Singer Acceleration Model
SSM	State-space model

Estimates

MAP	Maximum a posteriori
ML	Maximum likelihood
MMSE	Minimum mean squared error
MV	Minimum variance

Civil Aviation Terminology

ADS-B	Automatic Dependent Surveillance - Broadcast
CPI	Critical Pair Identification
EP	Entry point
EPU	Estimated position uncertainty
LoS	Loss of separation
NAC_P	Navigation accuracy category for position
NAS	National Airspace System
NextGen	Next Generation Air Transportation System
NMAC	Near mid-air collision
PSPC	Parameter sweep of a pairwise conflict
UAS	Unmanned Aerial System
WC	Well Clear

ABSTRACT

Jacobs, Michael A. Ph.D., Purdue University, May 2019. Distributed and Adaptive Target Tracking with a Sensor Network. Major Professor: Daniel A. DeLaurentis.

Ensuring the robustness and resilience of safety-critical systems from civil aviation to military surveillance technologies requires improvements to target tracking capabilities. Implementing target tracking as a distributed function can improve the quality and availability of information for end users. Any errors in the model of a target's dynamics or a sensor network's measurement process will result in estimates with degraded accuracy or even filter divergence. This dissertation solves a distributed estimation problem for estimating the state of a dynamical system and the parameters defining a model of that system. The novelty of this work lies in the ability of a sensor network to maintain consensus on state and parameter estimates through local communications between sensor platforms.

The system for the target dynamics and sensor network's measurement process was defined as a parameterized distributed state-space model (PDSSM). Two solutions were presented for this distributed state and parameter estimation problem: the Adaptive Centralized Kalman Filter (ACKF) and the Adaptive Distributed Kalman Filter (ADKF). The algorithms were derived in terms of state, parameter, and consensus estimation. These solutions were designed to highlight the difference between utilizing a Kalman Filter and a Distributed Kalman Filter for the state estimation at each sensor platform. An analysis was provided for the computational complexity, communication cost, optimality, stability, and simulation-based performance. The algorithms provided very similar results under nominal conditions, but the ADKF had a much high communication cost. Furthermore, the ADKF is likely to present a significant challenge to realize a communication network with a sufficient message size and

broadcast rate for target tracking applications. Then, the ACKF was implemented in a civil aviation simulation in order to demonstrate the capability for improving the safety of an airspace. The safety of an airspace was quantified by calculating the time until a future conflict will exist. The ACKF can improve the availability and quality of estimated aircraft state data, thus reducing the uncertainty an aircraft's safety assessment. The availability of state data refers to the degree to which aircraft and air traffic control stations can calculate state estimates of each aircraft for traffic management. Currently, the ACKF is the recommended algorithm over the ADKF based on the significantly smaller communication cost for calculating similar results. In order to make the ADKF a more robust solution than the ACKF, the parameter estimation routine can be altered at the expense of a higher communication cost.

1. INTRODUCTION

The future vision of civil aviation is for aircraft to participate as intelligent, cooperative agents within the air transportation network. Similar sentiments are being expressed for other civil applications such as Urban Air Mobility (UAM), Unmanned Aircraft System Traffic Management (UTM), and Intelligent Transportation System (ITS). Military applications include systems for tracking or intercepting ground, aerial, naval, and space targets. These systems depend upon a sensor network — a system of sensor platforms capable of measurement, computation, and communication. The potential advantages of employing a sensor network with fully distributed functions over a system with centralized functions are well documented and often include cost, robustness, resilience, accuracy, coverage, and scalability.

Distributed estimation is a fundamental problem of sensor networks, in which each sensor platform calculates an estimate while maintaining a consensus on that estimate throughout the network. Furthermore, each sensor platform can share information with neighboring sensor platforms to improve that estimate. A multi-sensor, multi-target system can require solving many subproblems concurrently (e.g., sensor-target allocation, data association, data discrimination, state estimation, parameter estimation, and fault mitigation) while maintaining consensus estimates. Many algorithms exist to solve these subproblems individually. However, can these algorithms be combined in a way that maintains the desirable performance characteristics such as optimality, stability, and scalability? Can the algorithms be decoupled without degrading the systems performance?

1.1 State, Parameter, and Consensus Estimation

This dissertation is concerned with state, parameter, and consensus estimation routines. More specifically, the problem consists of estimating the state of a dynamical system and the parameters defining a model of that system. The sensor platforms broadcast messages to neighboring sensor platforms with the intent of maintaining consensus on state and parameter estimates. This enables a sensor network to track the state of a target when the dynamics are unknown or incorrectly modeled. This distributed form of state and parameter estimation must address many concerns simultaneously to ensure the quality of the estimates. The concerns generally relate to optimality, stability, and scalability, which are well researched topics for each individual estimation problem. The difficulty of state, parameter, and consensus estimation is how to structure the algorithms properly while addressing these concerns.

The quality of a solution may be discussed in terms of the accuracy of the estimates and the rate of convergence of those estimates. A desirable outcome for the accuracy of an estimate is an unbiased estimate. A desirable outcome for the rate of convergence is a known upper bound that ensures convergence in the required amount of time. Consider these characteristics for a consensus estimation routine. Improper data discrimination and data fusion can result in the network calculating a biased value or filter divergence. Furthermore, quantized data and an out-of-sequence message (OOSM) — a delayed, missing, or faulty message — can also impact the accuracy and rate of convergence. The rate of convergence can be significantly degraded by a sparsely connected network topology. Errors in the consensus estimation can then propagate and degrade the quality of state and parameter estimates.

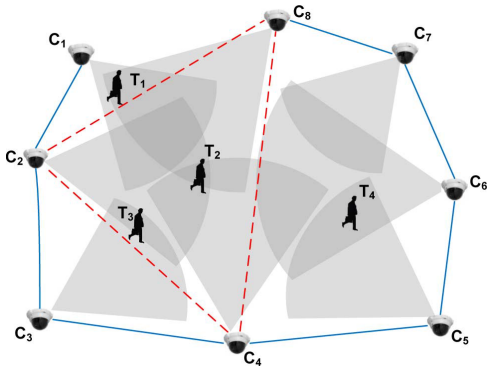
The scalability and tractability of a solution is determined by the communication and computational complexity as a function of the number of sensors, states, parameters, consensus routines, and measurements. The communication complexity characterizes the difficulty of maintaining consensus in terms of the network topology,

broadcast rate, and message size. The computational complexity characterizes the difficulty of the numerical computation at each node.

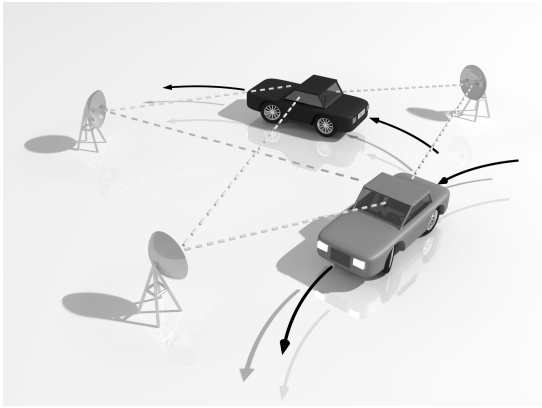
1.2 Problem Statement

The primary question of this dissertation is as follows: *how can a sensor network maintain consensus while estimating the state of a dynamical system and the parameters defining a model of that system?* Determining the target and measurement dynamics during state estimation can be a difficult problem. An intuitive example is tracking a target's position with a sensor network as shown with multiple examples in Fig. 1.1. In this case, state estimation for the target can be extremely difficult if the dynamics are unknown or the target maneuvers erratically. Errors in the dynamical model will result in estimates with degraded accuracy or even filter divergence.

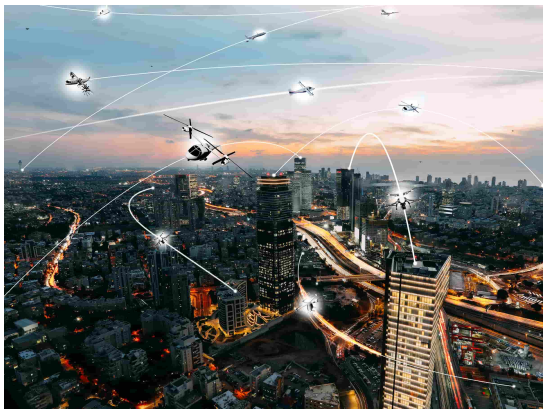
The sensor network must communicate through local interactions in a way that guarantees consensus on underlying information so that every sensor platform can calculate the same state and parameter estimates. A recent research area, the Distributed Kalman Filter (DKF), aims to provide a scalable and optimal solution to the distributed form of state estimation. Current literature on the DKF focuses on linear and nonlinear dynamical models, while adaptive models to address errors in the dynamics have scarcely been addressed. The performance characteristics in the previous subsection raises a secondary question that relates to the DKF research. *Can an adaptive formulation of a DKF guarantee desirable results for real-time implementation?* Attention needs to be given to scalable solutions for target tracking of ground, aerial, and space objects in safety-critical systems. A realistic expectation is a sensor network with sensor platforms that contain inactive sensors — sensors platforms that do not take measurements but participate in the consensus estimation routine by communicating with neighboring sensor platforms. For example, the sensor networks in Fig. (1.1) may have a sensor with a limited field of view or environmental obstructions that prevent a target's detection.



(a) Camera sensor network [1].



(b) Ground vehicle target tracking [2].



(c) Urban Air Mobility (UAM) [3].

(d) Unmanned Aerial System (UAS) Traffic Management (UTM) [4].

Figure 1.1. Examples of sensor networks in target tracking applications.

A state variable is an element of the vector $\mathbf{x}_k \in \mathbb{R}^n$ for representing the state of a system at a given time instance k. A parameter is an element of the vector $\boldsymbol{\theta} \in \mathbb{R}^p$ that differs from a state as a time-invariant variable. Sensor i can measure the state at time instance k as the measurement $\mathbf{y}_k^i \in \mathbb{R}^m$. The states form a Markov sequence, which is depicted as a Bayesian network in Fig. 1.2 in which the states and parameters are hidden, but the measurements are observed by N sensors. The objective is to maintain a consensus on state and parameter estimates throughout a sensor network with local communications.

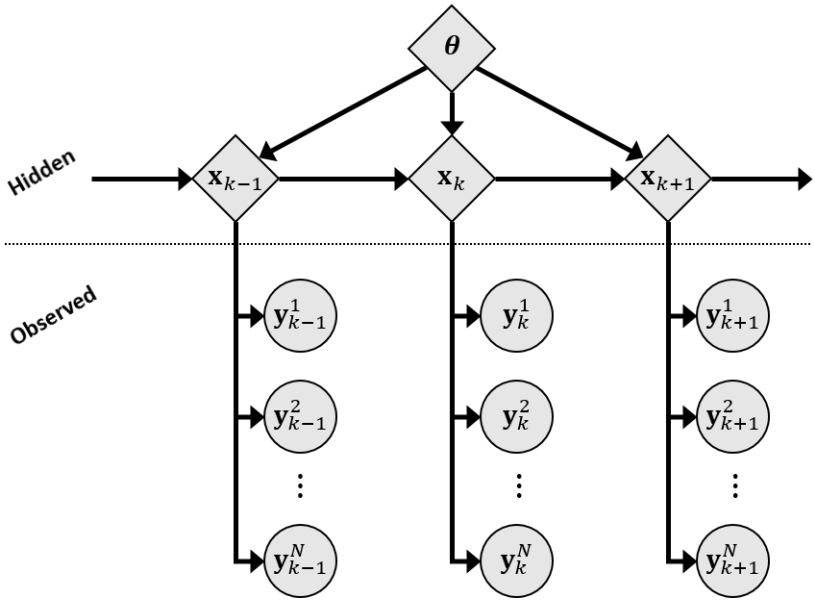


Figure 1.2. Bayesian network for hidden states and parameters which are observed with a sensor network.

1.3 Methodology

The objective is to formulate an Adaptive Distributed Kalman Filter (ADKF) for real-time target tracking from existing algorithms as shown in Fig. 1.3. The solutions of interest for state and consensus estimation are the Kalman Filter (KF) and the Consensus Filter (CF), respectively. The KF is a common algorithm for filtering noisy measurement in aerospace applications. The CF has a high communication cost compared to other consensus algorithms, but is a reliable method for achieving consensus in a communication network with faults Simultaneous state and consensus estimation has been a well researched topic over the past two decades with a DKF that is based on the KF and CF. Consequently, the chosen methodology for this dissertation considers adding an adaptive capability to a DKF.

Mehra identified in [5] four approaches to adaptive filtering: (a) Bayesian estimation, (b) maximum likelihood (ML) methods, (c) covariance-matching methods, and (d) correlation methods. The chosen method for the adaptive component in this

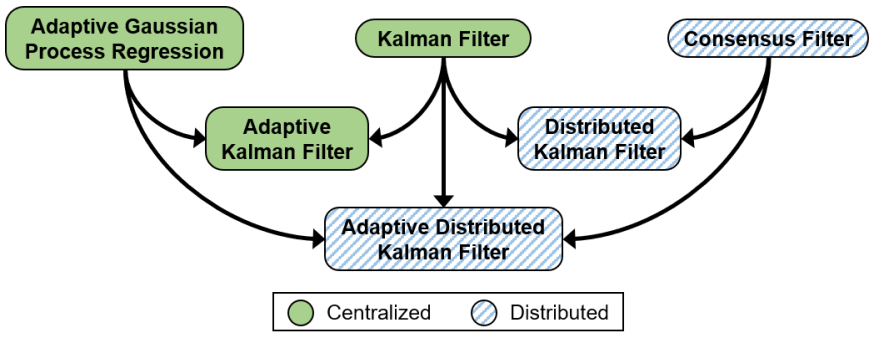


Figure 1.3. The relevant algorithms for constructing an Adaptive Distributed Kalman Filter.

work, the maximum a posteriori (MAP) parameter estimate, falls under (a) Bayesian estimation as a Bayesian point estimate. A state-space model (SSM) can be written as a function of parameters that can be optimized. This model is referred to as a parameterized state-space model (PSSM). To determine a desirable parameterization, a connection needs to be made to another common methodology for parameter estimation: Adaptive Gaussian Process Regression (AGPR). AGPR has seen a wide variety of applications as a machine learning methodology, which generally involves calculating the ML or MAP estimate of the parameters with a gradient-based optimization routine. However, the regression calculation has a high computational cost. Recent research has demonstrated that the covariance function as an underlying model for AGPR can be reformulated as a PSSM for an Adaptive Kalman Filter (AKF). This conversion can significantly reduce the computational cost and enable the use of machine learning models in addition to the typical target tracking models. This research considers a distributed formulation of this PSSM in order to provide flexibility for choosing a model.

1.4 Overview and Contributions of this Dissertation

Chapter 2 starts with a literature review of the centralized algorithms in Fig. 1.3: AGPR, KF, and AKF. This chapter addresses the relationship between dynamical models and machine learning models in target tracking applications. These models are compared individually for their application in target tracking. Then, their joint capability is discussed in the context of two example applications: fault detection of aircraft state data and tracking of a hypersonic boost-glide vehicle.

Chapter 3 starts with a literature review of the distributed algorithms: CF and DKF. The distributed estimation problem is then defined as a distributed parameterized state-space model (DPSSM). Then, the main contributions of this dissertation are presented, which starts with two solutions: the ADKF and the Adaptive Centralized Kalman Filter (ACKF). The algorithms consists of state, parameter, and consensus estimation routines which are derived in the context of Bayesian filters. These solutions are designed to enable a comparison between utilizing a DKF and KF for the state estimator. Then, the algorithms are characterized in terms of their performance characteristics: optimality, stability, computational complexity, communication complexity, and simulation-based performance.

Chapter 4 presents a safety motivation in civil aviation. With increasing automation in the National Airspace System (NAS), a comprehensive understanding of potentially faulty communication, navigation, and surveillance technologies is necessary to ensure safety. The distributed state and parameter estimation is demonstrated as a method to improve the availability and quality of estimated aircraft state data. The availability of state data refers to the degree to which aircraft and air traffic control stations can calculate state estimates of each aircraft for traffic management. The improvement in the quality of the state estimate translate to an improved safety assessment measured by the time until a future conflict will exist.

Lastly, Chapter 5 contains concluding remarks, highlights the contributions of this dissertation, and discusses potential future work.

1.4.1 Software for simulations

Simulation results are provided throughout the dissertation, which were generated with code in MATLAB. The code required the GPstuff toolbox[1] (version 4.7) from the Department of Computer Science at Aalto University. Although other GP computational tools exist, GPstuff was utilized for the incorporation of covariance functions that could be converted to state-space models. Another useful toolbox for optimal filters and smoothers is the EKF/UKF toolbox[2] (version 1.3) from the Department of Biomedical Engineering and Computational Science at Aalto University.

1.4.2 Mathematical notation

Scalars are denoted with italics while vectors and matrices are denoted with bold type. $\mathbb{R}$ denotes the set of real numbers, $\mathbb{R}_+$ denotes the set of nonnegative real numbers, $\mathbb{R}_{++}$ denotes the set of positive real numbers, $\mathbb{R}^n$ denotes the set of real $n \times 1$ vectors, and $\mathbb{R}^{n \times m}$ denotes the set of real $n \times m$ matrices. $\mathbb{S}^k$ denotes the set of symmetric $k \times k$ matrices, $\mathbb{S}^k_+$ denotes the set of symmetric positive semidefinite $k \times k$ matrices, and $\mathbb{S}^k_{++}$ denotes the set of symmetric positive definite $k \times k$ matrices. The zero vector is the column vector $\mathbf{0} = [\,0\,,\,\cdots\,,\,0\,]^\top$. The unit vector is the column vector $\mathbf{1} = [\,1\,,\,\cdots\,,\,1\,]^\top$. The identity matrix is a diagonal matrix where the main diagonal is all ones, which is denoted as $\mathbf{I} = \mathrm{Diag}(\,1\,,\,\cdots\,,\,1\,)$.

[1]For the GPstuff toolbox, see `http://research.cs.aalto.fi/pml/software/gpstuff`
[2]For the EKF/UKF toolbox, see `http://becs.aalto.fi/en/research/bayes/ekfukf/install.html`

2. SELECTION OF AN ADAPTIVE FILTER

A problem with statistical inference in target tracking is the incompatibility of models across different applications. Since the 1950's, target tracking matured from a research problem into a technology. However, many tracking algorithms were developed for aeronautics with many recent efforts exploring ground target tracking. Ground and aerial target tracking might appear comparable, but are actually quite different as discussed by Chong [6]. A quantitative and qualitative comparison was performed to demonstrate that existing aerial target tracking algorithms are unsuitable for use in ground target tracking. With this application problem in mind, this chapter focuses on selecting an adaptive filter and the underlying system model for a wide variety of target tracking applications.

For target tracking applications, an adaptive formulation of the Kalman Filter (KF) can address changes in the target dynamics with parameter adjustment, state augmentation, or multiple models [6]. As discussed by Bar-Shalom [7] and Li and Jilkov [8], many target tracking models exist with parameters that can be adjusted. However, a flexible framework for learning the dynamical process should include the optimization of unknown parameters in the target's dynamics in addition to the sensor's measurement process. The methodology of Adaptive Gaussian Process Regression (AGPR) is a machine learning technique for statistical inference that utilizes covariance functions with adaptable parameters. A set of common covariance functions exist and their underlying properties make them ideal for particular applications [9, 10]. A recent research area has demonstrated how these covariance functions can be converted into a parameterized state-space model (PSSM). Consequently, the chosen model for parameter adjustment in this dissertation is the PSSM that is compatible with the target tracking and machine learning models. This chapter introduces these concepts, compares the machine learning and target tacking models

for target tracking applications, and presents two example problems to demonstrate their potential applications.

2.1 Literature Review

Figure 2.1 depicts the relevant algorithms to construct an Adaptive Distributed Kalman Filter (ADKF) as a solution to the distributed state and parameter estimation problem. The grey region highlights the algorithms relevant to this chapter which focuses on an Adaptive Kalman Filter (AKF) In order to present the fundamental concepts, the literature review presents the AGPR and the KF.

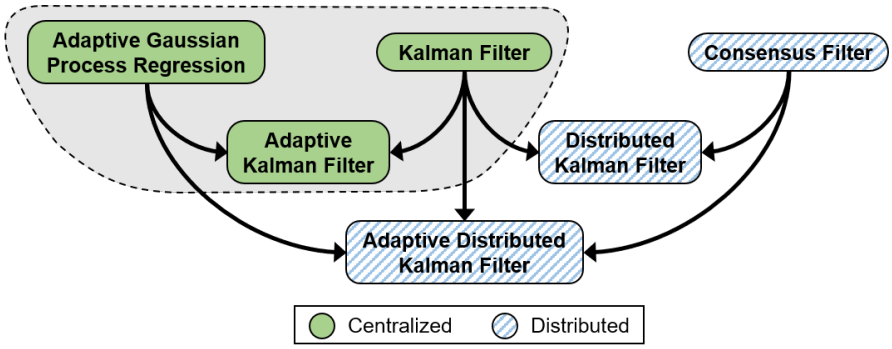


Figure 2.1. The grey region highlights the algorithms discussed within this chapter.

2.1.1 Kalman Filter

For state estimation, the Kalman Filter (KF) is a statistical inference algorithm for estimating the state of a noisy, linear dynamical system. However, each filtered state estimate is conditioned on the previous measurements. After filtering, the Rauch-Tung-Striebel Smoother (RTSS) can then condition every smoothed state estimate on every measurement. The KF and RTSS were originally published in [11] and [12],

respectively. See the book [2] by Särkkä for a discussion of filters and smoothers in the context of Bayesian inference.

Consider a dynamic process measured by a single sensor defined with the linear state-space model (SSM) as

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}\,, \qquad \mathbf{q}_{k-1} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{Q}_{k-1}\big)\,, \tag{2.1a}$$

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{r}_k\,, \qquad \mathbf{r}_k \sim \mathcal{N}\big(\mathbf{0}, \mathbf{R}_k\big)\,, \tag{2.1b}$$

or with the probabilistic notation as

$$p\big(\mathbf{x}_k \mid \mathbf{x}_{k-1}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \mathbf{A}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}\big)\,, \tag{2.2a}$$

$$p\big(\mathbf{y}_k \mid \mathbf{x}_k\big) = \mathcal{N}\big(\mathbf{y}_k \mid \mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k\big)\,, \tag{2.2b}$$

for the time instance $k \in [\,1\,, \cdots, T\,]$. For the process dynamics, $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector, $\mathbf{A}_k$ is the process matrix, and $\mathbf{q}_k$ is the process noise with covariance $\mathbf{Q}_k$. For the measurement model, $\mathbf{y}_k \in \mathbb{R}^m$ is the measurement vector, $\mathbf{H}_k$ is the measurement matrix, and $\mathbf{r}_k$ is the measurement noise with covariance $\mathbf{R}_k$. The noise terms are zero-mean white Gaussian noise such that

$$\mathbb{E}\big[\mathbf{q}_k(\mathbf{q}_\ell)^\top\big] = \mathbf{Q}_k\delta_{k\ell}\,, \tag{2.3a}$$

$$\mathbb{E}\big[\mathbf{r}_k(\mathbf{r}_\ell)^\top\big] = \mathbf{R}_k\delta_{k\ell}\,, \tag{2.3b}$$

where the Kronecker delta function $\delta_{k\ell}$ is defined as $\delta_{k\ell} = 1$ if $k = \ell$ and $\delta_{k\ell} = 0$ if $k \neq \ell$.

In Fig. 2.2, this Markov sequence is depicted as a Bayesian network where the states are hidden, but the measurements are observed. The KF makes a forward pass over the measurements in a recursive, two step process of predict and update.

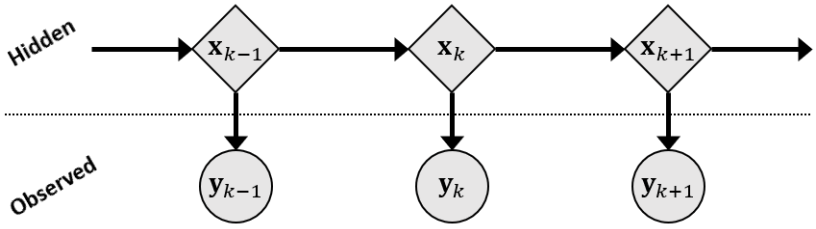


Figure 2.2. Bayesian network for the Markov sequence of the state-space model.

Then, a smoother can make a backward pass over the filtered results. The predictive, filtering, and smoothing distributions are denoted as

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \bar{\mathbf{x}}_k, \bar{\mathbf{P}}_k\big)\,, \tag{2.4a}$$

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \widetilde{\mathbf{x}}_k, \widetilde{\mathbf{P}}_k\big)\,, \tag{2.4b}$$

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:T}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \widehat{\mathbf{x}}_k, \widehat{\mathbf{P}}_k\big)\,. \tag{2.4c}$$

In the prediction step, the *a priori* statistics are

$$\bar{\mathbf{x}}_k = \mathbf{A}_{k-1}\widetilde{\mathbf{x}}_{k-1}\,, \tag{2.5a}$$

$$\bar{\mathbf{P}}_k = \mathbf{A}_{k-1}\widetilde{\mathbf{P}}_{k-1}\mathbf{A}_{k-1}^\top + \mathbf{Q}_{k-1}\,, \tag{2.5b}$$

where the state estimate is $\bar{\mathbf{x}}_k \in \mathbb{R}^n$ and covariance is $\bar{\mathbf{P}}_k \in \mathbb{R}^{n\times n}$. In the update step, the *a posteriori* statistics are

$$\mathbf{v}_k = \mathbf{y}_k - \mathbf{H}_k\bar{\mathbf{x}}_k\,, \tag{2.6a}$$

$$\mathbf{S}_k = \mathbf{H}_k\bar{\mathbf{P}}_k\mathbf{H}_k^\top + \mathbf{R}_k\,, \tag{2.6b}$$

$$\mathbf{K}_k = \bar{\mathbf{P}}_k\mathbf{H}_k^\top\mathbf{S}_k^{-1}\,, \tag{2.6c}$$

$$\widetilde{\mathbf{x}}_k = \bar{\mathbf{x}}_k + \mathbf{K}_k\mathbf{v}_k\,, \tag{2.6d}$$

$$\widetilde{\mathbf{P}}_k = \bar{\mathbf{P}}_k - \mathbf{K}_k\mathbf{H}_k\bar{\mathbf{P}}_k\,, \tag{2.6e}$$

where the state estimate is $\widetilde{\mathbf{x}}_k \in \mathbb{R}^n$ and covariance is $\widetilde{\mathbf{P}}_k \in \mathbb{R}^{n\times n}$. The smoothed statistics are

$$\mathbf{G}_k = \widetilde{\mathbf{P}}_k \mathbf{A}_k^\top \bar{\mathbf{P}}_{k+1}^{-1} , \tag{2.7a}$$

$$\widehat{\mathbf{x}}_k = \widetilde{\mathbf{x}}_k + \mathbf{G}_k \left[\widehat{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}_{k+1}\right] , \tag{2.7b}$$

$$\widehat{\mathbf{P}}_k = \widetilde{\mathbf{P}}_k + \mathbf{G}_k \left[\widehat{\mathbf{P}}_{k+1} - \bar{\mathbf{P}}_{k+1}\right] \mathbf{G}_k^\top , \tag{2.7c}$$

where the smoothed state estimate is $\widehat{\mathbf{x}}_k \in \mathbb{R}^n$ and covariance is $\widehat{\mathbf{P}}_k \in \mathbb{R}^{n\times n}$.

The recursion starts from the prior distribution denoted as

$$p\big(\mathbf{x}_0\big) = \mathcal{N}\big(\mathbf{x}_0 \mid \widetilde{\mathbf{x}}_0, \widetilde{\mathbf{P}}_0\big) . \tag{2.8}$$

The update step in Eq. (2.6) was derived with the Kalman gain $\mathbf{K}_k \in \mathbb{R}^{n\times n}$ that calculates the minimum mean squared error (MMSE) estimate:

$$\widetilde{\mathbf{x}}_{k,\mathrm{MMSE}} = \arg\min_{\widetilde{\mathbf{x}}_k} \mathbb{E}\big[\, \|\widetilde{\mathbf{x}}_k - \mathbf{x}_k\|^2 \,\big] . \tag{2.9}$$

For the exact model described in Eqs. (2.2) and (2.3), this formulation of the KF is optimal in the sense of MMSE. A common methodology to determine the quality of the regression involves the analysis of the residual vector $\mathbf{v}_k \in \mathbb{R}^m$ and residual covariance matrix $\mathbf{S}_k \in \mathbb{R}^{m\times m}$.

The Information Filter (IF) is an alternative formulation of the KF based on the inverse-covariance matrix. The transformations to calculate the *a priori* information vector $\bar{\mathbf{z}}_k \in \mathbb{R}^n$ and information matrix $\bar{\mathbf{Z}}_k \in \mathbb{R}^{n\times n}$ are

$$\bar{\mathbf{z}}_k = \bar{\mathbf{P}}_k^{-1} \bar{\mathbf{x}}_k , \tag{2.10a}$$

$$\bar{\mathbf{Z}}_k = \bar{\mathbf{P}}_k^{-1} . \tag{2.10b}$$

Similarly, the transformations to calculate the *a posteriori* information vector $\widetilde{\mathbf{z}}_k \in \mathbb{R}^n$ and information matrix $\widetilde{\mathbf{Z}}_k \in \mathbb{R}^{n \times n}$ are

$$\widetilde{\mathbf{z}}_k = \widetilde{\mathbf{P}}_k^{-1} \widetilde{\mathbf{x}}_k \,, \tag{2.11a}$$

$$\widetilde{\mathbf{Z}}_k = \widetilde{\mathbf{P}}_k^{-1} \,. \tag{2.11b}$$

In the prediction step, the *a priori* statistics are

$$\mathbf{M}_k = \left[\mathbf{A}_{k-1}^{\top}\right]^{-1} \widetilde{\mathbf{Z}}_{k-1} \mathbf{A}_{k-1}^{-1} \,, \tag{2.12a}$$

$$\mathbf{\Sigma}_k = \mathbf{M}_k + \mathbf{Q}_{k-1}^{-1} \,, \tag{2.12b}$$

$$\bar{\mathbf{Z}}_k = \mathbf{M}_k - \mathbf{M}_k \mathbf{\Sigma}_k^{-1} \mathbf{M}_k \,, \tag{2.12c}$$

$$\bar{\mathbf{z}}_k = \bar{\mathbf{Z}}_k \mathbf{A}_{k-1} \widetilde{\mathbf{Z}}_{k-1}^{-1} \widetilde{\mathbf{z}}_{k-1} \,. \tag{2.12d}$$

In the update step, the *a posteriori* statistics are

$$\widetilde{\mathbf{z}}_k = \bar{\mathbf{z}}_k + \mathbf{i}_k \,, \tag{2.13a}$$

$$\widetilde{\mathbf{Z}}_k = \bar{\mathbf{Z}}_k + \mathbf{I}_k \,, \tag{2.13b}$$

where the new information is

$$\mathbf{i}_k = \mathbf{H}_k^{\top} \mathbf{R}_k^{-1} \mathbf{y}_k \,, \tag{2.14a}$$

$$\mathbf{I}_k = \mathbf{H}_k^{\top} \mathbf{R}_k^{-1} \mathbf{H}_k \,. \tag{2.14b}$$

One of the main advantages of the IF over the KF is that the update step is simply an addition operation. Another advantageous property of the IF is that the initialization of the information matrix with zeros corresponds to infinite uncertainty in the initial state. The KF, IF, and RTSS pseudocode is provided in Appendix B as Algorithms 1, 2, and 3, respectively.

Example of a Kalman Filter and Rauch-Tung-Striebel Smoother

To demonstrate the implementation of the KF and RTSS, an example is provided for estimating the position of a target. The target dynamics were defined by the Nearly Constant Velocity Model (NCVM) [7]:

$$\mathbf{A}_k = \begin{bmatrix} 1 & \Delta t_k \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{Q}_k = q_c \begin{bmatrix} \frac{1}{3}(\Delta t_k)^3 & \frac{1}{2}(\Delta t_k)^2 \\ \frac{1}{2}(\Delta t_k)^2 & \Delta t_k \end{bmatrix}, \tag{2.15}$$

where Δt_k is a sampling rate and q_c is the power spectral density of the process noise. The state vector

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} & x_{2,k} \end{bmatrix}^\top \tag{2.16}$$

was composed of the position $x_{1,k}$ and velocity $x_{2,k}$. The measurement process for observing the position was defined with

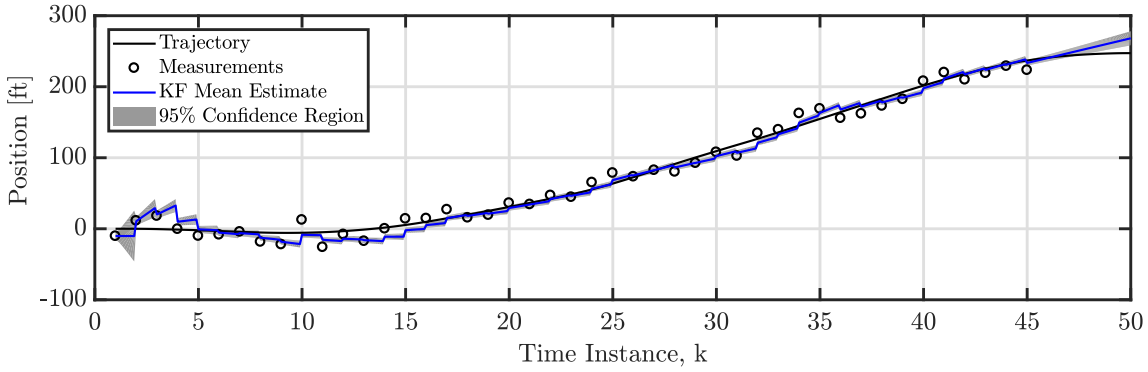
$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad \mathbf{R}_k = \sigma_n^2, \tag{2.17}$$

where σ_n is the standard deviation of the measurement noise. The data set in the example was a random sample of the NCVM with $\Delta t_k = 1$ s, $q_c = 0.1$ ft^2/s^3, and $\sigma_n = \sqrt{10}$ ft. The design choices for the solution are summarized in Table 2.1. The state estimates were computed with the KF and RTSS, which assumed perfect knowledge of the constants in the NCVM model.

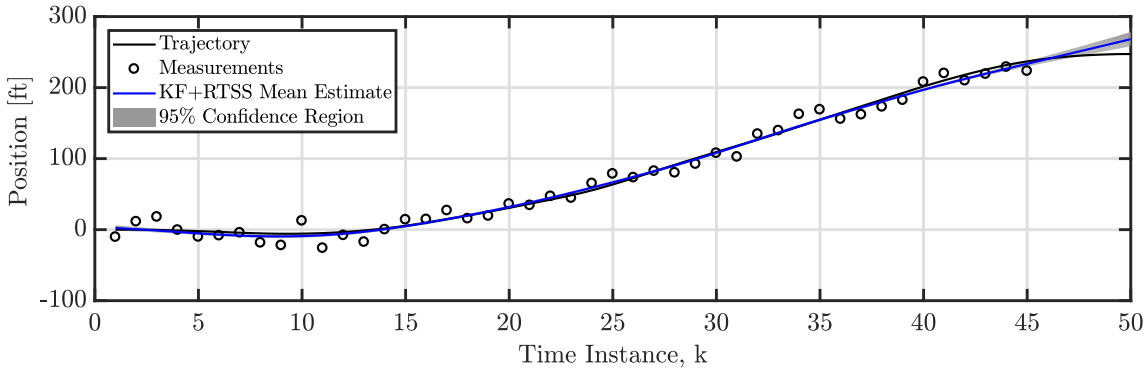
Table 2.1. The design choices for an example of a Kalman Filter and Rauch-Tung-Striebel Smoother.

(1) **State-Space Model** - NCVM
(2) **Constants** - $\Delta t = 1$ s , $q_c = 0.1$ ft^2/s^3 , and $\sigma_n = \sqrt{10}$ ft
(3) **State Estimation** - KF and RTSS

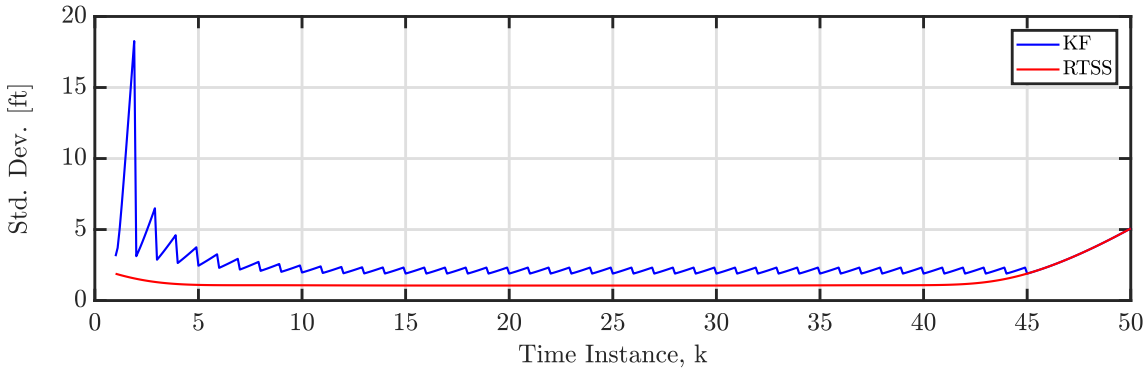
Figure 2.3 depicts the example of the filtering and smoothing process for generating state estimates. The black line is the target's true trajectory while the circles are the sensor's noisy measurements. The blue line is the estimated mean while the gray region is the 95% confidence region. In Fig. 2.3(a), the filtered estimates are conditioned only on the previous measurements. Consequently, the covariance estimate takes a few time instances to converge. After each update step, the uncertainty in the prediction increases with time until a new measurement is available resulting in the non-smooth prediction. In Fig. 2.3(b), the smoothed estimates are conditioned on all the measurements. The predicted estimates at the last measurement and in the future (i.e., $k \geq T$) are the same for the filtered and smoothed results. The prediction can become a poor predictor rather quickly if the target continues to maneuver erratically as in this example around the time instance $k = 50$. The convergence and prediction properties can also be analyzed by plotting the standard deviation of the estimates as shown in Fig. 2.3(c).



(a) Kalman Filter.



(b) Rauch-Tung-Striebel Smoother.



(c) Standard deviation.

Figure 2.3. State estimates for the Kalman Filter and Rauch-Tung-Striebel Smoother.

2.1.2 Adaptive Gaussian Process Regression

A Gaussian Process (GP) is a type of stochastic process while Gaussian Process Regression (GPR) calculates smoothed estimates with a non-parametric model given a set of measurements. In 1951, Danie Krige utilized GPR in his thesis for gold mine valuations based on limited samples. Eventually, the methodology found usage in geostatistics under the name Kriging. The potential applications for GPR are diverse and include the general topics of machine learning, spatial statistics, statistical inference, physical inverse problems, and signal processing. The methods popularity in a variety of domains is due to the effective modeling of nonlinear phenomena.

In the preeminent work [9] on GPs, Rasmussen and Williams loosely describe a GP as a generalization of the Gaussian distribution that can be thought of as a distribution over functions. More precisely, a GP is a random function $f(\mathbf{t}) \in \mathbb{R}$ with the input $\mathbf{t} \in \mathbb{R}^d$ in which any finite set of random variables has a joint Gaussian distribution. Here, the input is denoted with $\mathbf{t}$ to avoid confusion with the state $\mathbf{x}$ in the state-space model. The input $\mathbf{t}$ is often a temporal or spatio-temporal location. A GP is completely defined by the mean function $m(\mathbf{t}) \in \mathbb{R}$ and covariance function $k(\mathbf{t}, \mathbf{t}') \in \mathbb{R}$ denoted as

$$f(\mathbf{t}) \sim \mathcal{GP}(m(\mathbf{t}), k(\mathbf{t}, \mathbf{t}')) \,, \tag{2.18}$$

where

$$m(\mathbf{t}) = \mathbb{E}\big[f(\mathbf{t}) \big] \,, \tag{2.19a}$$

$$k(\mathbf{t}, \mathbf{t}') = \mathbb{E}\big[\left(f(\mathbf{t}) - m(\mathbf{t}) \right) \left(f(\mathbf{t}') - m(\mathbf{t}') \right) \big] \,. \tag{2.19b}$$

The covariance function encodes information about the spatial correlation between inputs. This area of research has resulted in a list of common covariance functions that include the squared exponential, neural network, and Matérn. Furthermore, the sum or product of two covariance functions results in covariance function.

The finite set of random variables has the joint Gaussian distribution

$$\begin{bmatrix} f(\mathbf{t}_1) \\ \vdots \\ f(\mathbf{t}_T) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{t}_1) \\ \vdots \\ m(\mathbf{t}_T) \end{bmatrix}, \begin{bmatrix} k(\mathbf{t}_1, \mathbf{t}_1) & \cdots & k(\mathbf{t}_1, \mathbf{t}_T) \\ \vdots & \ddots & \vdots \\ k(\mathbf{t}_T, \mathbf{t}_1) & \cdots & k(\mathbf{t}_T, \mathbf{t}_T) \end{bmatrix} \right) . \tag{2.20}$$

The collection of T input locations can be denoted as $\mathbf{t}_{1:T} = [\, \mathbf{t}_1 \,, \cdots \,, \mathbf{t}_T \,]$ where the k^{th} input location is $\mathbf{t}_k \in \mathbb{R}^d$. For simplicity, the equation is denoted as

$$\mathbf{f}(\mathbf{t}_{1:T}) \sim \mathcal{N}\big(\mathbf{M}(\mathbf{t}_{1:T}) \,, \mathbf{K}(\mathbf{t}_{1:T}, \mathbf{t}_{1:T})\big) \,, \tag{2.21}$$

where the random vector is $\mathbf{f}(\mathbf{t}_{1:T}) \in \mathbb{R}^T$, the mean vector is $\mathbf{M}(\mathbf{t}_{1:T}) \in \mathbb{R}^T$, and the covariance matrix is $\mathbf{K}(\mathbf{t}_{1:T}, \mathbf{t}_{1:T}) \in \mathbb{R}^{T \times T}$.

Consider the noisy measurement process defined by

$$y_k = f(\mathbf{t}_k) + r_k, \qquad r_k \sim \mathcal{N}\big(0, R\big) \,. \tag{2.22}$$

The set of measurements are denoted as $\mathbf{y}_{1:T} = [\, \mathbf{y}_1 \,, \cdots \,, \mathbf{y}_T \,]$ where the k^{th} measurement $y_k \in \mathbb{R}$ and r_k is the measurement noise with covariance R. The noise term is zero-mean white Gaussian noise such that

$$\mathbb{E}\big[r_k \, (r_\ell)^\top \big] = R \delta_{k\ell} \,, \tag{2.23}$$

where $\delta_{k\ell}$ is the Kronecker delta function.

The objective of Adaptive GPR (AGPR) is to train the model on a set of training data and then calculate predictions at target locations. More specifically, the training data consists of the measurements $\mathbf{y}_{1:T}$ and the input locations $\mathbf{t}_{1:T}$. Upon determining a sufficient model in the training phase, the predicted mean and covariance are calculated at the target locations $\mathbf{t}_{1:T_*}$. Note, there are T training locations and T_*

target locations. The remainder of the AGPR discussion adopts the more compact notation:

$$\begin{aligned} \mathbf{Y} &= \mathbf{y}_{1:T}\,, & \mathbf{K} &= \mathbf{K}(\mathbf{t}_{1:T}, \mathbf{t}_{1:T})\ , \\ \mathbf{T} &= \mathbf{t}_{1:T}\,, & \mathbf{K}_* &= \mathbf{K}(\mathbf{t}_{1:T}, \mathbf{t}_{1:T_*})\,, \\ \mathbf{T}_* &= \mathbf{t}_{1:T_*}\,, & \mathbf{K}_{**} &= \mathbf{K}(\mathbf{t}_{1:T_*}, \mathbf{t}_{1:T_*})\ . \end{aligned}$$

Training and prediction

Training involves tuning free parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ in the model to improve the quality of how the model fits the data. The parameters may be a combination of parameters from the mean function, the covariance function, or the noise. This work considers a zero-mean function such that the parameters are a combination of the latter two options.

An optimization routine may determine a parameter estimate by minimizing the expected loss (or maximizing the utility), such that

$$\mathbb{E}\big[C(\boldsymbol{\theta}, \mathbf{a}) \mid \mathbf{Y}\,,\, \mathbf{T}\big] = \int C(\boldsymbol{\theta}, \mathbf{a})\, p\big(\boldsymbol{\theta} \mid \mathbf{Y}, \mathbf{T}\big) \mathrm{d}\boldsymbol{\theta}\,, \tag{2.24}$$

where $\mathbf{a}$ is the action and C is the cost function. A common choice is the 0-1 loss function

$$C(\boldsymbol{\theta}, \mathbf{a}) = -\delta(\mathbf{a} - \boldsymbol{\theta})\ , \tag{2.25}$$

where δ is the Dirac delta function. For the 0-1 loss function, the optimal choice is the mode of the posterior distribution, which is the maximum a posteriori (MAP) estimate

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} \big[p\big(\boldsymbol{\theta} \mid \mathbf{Y}, \mathbf{T}\big)\big]\,. \tag{2.26}$$

Applying Bayes' theorem, the posterior distribution of the parameters $\boldsymbol{\theta}$ is

$$p\big(\boldsymbol{\theta} \mid \mathbf{Y}, \mathbf{T}\big) = \frac{p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big) p\big(\boldsymbol{\theta}\big)}{p\big(\mathbf{Y} \mid \mathbf{T}\big)}\,, \tag{2.27}$$

where the distributions are defined:

$$\text{posterior} \qquad p\big(\boldsymbol{\theta} \mid \mathbf{Y}, \mathbf{T}\big)\,, \tag{2.28a}$$

$$\text{likelihood} \qquad p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big)\,, \tag{2.28b}$$

$$\text{prior} \qquad p\big(\boldsymbol{\theta}\big)\,, \tag{2.28c}$$

$$\text{normalization term} \qquad p\big(\mathbf{Y} \mid \mathbf{T}\big)\,. \tag{2.28d}$$

The normalization term is independent of $\boldsymbol{\theta}$ and may be neglected as

$$p\big(\boldsymbol{\theta} \mid \mathbf{Y}, \mathbf{T}\big) \propto p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big) p\big(\boldsymbol{\theta}\big)\,. \tag{2.29}$$

An optimal parameter estimate can be calculated with the un-normalized marginal posterior instead of the posterior. The MAP estimate is the argument that minimizes the energy function φ_T such that

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \arg\min_{\boldsymbol{\theta}} \left[\varphi_T(\boldsymbol{\theta})\right]\,, \tag{2.30}$$

where the energy function is the negative log of the un-normalized marginal posterior

$$\varphi_T(\boldsymbol{\theta}) = -\log p\big(\boldsymbol{\theta}\big) - \log p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big)\,. \tag{2.31}$$

The substitution is valid as the maximum of the posterior and the minimum of the energy function both occur at the MAP estimate $\hat{\boldsymbol{\theta}}_{\mathrm{MAP}}$. The negative sign is preferred so that the resulting values may be thought of as penalties, which simply requires a switch from a maximization optimization problem to a minimization optimization problem. The logarithm does not change the MAP estimate, but is a more convenient domain for gradient-based optimization.

For the zero-mean GP and measurement process in Eq. (2.22), the likelihood is

$$p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})\big)\,, \tag{2.32}$$

where $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \mathbf{K}(\boldsymbol{\theta}) + R(\boldsymbol{\theta})\,\mathbf{I}$. Recall, an unknown parameter can be the noise magnitude $R(\boldsymbol{\theta}) = \sigma_n^2$ as well as parameters of the covariance function $\mathbf{K}(\boldsymbol{\theta})$. For gradient-based optimization, the derivatives of the energy function are known as the sensitivity equations defined as

$$\frac{\partial \varphi_T(\boldsymbol{\theta})}{\partial \theta_j} = -\frac{\partial \log p\big(\boldsymbol{\theta}\big)}{\partial \theta_j} - \frac{\partial \log p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big)}{\partial \theta_j}\,. \tag{2.33}$$

By evaluating the multivariate Gaussian distribution in Eq. (2.32), the log likelihood and its derivative with respect to parameter θ_j are

$$\log p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big) = -\frac{1}{2}\left(\mathbf{Y}^\top \boldsymbol{\alpha}(\boldsymbol{\theta}) + \log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| + T\log 2\pi\right)\,, \tag{2.34a}$$

$$\frac{\partial \log p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big)}{\partial \theta_j} = \frac{1}{2}\mathrm{Tr}\left(\left(\boldsymbol{\alpha}(\boldsymbol{\theta})\,[\boldsymbol{\alpha}(\boldsymbol{\theta})]^\top - [\boldsymbol{\Sigma}(\boldsymbol{\theta})]^{-1}\right)\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\right)\,, \tag{2.34b}$$

where $\mathrm{Tr}(\cdot)$ is the trace and

$$\boldsymbol{\alpha}(\boldsymbol{\theta}) = [\boldsymbol{\Sigma}(\boldsymbol{\theta})]^{-1}\,\mathbf{Y}\,. \tag{2.35}$$

In the log likelihood in Eq. (2.34a), the first and second terms may be interpreted as penalties for the data fit and model complexity (to avoid over-fitting), respectively. The predictive distribution is

$$p\big(\mathbf{f}_* \mid \mathbf{Y}, \mathbf{T}, \mathbf{T}_*, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{f}_* \mid \bar{\mathbf{f}}_*(\boldsymbol{\theta})\,, \mathrm{Cov}(\mathbf{f}_*(\boldsymbol{\theta}))\big)\,, \tag{2.36}$$

where the predicted mean and covariance are

$$\bar{\mathbf{f}}_*(\boldsymbol{\theta}) = [\mathbf{K}_*(\boldsymbol{\theta})]^\top\,\boldsymbol{\alpha}(\boldsymbol{\theta})\,, \tag{2.37a}$$

$$\mathrm{Cov}(\mathbf{f}_*(\boldsymbol{\theta})) = \mathbf{K}_{**}(\boldsymbol{\theta}) - [\mathbf{K}_*(\boldsymbol{\theta})]^\top\,[\boldsymbol{\Sigma}(\boldsymbol{\theta})]^{-1}\,\mathbf{K}_*(\boldsymbol{\theta})\,. \tag{2.37b}$$

Notice that the training requires calculating $\boldsymbol{\alpha}(\boldsymbol{\theta})$ for the log likelihood and its derivative in Eq. (2.34), but not the entire predictive distribution.

In order to reduce the computational complexity and increase the numerical stability, the AGPR is typically implemented with the Cholesky decomposition

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \mathbf{L}(\boldsymbol{\theta}) \left[\mathbf{L}(\boldsymbol{\theta})\right]^{\top} , \tag{2.38}$$

where $\mathbf{L}(\boldsymbol{\theta})$ is a lower triangular matrix. Then, the log likelihood and its derivative with respect to each parameter are

$$\log p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big) = -\frac{1}{2}\mathbf{Y}^{\top}\boldsymbol{\alpha}(\boldsymbol{\theta}) - \sum_{i} \log L_{ii}(\boldsymbol{\theta}) - \frac{T}{2}\log 2\pi \,, \tag{2.39a}$$

$$\frac{\partial \log p\big(\mathbf{Y} \mid \mathbf{T}, \boldsymbol{\theta}\big)}{\partial \theta_j} = \frac{1}{2}\mathrm{Tr}\left(\left(\boldsymbol{\alpha}(\boldsymbol{\theta})\left[\boldsymbol{\alpha}(\boldsymbol{\theta})\right]^{\top} - \left[\boldsymbol{\Sigma}(\boldsymbol{\theta})\right]^{-1}\right)\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_j}\right) , \tag{2.39b}$$

where θ_j is the j^{th} element of $\boldsymbol{\theta}$. The predictive distribution is then defined as

$$\mathbf{L}(\boldsymbol{\theta}) = \text{Cholesky}(\boldsymbol{\Sigma}(\boldsymbol{\theta})) \,, \tag{2.40a}$$

$$\boldsymbol{\alpha}(\boldsymbol{\theta}) = \left[\mathbf{L}(\boldsymbol{\theta})\right]^{-\top}\left[\mathbf{L}(\boldsymbol{\theta})\right]^{-1}\mathbf{Y} \,, \tag{2.40b}$$

$$\bar{\mathbf{f}}_*(\boldsymbol{\theta}) = \left[\mathbf{K}_*(\boldsymbol{\theta})\right]^{\top}\boldsymbol{\alpha}(\boldsymbol{\theta}) \,, \tag{2.40c}$$

$$\boldsymbol{\beta}(\boldsymbol{\theta}) = \left[\mathbf{L}(\boldsymbol{\theta})\right]^{-1}\mathbf{K}_*(\boldsymbol{\theta}) \,, \tag{2.40d}$$

$$\mathrm{Cov}(\mathbf{f}_*(\boldsymbol{\theta})) = \mathbf{K}_{**}(\boldsymbol{\theta}) - \left[\boldsymbol{\beta}(\boldsymbol{\theta})\right]^{\top}\boldsymbol{\beta}(\boldsymbol{\theta}) \,, \tag{2.40e}$$

where Cholesky$(\cdot)$ is the Cholesky decomposition. For a more thorough explanation of these and related topics, see the seminal work [9] by Rasmussen and Williams.

Example of Adaptive Gaussian Process Regression

To demonstrate the implementation of AGPR, an example is provided for estimating the atmospheric carbon dioxide (CO_2) readings at the Mauna Loa Observatory. The average monthly measurements of CO_2 contain seemingly unpredictable short- and long-term variations, an exponential rise, and periodic trends. The av-

erage monthly measurements[1] were compiled by Dr. Pieter Tans of NOAA Earth System Research Lab (ESRL) and Dr. Ralph Keeling of the Scripps Institution of Oceanography (SIO). The measurements are reported in parts per million (PPM). This problem is a common demonstration for statistical inference and was presented with AGPR by Rasmussen in [9]. The following example presents the AGPR method, but with a different covariance function. The example was programmed with GPstuff — a MATLAB-based toolbox developed at Aalto University [13].

The system was modeled with the zero-mean GP and the noisy measurement process in Eq. (2.22). The design choices for the AGPR are summarized in Table 2.2. The measurement model had additive Gaussian noise with covariance $R(\boldsymbol{\theta}) = \sigma_{\mathrm{n}}^2$. The covariance function was the addition of the squared exponential, periodic, and Matérn ($\nu = 5/2$) covariance functions

$$k(t_i, t_j) = k_{\mathrm{SE}}(t_i, t_j) + k_{\mathrm{P}}(t_i, t_j) + k_{\mathrm{M52}}(t_i, t_j) \ , \tag{2.41a}$$

$$k_{\mathrm{SE}}(t_i, t_j) = \sigma_{\mathrm{SE}}^2 \exp\left(-\frac{\tau^2}{2\ell_{\mathrm{SE}}^2}\right) \ , \tag{2.41b}$$

$$k_{\mathrm{P}}(t_i, t_j) = \sigma_{\mathrm{P}}^2 \exp\left(-\frac{2\sin^2\left(\pi\left(t_i - t_j\right)\right)}{\ell_{\mathrm{P}}^2} - \frac{\tau^2}{2\ell_{\mathrm{P,SE}}^2}\right) \ , \tag{2.41c}$$

$$k_{\mathrm{M52}}(t_i, t_j) = \sigma_{\mathrm{M52}}^2 \left(1 + \frac{\sqrt{5}\tau}{\ell_{\mathrm{M52}}} + \frac{5\tau^2}{3\ell_{\mathrm{M52}}^2}\right) \exp\left(-\frac{\sqrt{5}\tau}{\ell_{\mathrm{M52}}}\right) \ , \tag{2.41d}$$

where $\tau = |t_i - t_j|$. The Matérn covariance function contains a polynomial that can account for a non-zero mean. The unknown parameters are the noise magnitude σ_n^2 and the parameters in the covariance function which are σ^2 for a magnitude and ℓ for a length-scale. The parameter optimization routine was the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which provided the adaptive component for the AGPR. The quasi-Newton BFGS algorithm is a good gradient-based routine for solving non-linear optimization problems. A log-uniform prior distribution was placed on each parameter.

[1]The data is available at `https://www.esrl.noaa.gov/gmd/ccgg/trends/data.html`.

Table 2.2. The design choices for an example of Adaptive Gaussian Process Regression.

(1) **Covariance Function** Squared Exponential, Periodic, and Matérn ($\nu = 5/2$)
(2) **State Estimation** - GPR
(3) **Parameter Optimization** - Quasi-Newton BFGS
(4) **Parameters** $\boldsymbol{\theta} = [\, \sigma_n^2 \,,\, \ell_{\mathrm{SE}} \,,\, \sigma_{\mathrm{SE}}^2 \,,\, \ell_{\mathrm{P}} \,,\, \sigma_{\mathrm{P}}^2 \,,\, \ell_{\mathrm{P,SE}} \,,\, \ell_{\mathrm{M52}} \,,\, \sigma_{\mathrm{M52}}^2 \,]$
(5) **Initial Values for Parameters** $\boldsymbol{\theta}_0 = [\, 1 \,,\; 100 \,,\; 5000 \,,\; 1 \,,\; 1 \,,\; 100 \,,\; 10 \,,\; 10 \,]$
(6) **Prior Distributions** - Log-Uniform Distributions $p\big(\log \theta_j\big) \propto 1 \quad \forall j$

The example results are displayed in Fig. 2.4. The markers represent yearly measurements from 1959 to 2006. The gray region is the 95% confidence region of the prediction with the MAP parameter estimates. The final values for the parameters were

$$\begin{aligned} \boldsymbol{\theta}_* = [\, &0.046015 \,,\; 116.29 \,,\; 1.3828\text{e+}05 \,,\; 1.3561 \,, \\ &6.3212 \,,\; 103.13 \,,\; 0.6578 \,,\; 0.29099 \,] \,, \end{aligned} \tag{2.42}$$

and a reduction of the objective function from 640.295 to 136.753 in 47 function calls. An iteration is one step of the quasi-Newton BFGS algorithm while a function count is the number of calls to the energy function and sensitivity equations. These functions may be called multiple times in an iteration as the quasi-Newton BFGS algorithm determines the next step. A component-based analysis of the covariance function is shown in Fig. 2.5 in order to understand the impact of each additive covariance function. The squared exponential, periodic, and Matérn ($\nu = 5/2$) covariance function

account for the long-term trend, quasi-periodic effect, and the short-scale variation, respectively.

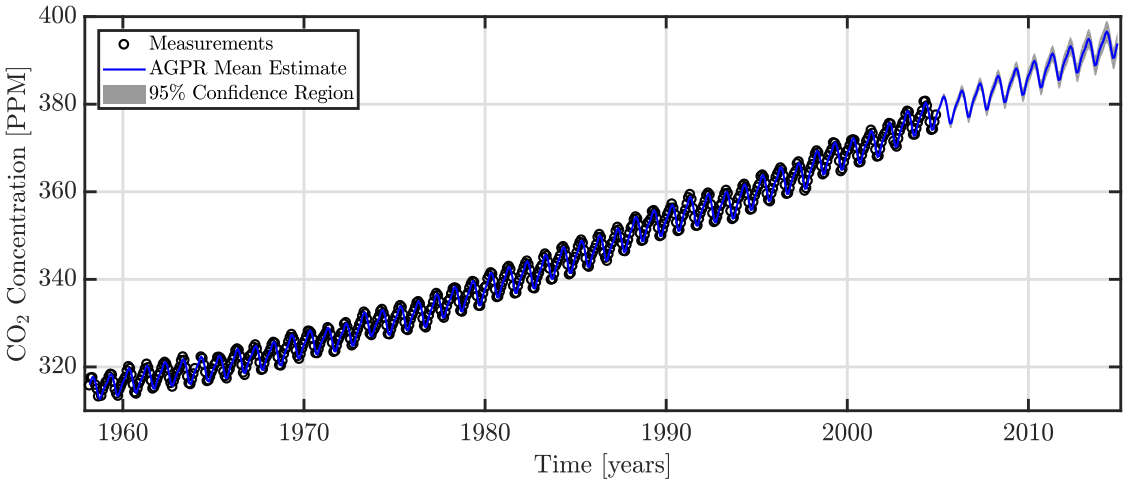
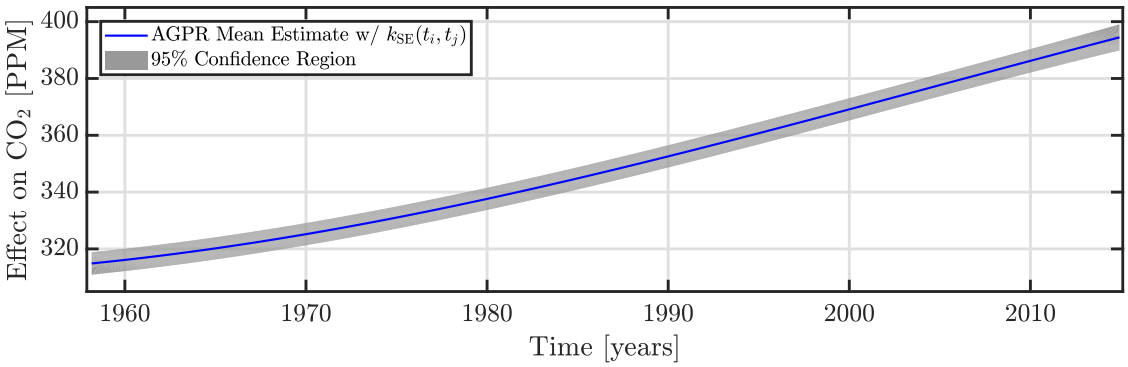
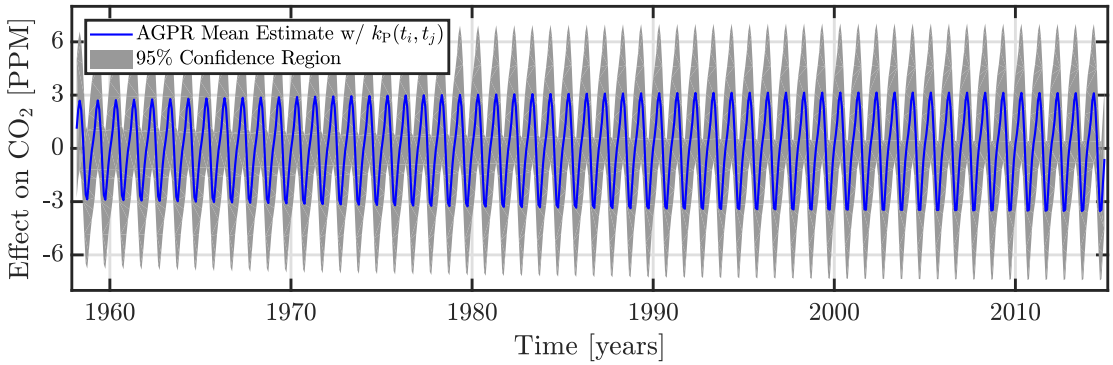


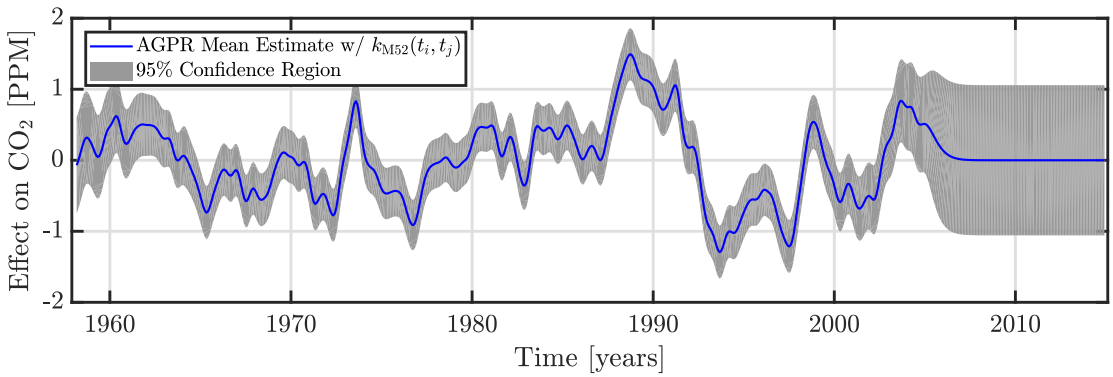
Figure 2.4. The Adaptive Gaussian Process Regression for estimating atmospheric CO_2 trends.



(a) The squared exponential covariance function captured the long-term trend.



(b) The periodic covariance function captured the quasi-periodic effect.



(c) The Matérn ($\nu = 5/2$) covariance function captured the short-scale variation.

Figure 2.5. The prediction can be plotted in terms of the components that handle the long-term trend, quasi-periodic effect, and the short-scale variation.

2.1.3 Adaptive Kalman Filter

For state and parameter estimation, Mehra [5] identified four approaches to adaptive filtering: (a) Bayesian estimation, (b) maximum likelihood methods, (c) correlation methods, and (d) covariance-matching methods. This subsection presents an Adaptive Kalman Filter (AKF) that falls under (a) Bayesian estimation for calculating the MAP parameter estimate. The advantage of this particular formulation is the capability to define a generalizable model whose unknown parameters can be efficiently optimized. Furthermore, the model is compatible with dynamical models in target tracking theory as well as covariance functions in machine learning, which could enhance functionality in aerospace applications. See [14–19] for additional information on this relationship in the context of GPR as a batch algorithm and the KF as a recursive algorithm.

A key concept behind this type of AKF is the duality between a covariance function and a state-space model (SMM). Consider a dynamic process measured by a single sensor which is defined by a GP and a measurement model with additive noise denoted as

$$f(t) \sim \mathcal{GP}\big(0, k(t, t', \boldsymbol{\theta})\big)\,, \tag{2.43a}$$

$$y_k = f(t_k) + r_k\,, \qquad r_k \sim \mathcal{N}\big(0, R(\boldsymbol{\theta})\big)\,, \tag{2.43b}$$

for the time instance $k \in [\,1\,, \cdots, T\,]$. The process is a function of the parameter $\boldsymbol{\theta} \in \mathbb{R}^p$. For the process model, $f(\cdot)$ is a random function, $k(\cdot)$ is the covariance function, and t is a scalar input. For the measurement model, y_k is a scalar measurement and r_k is the measurement noise with covariance $R(\boldsymbol{\theta})$. The noise term is zero-mean white Gaussian noise such that

$$\mathbb{E}\big[r_k\,(r_\ell)^\top\big] = R(\boldsymbol{\theta})\,\delta_{k\ell}\,, \tag{2.44}$$

where $\delta_{k\ell}$ is the Kronecker delta function.

The system in Eq. (2.43) can be reformulated as the hybrid parameterized state-space model (PSSM):

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\boldsymbol{\theta})\,\mathbf{x}(t) + \mathbf{L}(\boldsymbol{\theta})\,w(t)\ , \tag{2.45a}$$

$$y_k = \mathbf{H}(\boldsymbol{\theta})\,\mathbf{x}_k + r_k\,, \qquad r_k \sim \mathcal{N}\big(\mathbf{0}, R(\boldsymbol{\theta})\big)\,, \tag{2.45b}$$

where the state $\mathbf{x}(t)$ contains n stochastic processes

$$\mathbf{x}(t) = \begin{bmatrix} f(t) & \frac{\mathrm{d}f(t)}{\mathrm{d}t} & \cdots & \frac{\mathrm{d}^{n-1}f(t)}{\mathrm{d}t^{n-1}} \end{bmatrix}^\top\ . \tag{2.46}$$

For the continuous-time dynamics, $\mathbf{F}$ is the continuous process matrix, $\mathbf{L}$ is the noise gain matrix, and w is a white noise process with spectral density q_c. For the discrete-time measurement model, $\mathbf{H}$ is the measurement matrix such that $\mathbf{H}(\boldsymbol{\theta})\,\mathbf{x}_k = f(t_k)$.

Upon discretization of the continuous dynamics in Eq. (2.45a), the discrete PSSM can be formulated as

$$\mathbf{x}_k = \mathbf{A}_{k-1}(\boldsymbol{\theta})\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1}\,, \qquad \mathbf{q}_{k-1} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{Q}_{k-1}(\boldsymbol{\theta})\big)\,, \tag{2.47a}$$

$$y_k = \mathbf{H}(\boldsymbol{\theta})\,\mathbf{x}_k + r_k\,, \qquad r_k \sim \mathcal{N}\big(\mathbf{0}, R(\boldsymbol{\theta})\big)\,, \tag{2.47b}$$

or with the probabilistic notation as

$$p\big(\mathbf{x}_k \mid \mathbf{x}_{k-1}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \mathbf{A}_{k-1}(\boldsymbol{\theta})\,\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}(\boldsymbol{\theta})\big)\,, \tag{2.48a}$$

$$p\big(y_k \mid \mathbf{x}_k\big) = \mathcal{N}\big(y_k \mid \mathbf{H}(\boldsymbol{\theta})\,\mathbf{x}_k, R(\boldsymbol{\theta})\big)\,, \tag{2.48b}$$

where $\mathbf{A}_k$ is the process matrix and $\mathbf{q}_k$ is the process noise with covariance $\mathbf{Q}_k$. The process matrix and process noise covariance are time-varying functions. The noise term is zero-mean white Gaussian noise such that

$$\mathbb{E}\big[\mathbf{q}_k\,(\mathbf{q}_\ell)^\top\big] = \mathbf{Q}_k(\boldsymbol{\theta})\,\delta_{k\ell}\,, \tag{2.49}$$

where $\delta_{k\ell}$ is the Kronecker delta function. The covariance function determines $\mathbf{A}_k(\boldsymbol{\theta})$, $\mathbf{Q}_k(\boldsymbol{\theta})$, and $\mathbf{H}(\boldsymbol{\theta})$ while the measurement model has additive Gaussian noise with covariance $R(\boldsymbol{\theta})$. For an example of converting a GP and measurement model to a PSSM, see Appendix D. In Fig. 2.6, this Markov sequence is depicted as a Bayesian network where the states and parameters are hidden, but the measurements are observed.

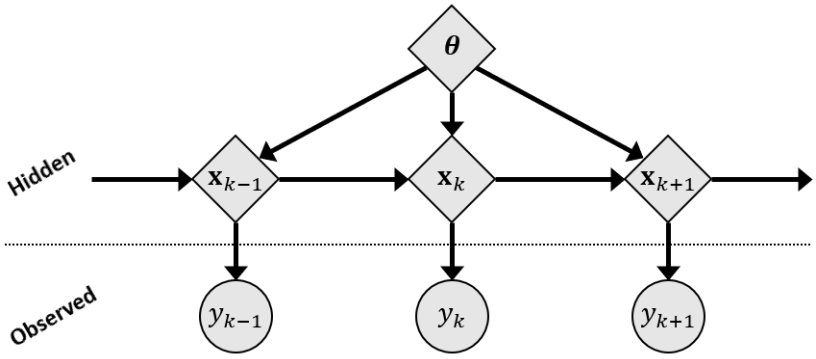


Figure 2.6. Bayesian network for the Markov sequence of the parameterized state-space model.

Figure 2.7 depicts a flow diagram of the AKF for calculating state and parameter estimates. The estimation routine consists of two phases: training and prediction. During the training phase, the algorithm alternates between optimizing state and parameter estimates. Here, the state estimator is the KF and the parameter estimator is the quasi-Newton BFGS algorithm. Once a feasible parameter estimate (e.g., the MAP parameter estimate $\hat{\boldsymbol{\theta}}_{\text{MAP}}$) is determined, the prediction phase calculates the smoothed state estimates at target locations with the KF and RTSS.

As described in [2], the MAP parameter estimate can be computed by a gradient-based optimization routine with the objective function and gradient known as the

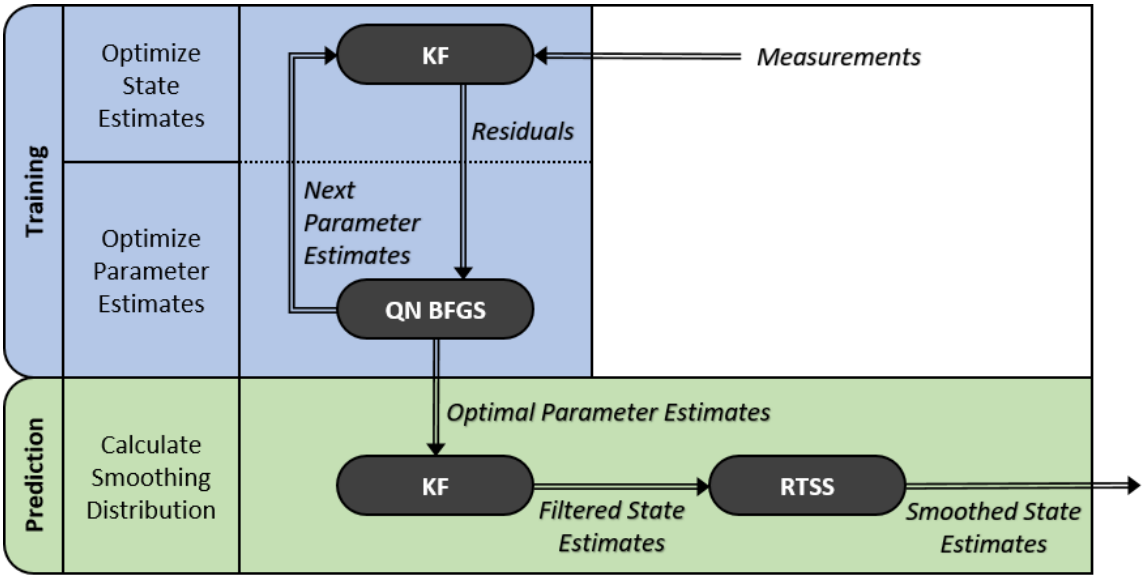


Figure 2.7. Flow Diagram for an Adaptive Kalman Filter.

energy function and the sensitivity equations, respectively. For the discrete PSSM in Eq. (2.47), the energy function and the sensitivity equations are

$$\varphi_T(\boldsymbol{\theta}) = -\log p\big(\boldsymbol{\theta}\big) + \frac{1}{2}\sum_{k=1}^{T}\Big[\log|2\pi S_k(\boldsymbol{\theta})| + [v_k(\boldsymbol{\theta})]^2\,[S_k(\boldsymbol{\theta})]^{-1}\Big], \tag{2.50a}$$

$$\begin{aligned}\frac{\partial\varphi_T(\boldsymbol{\theta})}{\partial\theta_j} = -\frac{\partial\log p\big(\boldsymbol{\theta}\big)}{\partial\theta_j} + \frac{1}{2}\sum_{k=1}^{T}\Bigg[&\mathrm{Tr}\left([S_k(\boldsymbol{\theta})]^{-1}\,\frac{\partial S_k(\boldsymbol{\theta})}{\partial\theta_j}\right)\\ &+ 2\,[v_k(\boldsymbol{\theta})]\,[S_k(\boldsymbol{\theta})]^{-1}\,\frac{\partial v_k(\boldsymbol{\theta})}{\partial\theta_j} - [v_k(\boldsymbol{\theta})]^2\,[S_k(\boldsymbol{\theta})]^{-2}\,\frac{\partial S_k(\boldsymbol{\theta})}{\partial\theta_j}\Bigg].\end{aligned} \tag{2.50b}$$

The energy function can be computed recursively as a byproduct of the KF with the marginal distribution

$$p\big(y_k \mid y_{1:k-1}\big) = \mathcal{N}\big(y_k \mid \mathbf{H}(\boldsymbol{\theta})\,\bar{\mathbf{x}}_k(\boldsymbol{\theta})\,, S_k(\boldsymbol{\theta})\big)\,, \tag{2.51}$$

where the residual vector and residual covariance matrix are

$$v_k(\boldsymbol{\theta}) = y_k - \mathbf{H}(\boldsymbol{\theta})\,\bar{\mathbf{x}}_k(\boldsymbol{\theta}) \;, \tag{2.52a}$$

$$S_k(\boldsymbol{\theta}) = \mathbf{H}(\boldsymbol{\theta})\,\bar{\mathbf{P}}_k(\boldsymbol{\theta})\,\mathbf{H}^\top(\boldsymbol{\theta}) + R(\boldsymbol{\theta}) \;. \tag{2.52b}$$

The sensitivity equations can be computed recursively with the derivatives of the KF with respect to each parameter.

A KF and RTSS can calculate a smoothing distribution that is equivalent to the solution in GPR. The key feature of GPR is the covariance function, while the KF intuitively uses a state vector. GPR is a batch algorithm while the KF and RTSS are recursive algorithms. The major benefit of the KF and RTSS over GPR is a reduction of the computational complexity from the cubic $\mathcal{O}\big(T^3\big)$ to the linear $\mathcal{O}\big(T\big)$ for T steps. This computation complexity is the results of a matrix inversion as discussed in [2, 14, 19].

Example of an Adaptive Kalman Filter

To demonstrate the implementation of the AKF and RTSS, an example is provided for estimating the atmospheric CO_2. The solution with AGPR was presented in Section 2.1.2 and by Rasmussen [9]. Here, the example presents the solution with an AKF and RTSS by Solin and Särkkä [19]. The system can be modeled with the zero-mean GP and a noisy measurement process, which can be converted to a discrete PSSM. The design choices for the AKF and RTSS are summarized in Table 2.3. The measurement model had additive Gaussian noise with covariance $R(\boldsymbol{\theta}) = \sigma_{\mathrm{n}}^2$. The covariance function was the addition of the squared exponential, periodic, and Matérn ($\nu = 5/2$) covariance functions, which were converted into a discrete PSSM. The state estimator was the KF and RTSS while the parameter optimization algorithm was the quasi-Newton BFGS algorithm. A log-uniform prior distribution was placed on each parameter.

Table 2.3. The design choices for an example of an Adaptive Kalman Filter and Rauch-Tung-Striebel Smoother.

(1) **State-Space Model** Squared Exponential, Periodic, and Matérn ($\nu = 5/2$) Model
(2) **State Estimation** - KF and RTSS
(3) **Parameter Optimization** - Quasi-Newton BFGS
(4) **Parameters** $\boldsymbol{\theta} = [\, \sigma_n^2 \,,\, \ell_{\mathrm{SE}} \,,\, \sigma_{\mathrm{SE}}^2 \,,\, \ell_{\mathrm{P}} \,,\, \sigma_{\mathrm{P}}^2 \,,\, \ell_{\mathrm{P,SE}} \,,\, \ell_{\mathrm{M52}} \,,\, \sigma_{\mathrm{M52}}^2 \,]$
(5) **Initial Values for Parameters** $\boldsymbol{\theta}_0 = [\, 1 \,,\; 100 \,,\; 5000 \,,\; 1 \,,\; 1 \,,\; 100 \,,\; 10 \,,\; 10 \,]$
(6) **Prior Distributions** - Log-Uniform Distributions $p\big(\log \theta_j\big) \propto 1 \quad \forall j$

The results are displayed in Fig. 2.8. The markers represent yearly measurements from 1959 to 2006. The gray region is the 95% confidence region of the prediction after filtering and smoothing with the MAP estimate of the parameters. The final values for the parameters were

$$\begin{aligned} \boldsymbol{\theta}_* = [\, &0.046004 \,,\; 124.35 \,,\; 1.4182\text{e+}05 \,,\; 1.3569 \,, \\ &6.3375 \,,\; 105.22 \,,\; 0.65691 \,,\; 0.28992 \,] \,, \end{aligned} \tag{2.53}$$

and a reduction of the objective function from 639.94 to 136.634 in 37 function calls. However, the AKF and RTSS increases the computational time to 118 s from about 6 s in AGPR. The conversion of the periodic covariance function into the discrete PSSM requires a large state vector. This example demonstrated that the computational benefit of a linear complexity with the number of measurements T can be dominated by the cubic complexity with the number of states n.

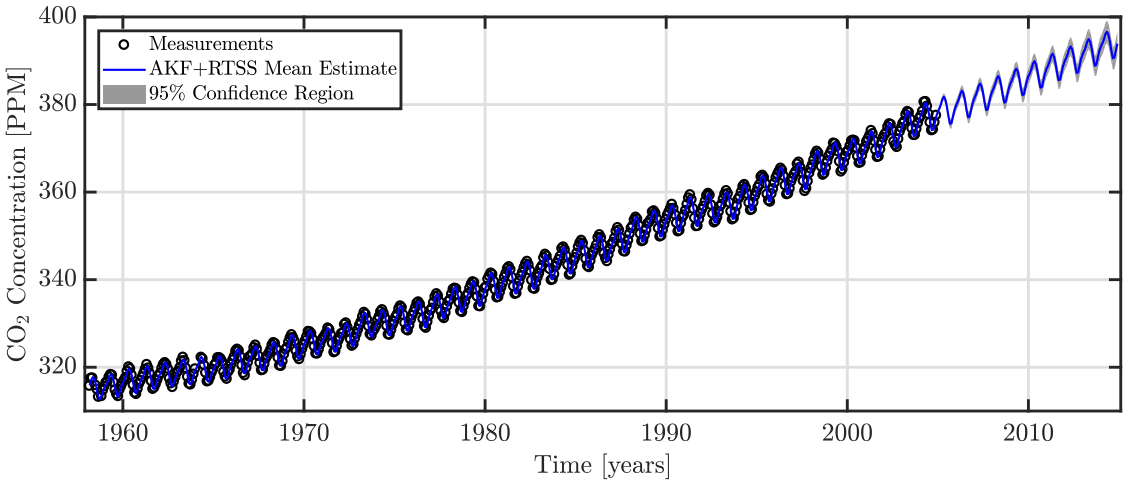


Figure 2.8. The Adaptive Kalman Filter and Rauch-Tung-Striebel Smoother for estimating atmospheric CO_2 trends.

2.2 Exploratory Work with Adaptive Filters in Aerospace Applications

This chapter has discussed an Adaptive Kalman Filter (AKF) based on parameterized state-space model (PSSM) that is compatible with machine learning models as well as dynamical models. The different underlying models can enable a wide variety of applications for adaptive filters in aerospace applications. The remainder of this chapter presents exploratory work of the PSSMs and two potential aerospace applications, which indicate future research directions. If the reader is interested primarily in the Adaptive Distributed Kalman Filter (ADKF), then see Chapter 3 as the remainder of this chapter is not necessary to understand the problem formulation and solution.

2.2.1 A target tracking comparison with different models

The objective is to understand how target tracking models compare with covariance functions as a PSSM, which can then be solved with an AKF and Rauch-Tung-Striebel Smoother (RTSS). Two common models for target tracking are the Nearly

Constant Acceleration Model (NCAM) and Singer Acceleration Model (SAM). In order to maintain compatibility with GPstuff, the code was programmed with these two target tracking models as a hybrid PSSM, which are then discretized for the AKF. The measurement model for a sensor tracking the target's position is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \qquad R = \sigma_n^2 . \tag{2.54}$$

Some common covariance functions include the rational quadratic, polynomial, Matérn, squared exponential, exponential, and neural network. In GPstuff, the covariance functions are converted to a hybrid PSSM and then discretized for the AKF. The discretization can be accomplished with an exact solution or matrix fraction decomposition. An example is provided in Appendix D. The parameters are generally a magnitude σ^2 and characteristic length-scale ℓ.

The design choices for the model comparison are summarized in Table 2.4. The covariance functions and target tracking models were all converted into a discrete PSSM with additive Gaussian noise. The state estimator was the KF and RTSS while the parameter optimization algorithm was the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. A log-uniform prior distribution was placed on each parameter. This approach is an extension of [15, 20] by Reece and Roberts in which a covariance function was formulated based on the NCAM, which could then be used with Gaussian Process Regression (GPR).

Nearly Constant Acceleration Model

The Nearly Constant Acceleration Model (NCAM) [7,8] is defined with the continuous-time matrices

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} . \tag{2.55}$$

Table 2.4. Design choices for a comparison of parameterized state-space models for target tracking in 1-D.

(1) **State-Space Model** - *See each example for details*
(2) **State Estimation** - KF and RTSS
(3) **Parameter Optimization** - Quasi-Newton BFGS
(4) **Parameters** - *See each example for details*
(5) **Initial Values for Parameters** - *See each example for details*
(6) **Prior Distributions** - Log-Uniform Distributions

The state vector

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) & x_3(t) \end{bmatrix}^\top \tag{2.56}$$

is composed of the position $x_1(t)$, velocity $x_2(t)$, and acceleration $x_3(t)$. The acceleration is driven by the continuous-time process noise $w(t)$ such that

$$\mathbb{E}\big[\, w(t) \,\big] = 0 \,, \tag{2.57a}$$

$$\mathbb{E}\big[\, w(t)\, w(t') \,\big] = q_c \delta(t - t') \,, \tag{2.57b}$$

where q_c is the power spectral density and $\delta(\cdot)$ is the Dirac delta function. The discrete-time matrices are

$$\mathbf{A}_k = \begin{bmatrix} 1 & \Delta t_k & \frac{1}{2}(\Delta t_k)^2 \\ 0 & 1 & \Delta t_k \\ 0 & 0 & 1 \end{bmatrix} \,, \qquad \mathbf{Q}_k = q_c \begin{bmatrix} \frac{1}{20}(\Delta t_k)^5 & \frac{1}{8}(\Delta t_k)^4 & \frac{1}{6}(\Delta t_k)^3 \\ \frac{1}{8}(\Delta t_k)^4 & \frac{1}{3}(\Delta t_k)^3 & \frac{1}{2}(\Delta t_k)^2 \\ \frac{1}{6}(\Delta t_k)^3 & \frac{1}{2}(\Delta t_k)^2 & \Delta t_k \end{bmatrix} \,, \tag{2.58}$$

where $\Delta t_k = t_{k+1} - t_k$ is the sampling time. The state vector

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} & x_{2,k} & x_{3,k} \end{bmatrix}^\top \tag{2.59}$$

is composed of the position $x_{1,k}$, velocity $x_{2,k}$, and acceleration $x_{3,k}$.

With this model, the matrices that were functions of the parameters included $\mathbf{Q}_k(\boldsymbol{\theta})$ and $R(\boldsymbol{\theta})$. Additionally, the initial state estimate was

$$\widetilde{\mathbf{x}}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top , \qquad \widetilde{\mathbf{P}}_0(\boldsymbol{\theta}) = \mathrm{Diag}\Big(p_{11} , \;\; p_{22} , \;\; p_{33} \Big) \; . \tag{2.60}$$

Table 2.5 summarizes the parameters and their initial values for the NCAM.

Table 2.5. Design choices for the Nearly Constant Acceleration Model.

(1) **State-Space Model** - NCAM
(4) **Parameters** - $\boldsymbol{\theta} = [\, \sigma_n^2 \,,\, q_c \,,\, p_{11} \,,\, p_{22} \,,\, p_{33} \,]$
(5) **Initial Values for Parameters** - $\boldsymbol{\theta}_0 = [\, 1 \,,\; 0.1 \,,\; 1 \,,\; 1 \,,\; 1 \,]$

The example's results are displayed in Fig. 2.9. The final values for the parameters were

$$\boldsymbol{\theta}_* = [\, 12.53 \,,\; 0.0011271 \,,\; 6.0114\text{e}{-07} \,,\; 0.25926 \,,\; 2.8091\text{e}{-21} \,] \; , \tag{2.61}$$

and a reduction of the objective function from 301.231 to 129.024 in 33 function calls.

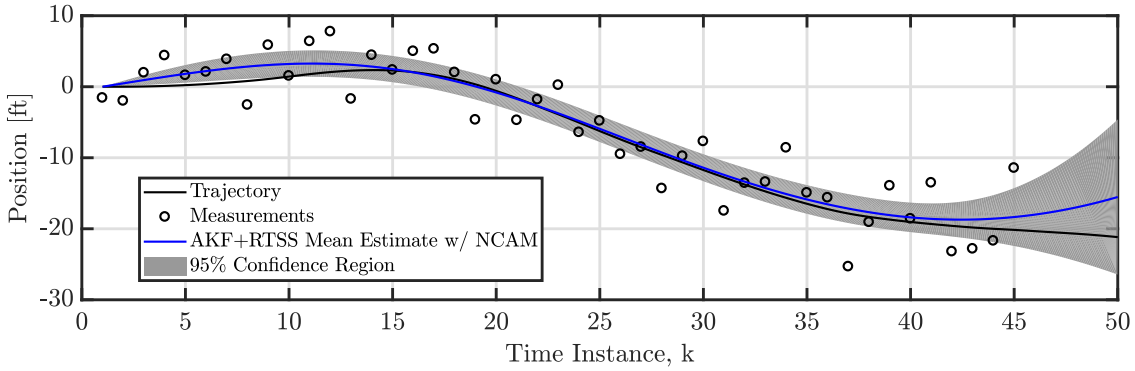


Figure 2.9. Optimal smoothed estimates with the Nearly Constant Acceleration Model.

Singer Acceleration Model

The Singer Acceleration Model (SAM) [7,8,21] is defined with the continuous-time matrices

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\alpha \end{bmatrix} , \qquad \mathbf{L} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} . \tag{2.62}$$

The state vector

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) & x_3(t) \end{bmatrix}^\top \tag{2.63}$$

is composed of the position $x_1(t)$, velocity $x_2(t)$, and acceleration $x_3(t)$. The acceleration is driven by the continuous-time process noise $w(t)$ such that

$$\dot{x}_3(t) = -\alpha x_3(t) + w(t) , \tag{2.64}$$

where

$$\mathbb{E}\big[w(t) \big] = 0 , \tag{2.65a}$$

$$\mathbb{E}\big[w(t)\, w(t') \big] = 2\alpha\sigma^2_{\text{SAM}} \delta(t - t') , \tag{2.65b}$$

where $1/\alpha$ is the time constant of the target acceleration autocorrelation, σ^2_{SAM} is the instantaneous variance, and $\delta(\cdot)$ is the Dirac delta function. The discrete-time matrices are

$$\mathbf{A}_k = \begin{bmatrix} 1 & \Delta t_k & \left(\alpha\Delta t_k - 1 + e^{-\alpha\Delta t_k}\right)/\alpha^2 \\ 0 & 1 & \left(1 - e^{-\alpha\Delta t_k}\right)/\alpha \\ 0 & 0 & e^{-\alpha\Delta t_k} \end{bmatrix} , \tag{2.66a}$$

$$\mathbf{Q}_k = 2\alpha\sigma^2_{\text{SAM}} \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} , \tag{2.66b}$$

where $\Delta t_k = t_{k+1} - t_k$ is the sampling time and

$$q_{11} = \frac{1}{2\alpha^5}\left[1 - \mathrm{e}^{-2\alpha\Delta t_k} + 2\alpha\Delta t_k + \frac{2\alpha^3\left(\Delta t_k\right)^3}{3} - 2\alpha^2\left(\Delta t_k\right)^2 \right.$$
$$\left. -4\alpha\Delta t_k \mathrm{e}^{-\alpha\Delta t_k}\right] , \tag{2.67a}$$

$$q_{12} = \frac{1}{2\alpha^4}\left[\mathrm{e}^{-2\alpha\Delta t_k} + 1 - 2\mathrm{e}^{-\alpha\Delta t_k} + 2\alpha\Delta t_k \mathrm{e}^{-\alpha\Delta t_k} - 2\alpha\Delta t_k + \alpha^2\left(\Delta t_k\right)^2\right] , \tag{2.67b}$$

$$q_{13} = \frac{1}{2\alpha^3}\left[1 - \mathrm{e}^{-2\alpha\Delta t_k} - 2\alpha\Delta t_k \mathrm{e}^{-\alpha\Delta t_k}\right] , \tag{2.67c}$$

$$q_{22} = \frac{1}{2\alpha^3}\left[4\mathrm{e}^{-\alpha\Delta t_k} - 3 - \mathrm{e}^{-2\alpha\Delta t_k} + 2\alpha\Delta t_k\right] , \tag{2.67d}$$

$$q_{23} = \frac{1}{2\alpha^2}\left[\mathrm{e}^{-2\alpha\Delta t_k} + 1 - 2\mathrm{e}^{-\alpha\Delta t_k}\right] , \tag{2.67e}$$

$$q_{33} = \frac{1}{2\alpha}\left[1 - \mathrm{e}^{-2\alpha\Delta t_k}\right] . \tag{2.67f}$$

The state vector

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} & x_{2,k} & x_{3,k} \end{bmatrix}^\top \tag{2.68}$$

is composed of the position $x_{1,k}$, velocity $x_{2,k}$, and acceleration $x_{3,k}$.

With this model, the matrices that were functions of the parameters included $\mathbf{A}_k(\boldsymbol{\theta})$, $\mathbf{Q}_k(\boldsymbol{\theta})$, and $R(\boldsymbol{\theta})$. Additionally, the initial state estimate was

$$\widetilde{\mathbf{x}}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top , \qquad \widetilde{\mathbf{P}}_0(\boldsymbol{\theta}) = \mathrm{Diag}\Big(p_{11}\,, \;\; p_{22}\,, \;\; p_{33}\Big) . \tag{2.69}$$

Table 2.6 summarizes the parameters and their initial values for the SAM.

Table 2.6. Design choices for the Singer Acceleration Model.

(1) **State-Space Model** - SAM
(4) **Parameters** - $\boldsymbol{\theta} = [\,\sigma_n^2\,,\,\alpha\,,\,\sigma_{\mathrm{SAM}}^2\,,\,p_{11}\,,\,p_{22}\,,\,p_{33}\,]$
(5) **Initial Values for Parameters** - $\boldsymbol{\theta}_0 = [\,1\,,\;0.01\,,\;0.1\,,\;1\,,\;1\,,\;1\,]$

The example's results are displayed in Fig. 2.10. The final values for the parameters were

$$\boldsymbol{\theta}_* = [\,12.656\,,\ 0.15269\,,\ 0.0097857\,,\ 1.7312\text{e}{-07}\,,\ 0.23625\,,\ 7.6397\text{e}{-12}\,]\,, \quad (2.70)$$

and a reduction of the objective function from 313.05 to 128.446 in 39 function calls.

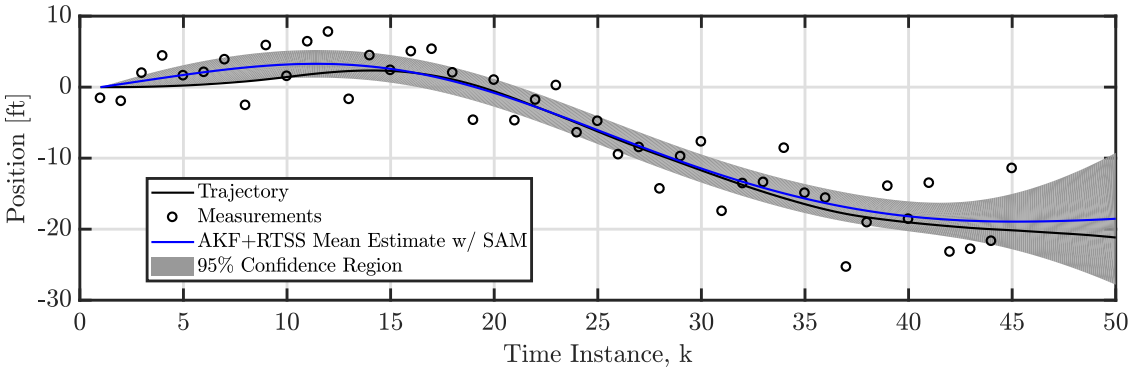


Figure 2.10. Optimal smoothed estimates with the Singer Acceleration Model.

Linear and constant covariance function

Table 2.7 summarizes the parameters and their initial values for the linear and constant covariance function denoted as

$$k_{\mathrm{L,C}}(t_i, t_j) = \sigma_{\mathrm{L}}^2 t_i t_j + \sigma_{\mathrm{C}}^2 \,. \tag{2.71}$$

Table 2.7. Design choices for the linear and constant covariance function.

(1) **State-Space Model** - Linear + Constant Model
(4) **Parameters** - $\boldsymbol{\theta} = [\,\sigma_n^2\,,\,\sigma_{\mathrm{L}}^2\,,\,\sigma_{\mathrm{C}}^2\,]$
(5) **Initial Values for Parameters** - $\boldsymbol{\theta}_0 = [\,1\,,\;1\,,\;100\,]$

The example's results are displayed in Fig. 2.11. The final values for the parameters were

$$\boldsymbol{\theta}_* = [\,22.17\,,\;0.36254\,,\;59.149\,]\,, \tag{2.72}$$

and a reduction of the objective function from 526.862 to 138.399 in 13 function calls.

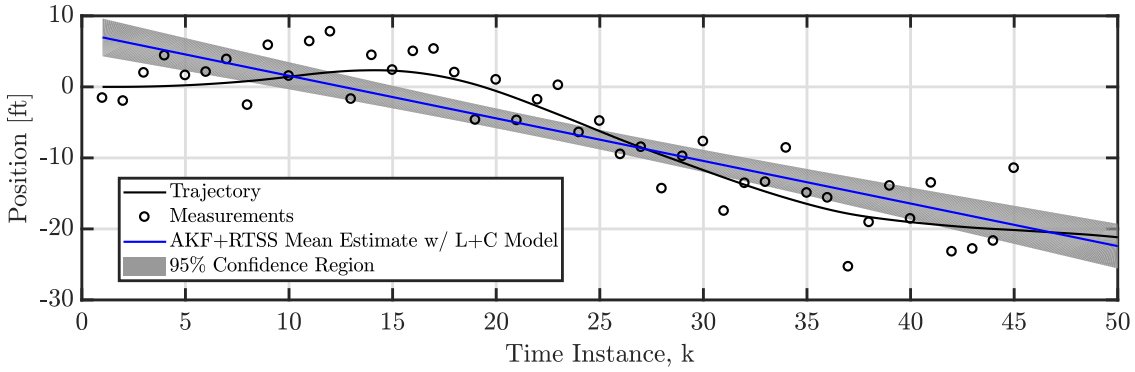


Figure 2.11. Optimal smoothed estimates with the linear and constant covariance function.

Matérn ($\nu = 3/2$) covariance function

Table 2.8 summarizes the parameters and their initial values for the Matérn ($\nu = 3/2$) covariance function denoted as

$$k_{\text{M32}}(t_i, t_j) = \sigma^2_{\text{M32}} \left(1 + \frac{\sqrt{3}\tau}{\ell_{\text{M32}}}\right) \exp\left(-\frac{\sqrt{3}\tau}{\ell_{\text{M32}}}\right) , \tag{2.73}$$

where $\tau = |t_i - t_j|$.

Table 2.8. Design choices for the Matérn ($\nu = 3/2$) covariance function.

(1) **State-Space Model** - Matérn ($\nu = 3/2$) Model
(4) **Parameters** - $\boldsymbol{\theta} = [\, \sigma_n^2 \,,\, \ell_{\text{M32}} \,,\, \sigma^2_{\text{M32}} \,]$
(5) **Initial Values for Parameters** - $\boldsymbol{\theta}_0 = [\, 1 \,,\; 1 \,,\; 100 \,]$

The example's results are displayed in Fig. 2.12. The final values for the parameters were

$$\boldsymbol{\theta}_* = [\, 12.918 \,,\; 30.694 \,,\; 146.63 \,] \,, \tag{2.74}$$

and a reduction of the objective function from 154.687 to 130.046 in 31 function calls.

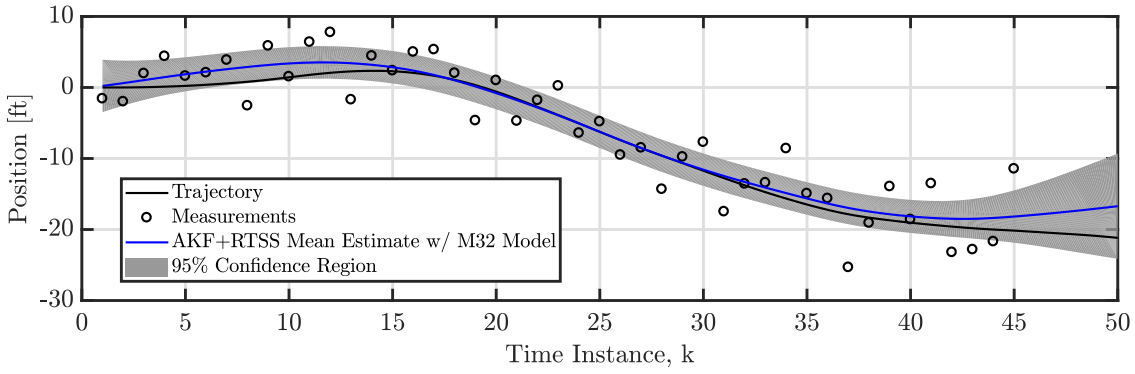


Figure 2.12. Optimal smoothed estimates with the Matérn ($\nu = 3/2$) covariance function.

Matérn ($\nu = 5/2$) covariance function

Table 2.9 summarizes the parameters and their initial values for the Matérn ($\nu = 5/2$) covariance function denoted as

$$k_{\text{M52}}(t_i, t_j) = \sigma^2_{\text{M52}} \left(1 + \frac{\sqrt{5}\tau}{\ell_{\text{M52}}} + \frac{5\tau^2}{3\ell^2_{\text{M52}}} \right) \exp\left(-\frac{\sqrt{5}\tau}{\ell_{\text{M52}}} \right) , \tag{2.75}$$

where $\tau = |t_i - t_j|$.

Table 2.9. Design choices for the Matérn ($\nu = 5/2$) covariance function.

(1) **State-Space Model** - Matérn ($\nu = 5/2$) Model
(4) **Parameters** - $\boldsymbol{\theta} = [\, \sigma_n^2 \,,\, \ell_{\text{M52}} \,,\, \sigma^2_{\text{M52}} \,]$
(5) **Initial Values for Parameters** - $\boldsymbol{\theta}_0 = [\, 1 \,,\; 1 \,,\; 100 \,]$

The example's results are displayed in Fig. 2.13. The final values for the parameters were

$$\boldsymbol{\theta}_* = [\, 12.811 \,,\; 24.033 \,,\; 142.82 \,] \,, \tag{2.76}$$

and a reduction of the objective function from 153.671 to 129.497 in 28 function calls.

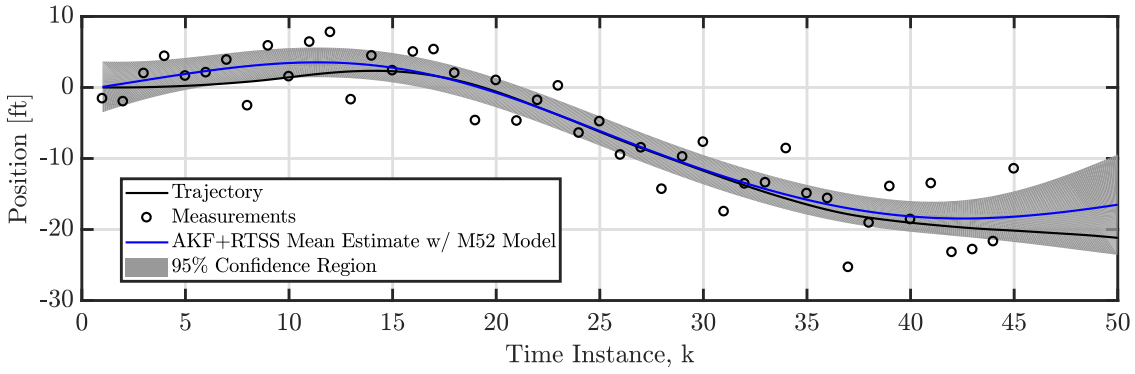


Figure 2.13. Optimal smoothed estimates with the Matérn ($\nu = 5/2$) covariance function.

Squared exponential covariance function

Table 2.10 summarizes the parameters and their initial values for the squared exponential covariance function denoted as

$$k_{\text{SE}}(t_i, t_j) = \sigma_{\text{SE}}^2 \exp\left(-\frac{\tau^2}{2\ell_{\text{SE}}^2}\right), \tag{2.77}$$

where $\tau = |t_i - t_j|$.

Table 2.10. Design choices for the squared exponential covariance function.

(1) **State-Space Model** - Squared Exponential Model
(4) **Parameters** - $\boldsymbol{\theta} = [\,\sigma_n^2\,,\,\ell_{\text{SE}}\,,\,\sigma_{\text{SE}}^2\,]$
(5) **Initial Values for Parameters** - $\boldsymbol{\theta}_0 = [\,1\,,\;100\,,\;100000\,]$

The example's results are displayed in Fig. 2.14. The final values for the parameters were

$$\boldsymbol{\theta}_* = [\,12.661\,,\;17.297\,,\;129.23\,]\,, \tag{2.78}$$

and a reduction of the objective function from 345.71 to 128.935 in 23 function calls.

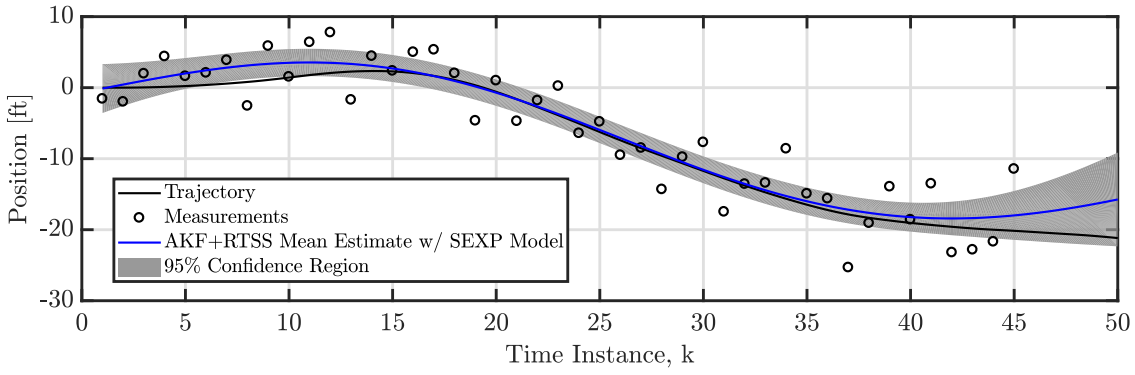


Figure 2.14. Optimal smoothed estimates with the squared exponential covariance function.

Exponential covariance function

Table 2.11 summarizes the parameters and their initial values for the exponential covariance function denoted as

$$k_{\mathrm{E}}(t_i, x_j) = \sigma_{\mathrm{E}}^2 \exp\left(-\frac{\tau}{\ell_{\mathrm{E}}}\right) , \tag{2.79}$$

where $\tau = |t_i - t_j|$.

Table 2.11. Design choices for the exponential covariance function.

(1) **State-Space Model** - Exponential Model
(4) **Parameters** - $\boldsymbol{\theta} = [\, \sigma_n^2 \,,\, \ell_{\mathrm{E}} \,,\, \sigma_{\mathrm{E}}^2 \,]$
(5) **Initial Values for Parameters** - $\boldsymbol{\theta}_0 = [\, 1 \,,\, 1 \,,\, 1 \,]$

The example's results are displayed in Fig. 2.15. The final values for the parameters were

$$\boldsymbol{\theta}_* = [\, 0.0023424 \,,\, 8.1542 \,,\, 114.82 \,] \,, \tag{2.80}$$

and a reduction of the objective function from 1004.49 to 137.023 in 30 function calls.

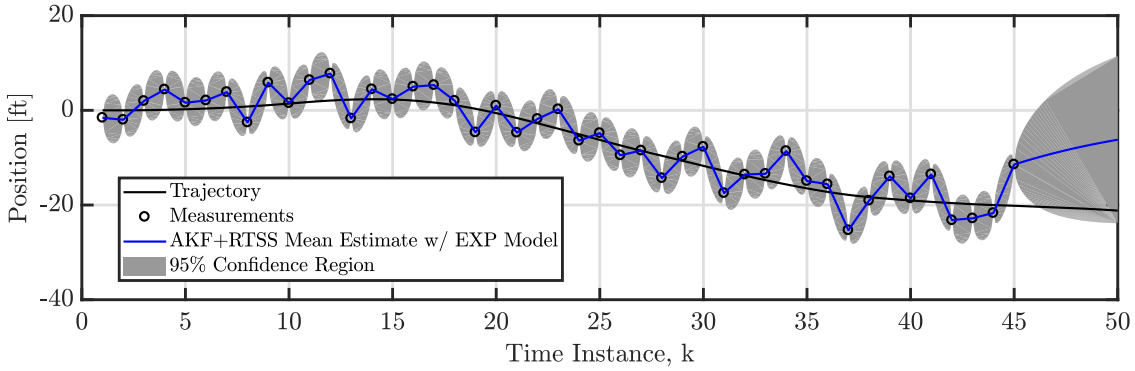


Figure 2.15. Optimal smoothed estimates with the exponential covariance function.

A performance summary of the parameterized state-space models

The previous plots were the smoothed results from the same data set, which was a random sample of the SAM with $\Delta t = 1$ s, $\alpha = 0.001$ Hz, $\sigma_{\text{SAM}} = 1$ ft/s^2, and $\sigma_n = \sqrt{10}$ ft. The experiment was repeated about 200 times so that a trend could be established regarding the performance, which is summarized in Table 2.12. The two target tracking models provided consistently good results for each randomly generated data set with the same dynamical model. The only covariance function that provided comparable performance was the squared exponential. A larger variety of simulations is needed for a more comprehensive comparison as the results are dependent on the fixed parameters defining the data set. For instance, generating a data set with a low signal-to-noise ratio can present significant difficulties in the state and parameter estimation. Furthermore, the convergence depends upon the selection of reasonable initial values. The key takeaway is that covariance functions, besides the squared exponential, can not replace the target tracking models given the difficulties with convergence of the parameter estimation and the regression quality. This result follows from the fact that the properties defining the covariance functions are weaker than the properties defining a dynamical system [22].

Table 2.12. Summary of convergence for the parameter estimation and regression quality for different parameterized state-space models.

State-Space Model	**Parameter Estimation**	**Regression Quality**
NCAM	Always converges	Good
SAM	Always converges	Good
Linear + Constant	Usually converges	Very poor
Matérn ($\nu = 3/2$)	Always converges	Usually good, but sometimes very poor
Matérn ($\nu = 5/2$)	Always converges	Usually good, but sometimes very poor
Squared Exponential	Always converges	Good
Exponential	Usually converges	Poor

2.2.2 Target tracking of a hypersonic boost-glide vehicle

Intercepting a hypersonic boost-glide vehicle requires accurate predictions of the trajectory given noisy measurements. As the interception problem would require on-line calculations, the primary problem is developing a solution to calculate accurate predictions in the necessary computational time. The trajectory of a hypersonic boost-glide vehicle consists of multiple phases: launch, glide, and terminal. The trajectory data for the glide and terminal phase in this example was provided by Sean Nolan.[2] The trajectory data was calculated by using an indirect method of optimal control theory to maximize the terminal energy. The controls include the angle of attack and bank angle while the constraints include the maximum altitude, heat rate, and g-loading. The optimization problem is a multi-point boundary value problem with ordinary differential equations that is solved via the MATLAB function: "`bvp4c`". Figure 2.16 depicts the altitude time history for an optimal trajectory with an initial altitude of 49 km, maximum altitude of 50 km, and downrange distance of 3,000 km. A set of measurements was created by randomly sampling a Gaussian distribution with variance $\sigma_n = \sqrt{10}$ km at irregular time intervals. The design choices are summarized in Table 2.13 for two solutions: the squared exponential (SE) model and SAM.

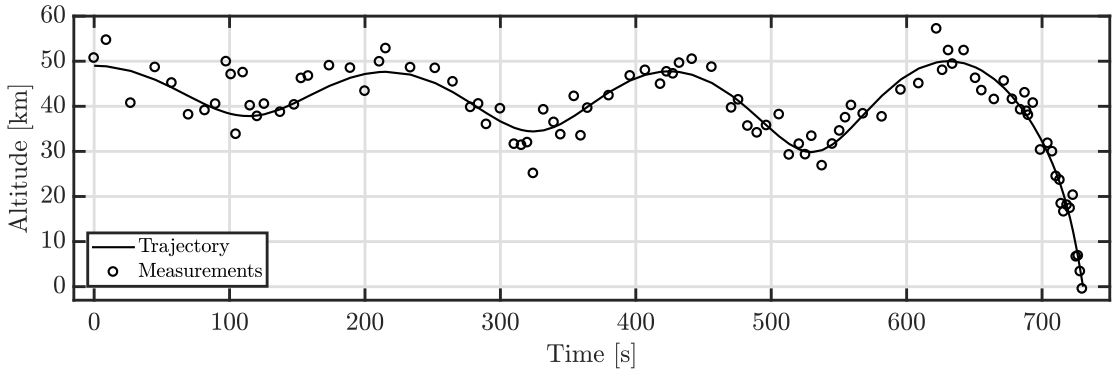


Figure 2.16. Altitude time history of a hypersonic boost-glide vehicle.

[2]Master's student at the School of Aeronautics and Astronautics, Purdue University

Table 2.13. Design choices for an example of target tracking of a hypersonic boost-glide vehicle.

(1) **State-Space Model**
(a) Squared Exponential Model
(b) SAM
(2) **State Estimation** - KF and RTSS
(3) **Parameter Optimization** - Quasi-Newton BFGS
(4) **Parameters**
(a) $\boldsymbol{\theta} = [\, \sigma_n^2 \,,\, \ell_{\text{SE}} \,,\, \sigma_{\text{SE}}^2 \,]$
(b) $\boldsymbol{\theta} = [\, \sigma_n^2 \,,\, \alpha \,,\, \sigma_{\text{SAM}}^2 \,,\, p_{11} \,,\, p_{22} \,,\, p_{33} \,]$
(5) **Initial Values for Parameters**
(a) $\boldsymbol{\theta}_0 = [\, 1 \,,\; 100 \,,\; 100000 \,]$
(b) $\boldsymbol{\theta}_0 = [\, 1 \,,\; 0.01 \,,\; 0.1 \,,\; 1 \,,\; 1 \,,\; 1 \,]$
(6) **Prior Distributions** - Log-Uniform Distributions
$p\big(\log \theta_j\big) \propto 1 \quad \forall j$

Figure 2.17 depicts the smoothed results for the SE model and SAM. The black line is the target's true trajectory while the circles are the sensor's noisy measurements. The blue line is the estimated mean for the SE model while the blue dotted line is the 95% confidence region. The red line is the estimated mean for the SAM while the red dotted line is the 95% confidence region. The final values for the parameters in the SE model were

$$\boldsymbol{\theta}_* = [\, 15.149 \,,\; 104.71 \,,\; 2804.2 \,] \,, \tag{2.81}$$

and a reduction of the objective function from 711.715 to 311.0361 in 16 function calls. The final values for the parameters in the SAM were

$$\boldsymbol{\theta}_* = [\, 14.108 \,,\; 0.02487 \,,\; 8.2665\text{e}{-05} \,,\; 2550.6 \,,\; 0.018407 \,,\; 8.9816\text{e}{-11} \,] \,, \tag{2.82}$$

and a reduction of the objective function from 1196.44 to 310.9371 in 99 function calls. Although the smoothed estimates were very similar, the total time for the SE model

and SAM was 0.84437 s and 9.448 s, respectively. Even though the data is based on a dynamical system, the machine learning model was an order of a magnitude faster than the target tracking model.

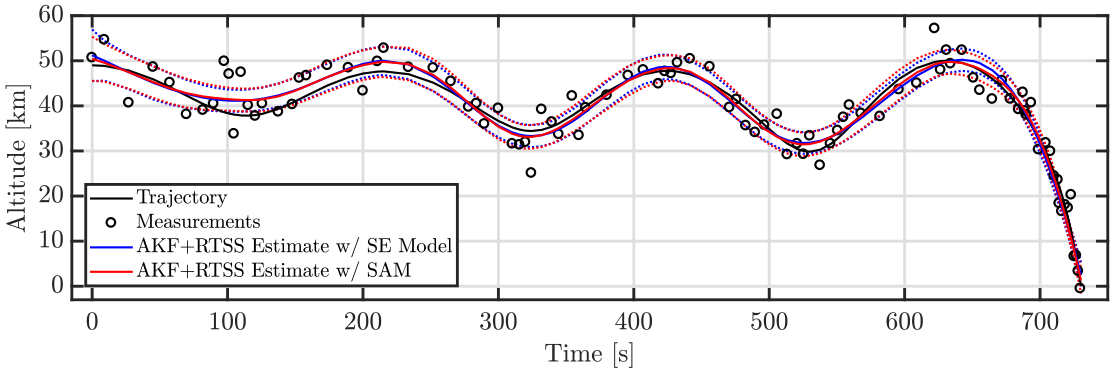


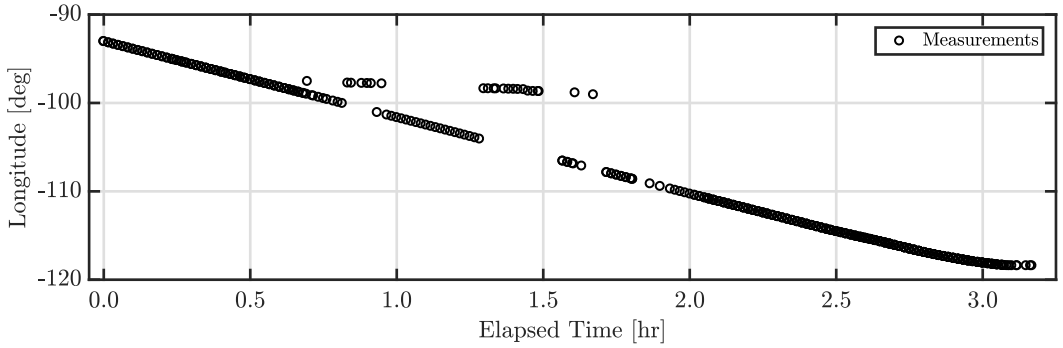
Figure 2.17. Optimal smoothed estimates with the SE model and SAM.

A PSSM based on the periodic covariance function was also attempted. In general, the periodic model would not converge. Furthermore, a joint model was attempted with the periodic model and NCAM. The joint model did not provide any improvements in terms of regression and had a much larger computational time.

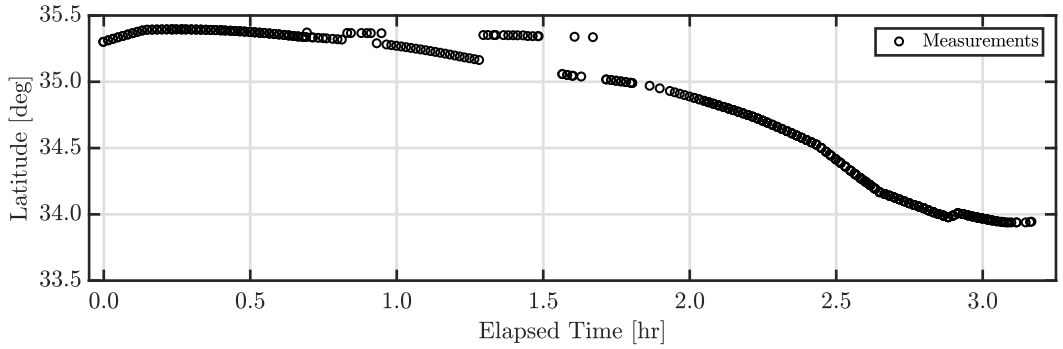
In a real-world scenario, the objective would be to predict the future trajectory of the hypersonic boost-glide vehicle for a potential interception. The results for these prediction scenarios were very poor at predicting the (a) impact location and (b) peaks and valleys as a result of a simplistic model. A potential improvement is to include the dynamics. However, this requires solving an optimization routine, which is not ideal for an on-line implementation. An alternative option is to use deep learning for the off-line training of a surrogate dynamical model, which can then be used in an on-line implementation.

2.2.3 Fault detection in aircraft state data

The joint use of machine learning and target tracking models can improve fault detection capabilities such as in aircraft state data. In 2007, the FAA established the System Wide Information Management (SWIM) Program as an information system for the NAS. SWIM provides users access to a variety of data products through the NAS Enterprise Messaging System (NEMS). For this analysis, the raw data was provided by Robust Analytics, Inc. after merging (a) Aircraft Situation Display (ASDI) data from the Traffic Flow Management (TFMS) product and (b) En Route flight data from the SWIM Flight Data Publication Service (SFDPS) product. As an end-user, the data can incorporate errors as shown in Fig. 2.18, which depicts the latitude and longitude for an aircraft traveling between KATL and KLAX.



(a) Latitude data.



(b) Longitude data.

Figure 2.18. Longitude and latitude data with multiple errors.

An example is provided for estimating the position of a target with an AKF. The design choices of the solution for the latitude and longitude data are summarized in Table 2.14. The underlying PSSM is a joint model of the Matérn ($\nu = 5/2$) and NCAM where σ_n is the standard deviation of the measurement noise. The state estimator is a KF and RTSS while the parameter optimization routine was the quasi-Newton BFGS.

Table 2.14. Design choices for an example of fault detection in aircraft state data.

(1) **State-Space Model** Matérn ($\nu = 5/2$) Model and NCAM
(2) **State Estimation** - KF and RTSS
(3) **Parameter Optimization** - Quasi-Newton BFGS
(4) **Parameters** $\boldsymbol{\theta} = [\, \sigma_n^2 \,,\, \ell_{\text{M52}} \,,\, \sigma_{\text{M52}}^2 \,,\, q_c \,,\, p_{11} \,,\, p_{22} \,,\, p_{33} \,]$
(5) **Initial Values for Parameters** $\boldsymbol{\theta}_0 = [\, 1 \,,\; 1 \,,\; 100 \,,\; 0.0001 \,,\; 1 \,,\; 1 \,,\; 1 \,]$
(6) **Prior Distributions** - Log-Uniform Distributions $p\big(\log \theta_j\big) \propto 1 \quad \forall j$

Figure 2.19 displays the smoothed results for the latitude data. In Fig. 2.19(a), the target's true trajectory is unknown, but the circles are the sensor's noisy measurements. The blue line is the smoothed mean estimate while the gray region is the 95% confidence region. The final values for the parameters in the latitude estimation were

$$\begin{aligned} \boldsymbol{\theta}_* = [\, &0.0001586 \,,\; 0.046618 \,,\; 0.0056514 \,,\; 0.000111 \,, \\ &2056.7 \,,\; 0.069571 \,,\; 2.3315 \,] \,, \end{aligned} \tag{2.83}$$

in 83 function calls and about 94 s. The effect of each component is shown in Fig. 2.19(b) and Fig. 2.19(c) for the NCAM and Matérn ($\nu = 5/2$) model, respec-

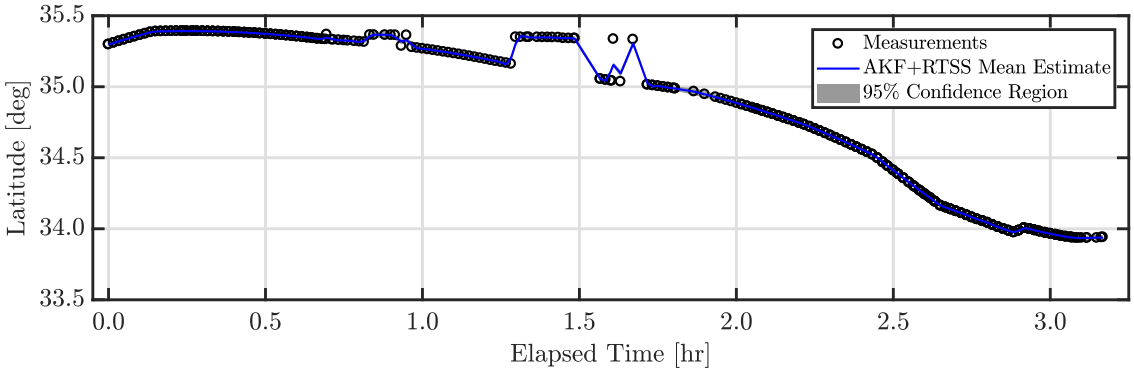
tively. The NCAM component captures the target dynamics while the Matérn model component captures the short term deviations caused by the errors.

Figure 2.20 displays the smoothed results for the longitude data. In Fig. 2.20(a), the target's true trajectory is unknown, but the circles are the sensor's noisy measurements. The blue line is the smoothed mean estimate while the gray region is the 95% confidence region. The final values for the parameters in the longitude estimation were

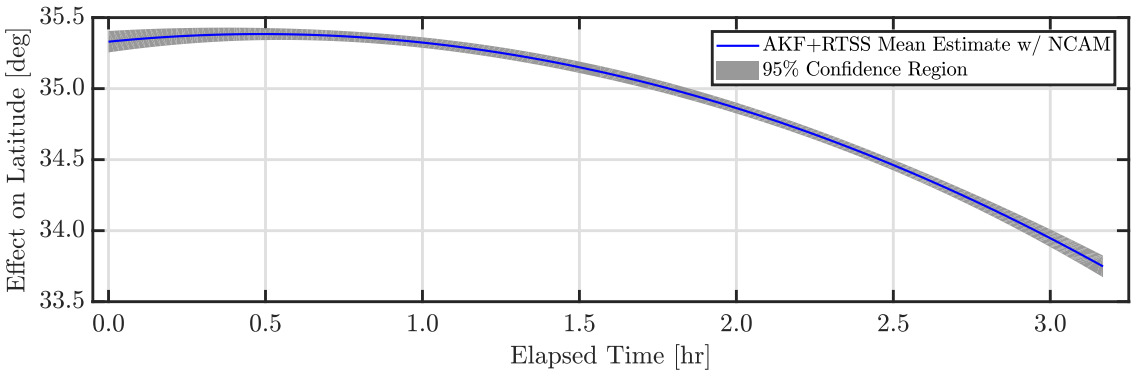
$$\begin{aligned}\boldsymbol{\theta}_* = [\,&0.21561\,,\ 0.47564\,,\ 5257.1\,,\ 9.9973\text{e}{-05}\,,\\ &0.26921\,,\ 0.054843\,,\ 0.0065388\,]\,,\end{aligned} \tag{2.84}$$

in 65 function calls and about 80 s. The effect of each component is shown in Fig. 2.20(b) and Fig. 2.20(c) for the NCAM and Matérn model, respectively. Contrary to the latitude example, the Matérn model component captures the target dynamics and the short term deviations caused by the errors while the NCAM component captures a near-zero offset. This violates the expectation that the target tracking model would account for the dynamics. The 95% confidence region in the longitude example is an order of a magnitude larger than that in the latitude example, which can be an indicator of poor regression quality.

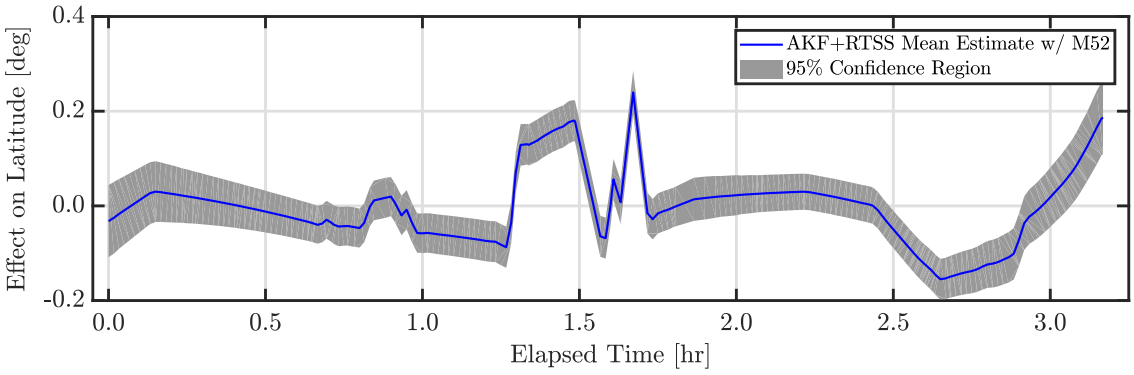
This example demonstrated the potential capability for fault detection with a joint target tracking and machine learning model. However, many application dependent decisions remain on how to consistently detect and mitigate erroneous data. Obvious approaches include outlier detection of the position and velocity data. Removing erroneous points one-by-one would be a computational slow method for correcting the data.



(a) Latitude estimate.

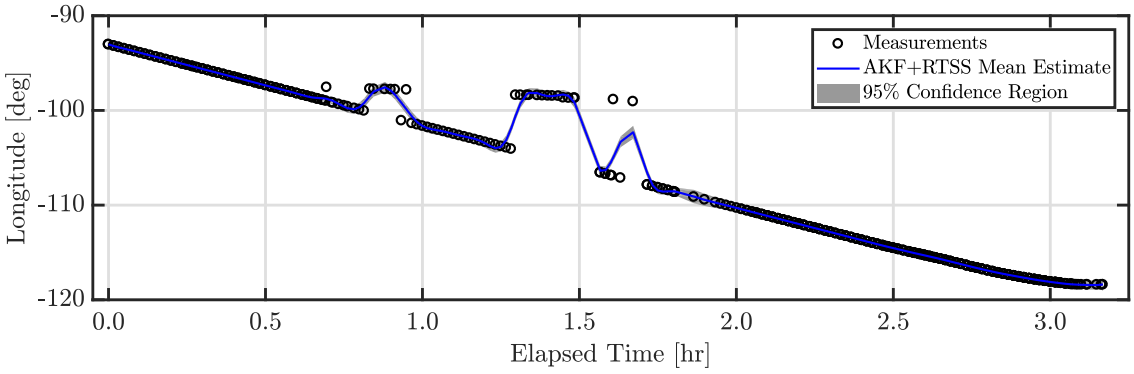


(b) Latitude component with the NCAM.

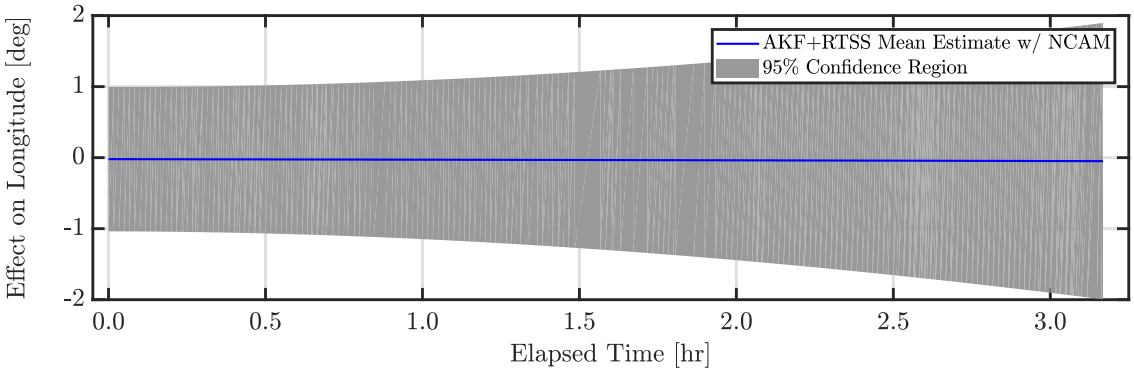


(c) Latitude component with the Matérn ($\nu = 5/2$) model.

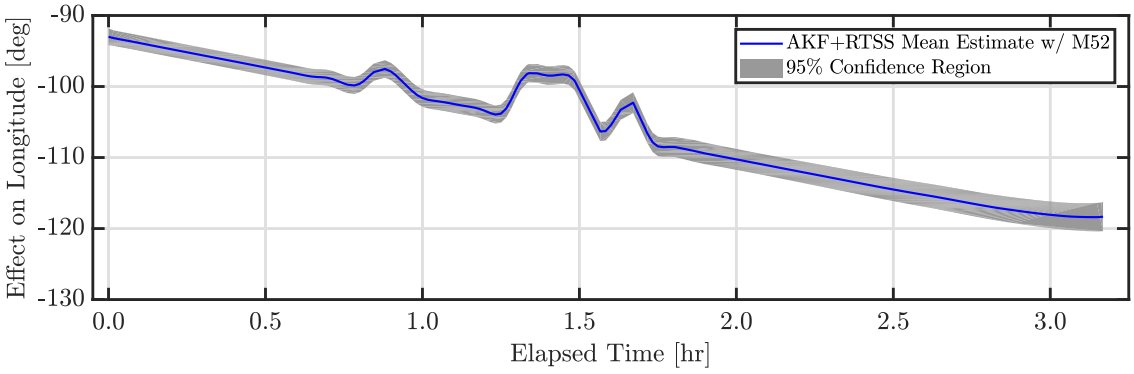
Figure 2.19. Optimal smoothed latitude estimates for the AKF and RTSS with the NCAM and Matérn ($\nu = 5/2$) model.



(a) Longitude estimate.



(b) Longitude component with the NCAM.



(c) Longitude component with the Matérn ($\nu = 5/2$) model.

Figure 2.20. Optimal smoothed longitude estimates for the AKF and RTSS with the NCAM and Matérn ($\nu = 5/2$) model.

2.3 Summary and Contributions

The objective of this chapter was to select an adaptive filter and the underlying system model. A literature review presented the following algorithms: Adaptive Gaussian Process Regression (AGPR), Kalman Filter (KF), Rauch-Tung-Striebel Smoother (RTSS), and Adaptive Kalman Filter (AKF). An AKF was presented for a parameterized state-space model (PSSM) that was compatible with machine learning models from AGPR as well as target tracking models commonly used with a KF and RTSS. The major benefit of the adaptive filtering framework over AGPR is a reduction of the computational complexity from the cubic $\mathcal{O}(T^3)$ to the linear $\mathcal{O}(T)$ for T measurements. Consequently, the next chapter utilizes a distributed formulation of this PSSM.

The relationship of machine learning and target tracking models as PSSM was discussed through a few examples. The first example consisted of tracking a target with the AKF and different discrete PSSMs. Besides the squared exponential, the covariance functions could not replace the target tracking models as a PSSM given the difficulties with convergence of the parameter estimation and the regression quality. The second example considered interception of a hypersonic boost-glide vehicle that requires an accurate, on-line calculation of the predicted trajectory given noisy measurements. Even though the data was based on a dynamical model, the squared exponential model was an order of a magnitude faster than the target tracking model. The prediction of future trajectories for interception were very poor as a result of a simplistic model. An alternative method is required to incorporate the dynamics or a surrogate model in order to enable an on-line implementation. Lastly, the joint use of machine learning and target tracking models was provided with an example of fault detection in aircraft state data. This methodology demonstrated the potential for an alternative method of fault detection and mitigation. However, many application dependent decisions remain on how to consistently detect and mitigate erroneous data.

3. DISTRIBUTED AND ADAPTIVE TARGET TRACKING

The previous chapter discussed the centralized algorithms for constructing an Adaptive Kalman Filter (AKF). The underlying model was a parameterized state-space model (PSSM) that could be defined by machine learning and target tracking models. This chapter presents a distributed PSSM for modeling a process observed by a sensor network. Then, the Adaptive Distributed Kalman Filter (ADKF) and the Adaptive Centralized Kalman Filter (ACKF) are presented as solutions to the distributed estimation problem. Furthermore, the algorithm is derived in the context of Bayesian state filters and then assessed in terms of its performance characteristics.

3.1 Literature Review

Figure 3.1 depicts the relevant algorithms to construct an ADKF. As the centralized algorithms were covered previously, this literature review will cover the Consensus Filter (CF) and Distributed Kalman Filter (DKF). For a sensor network, the CF enables sensor platforms to maintain a consensus on information while the DKF enables distributed state estimation with a sensor network's noisy measurements.

3.1.1 Consensus Filter

A Consensus Filter (CF) solves the problem of consensus within a network through local interactions. The basic idea of a CF is that consensus can be reached if each node's state converges to an average state of the neighboring nodes. In [23], Katragadda et al. compared the CF with four other fusion schemes: centralized fusion, flooding, token passing, and dynamic clustering. The CF had the highest communi-

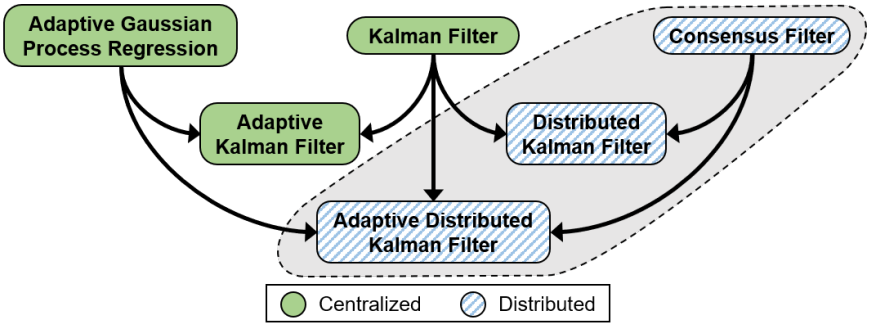


Figure 3.1. The grey region highlights the algorithms discussed within this chapter.

cation cost but provided a robust solution for consensus in a network with a dynamic topology (i.e., failures, delays, and switching signals) given a sufficient number of rounds of communication.

Consider a network of N nodes in which the state of the network is defined by the discrete-time, linear update dynamics

$$\mathbf{x}_{\ell+1} = \mathbf{W}_\ell \mathbf{x}_\ell \,, \tag{3.1}$$

where the time instance is ℓ, the state vector of the network is $\mathbf{x}_\ell \in \mathbb{R}^N$, and the state-transition matrix is $\mathbf{W}_\ell \in \mathbb{R}^{N\times N}$. A stochastic matrix is a matrix where each element is greater than zero and the sum of each row is one: $[\mathbf{W}_\ell]^{ij} = w_\ell^{ij} \geq 0$ and $\mathbf{W}_\ell \mathbf{1} = \mathbf{1}$ where $\mathbf{1} = [\,1\,,\,\cdots\,,\,1\,]^\top$. For a state-transition matrix that is a stochastic matrix, the nodes reach a consensus when the value at each node converges to the value α:

$$\lim_{\ell\to\infty} \mathbf{x}_\ell = \alpha \mathbf{1} \,. \tag{3.2}$$

Active research areas with consensus include the rate of convergence, network topology, quantization of numeric messages, and delay (due to actuation, control, communication, or computation) [24]. See [25–27] for a comprehensive discussion on the CF

and its properties. The following content presents the relevant concepts for a useful application of consensus with sensor networks.

Average consensus with the Perron matrix

A doubly stochastic matrix is a stochastic matrix with the additional property that the sum of each column is one: $\mathbf{1}^\top \mathbf{W}_\ell = \mathbf{1}^\top$. For a state-transition matrix that is a doubly stochastic matrix, the nodes solve the average consensus problem when the value at each node converges to the average initial value:

$$\alpha = \frac{1}{N}\mathbf{1}^\top \mathbf{x}_0 \,. \tag{3.3}$$

A common solution is based on the Perron matrix, which requires a review of Graph Theory.

A directed graph $\mathcal{G}$ is defined with the notation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of nodes is $\mathcal{V} = [\,1\,,\,2\,,\,\cdots\,,\,N\,]$ where N is the number of nodes. The set of directed edges is $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ in which a directed edge from node i to node j is a 2-element subset defined as $(i, j) \in \mathcal{E}$. For an undirected graph, the edge $(i, j) \in \mathcal{E}$ implies the edge $(j, i) \in \mathcal{E}$. Node j is a neighbor of node i if an edge exists from node i to node j such that

$$j \in \mathcal{N}(i) \quad \text{if } (i, j) \in \mathcal{E} \text{ and } i \neq j \,. \tag{3.4}$$

For an undirected graph, the neighbor $j \in \mathcal{N}(i)$ implies the neighbor $i \in \mathcal{N}(j)$. The degree of node i is the number of neighbors for that node. For an undirected graph, the out-degree and in-degree at node i are equal.

This dissertation considers a simple connected graph — a graph that is undirected, unweighted, and connected with no self-loops or multiple-edges. Consequently, the following definitions apply for a simple connected graph.

The adjacency matrix $\boldsymbol{\mathcal{A}} \in \mathbb{S}^{N\times N}$ represents the neighbors in the graph where each element $[\boldsymbol{\mathcal{A}}]^{ij} = a^{ij}$ is defined as

$$a^{ij} = \begin{cases} 1 & \text{if } j \in \mathcal{N}(i)\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{3.5}$$

The degree matrix $\boldsymbol{\mathcal{D}} \in \mathbb{S}^{N\times N}_{++}$ represents the degree of each node $[\boldsymbol{\mathcal{D}}]^{ij} = d^{ij}$ where

$$d^{ij} = \begin{cases} \sum_{i\neq k} a^{ik} & \text{if } i = j\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{3.6}$$

Consequently, the degree matrix is diagonal. The Laplacian matrix is a symmetric, positive semi-definite matrix $\boldsymbol{\mathcal{L}} \in \mathbb{S}^{N\times N}_{+}$ defined as

$$\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{D}} - \boldsymbol{\mathcal{A}}\,. \tag{3.7}$$

The sum of each row and column equals zero: $\mathbf{1}^\top\boldsymbol{\mathcal{L}} = \mathbf{0}^\top$ and $\boldsymbol{\mathcal{L}}\mathbf{1} = \mathbf{0}$ where the zero-vector is $\mathbf{0} = [\,0\,,\,\cdots\,,\,0\,]^\top$. The eigenvalues of $\boldsymbol{\mathcal{L}}$ in an ascending order are $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \leq 2\Delta$ where Δ is the maximum degree all nodes in of the network. The Perron matrix is a symmetric matrix $\boldsymbol{\mathcal{P}} \in \mathbb{S}^{N\times N}$ defined as

$$\boldsymbol{\mathcal{P}} = \mathbf{I} - \epsilon\boldsymbol{\mathcal{L}}\,, \tag{3.8}$$

where the consensus gain $\epsilon > 0$. The sum of each row and column equals one: $\mathbf{1}^\top\boldsymbol{\mathcal{P}} = \mathbf{1}^\top$ and $\boldsymbol{\mathcal{P}}\mathbf{1} = \mathbf{1}$. For the condition $0 < \epsilon < 1/\Delta$, the eigenvalue magnitudes of $\boldsymbol{\mathcal{P}}$ in a descending order are $1 = |\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_n|$.

A static state-transition matrix with the Perron matrix,

$$\mathbf{x}_{\ell+1} = \boldsymbol{\mathcal{P}}\mathbf{x}_\ell\,, \tag{3.9}$$

solves the average consensus problem. The local computation at node i is

$$x_{\ell+1}^i = x_\ell^i + \epsilon \sum_{j \in \mathcal{N}(i)} \left(x_\ell^j - x_\ell^i \right) , \tag{3.10}$$

where the state of each node is the element $[\mathbf{x}_\ell]^i = x_\ell^i$. For the consensus gain $0 < \epsilon < 1/\Delta$, this solution provides asymptotic convergence that is related to the second eigenvalue μ_2 of the Perron matrix [26]. However, the algorithm requires knowledge of the graph topology in order to determine a consensus gain ϵ that provides convergence in a minimal number of consensus iterations.

Example of average consensus with the Perron matrix

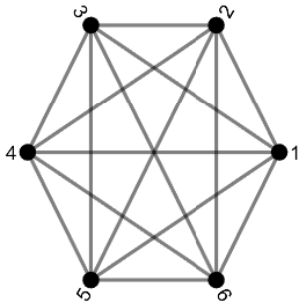
Consider a network with six sensors that each take a measurement y^i of the altitude of a target. This example solves the average consensus problem for estimating the target's altitude with two different communication topologies. Table 3.1 summarizes the design parameters as a CF with the Perron matrix and the consensus gain ϵ. The initial state vector $\mathbf{x}_0$ was populated with each measurement y^i such that

$$\mathbf{x}_0 = \begin{bmatrix} 31{,}950 & 31{,}550 & 31{,}690 & 31{,}598 & 31{,}650 & 31{,}750 \end{bmatrix}^\top \text{ ft} . \tag{3.11}$$

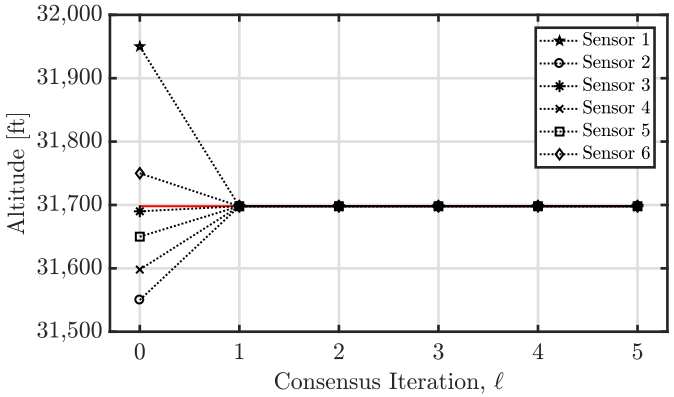
The fully and partially connected network topologies are shown in Figs. 3.2(a) and 3.2(c), respectively. A different consensus gain was utilized near the maximum bound of $1/\Delta$ for each network in order to increase the rate of convergence. The CF results are presented in Figs. 3.2(b) and 3.2(d). The red line indicates the average initial value $\alpha = 31{,}698$ ft. The fully connected network solves the average consensus problem in one iteration, but does require more iterations for other values of the consensus gain ϵ. The partially connected network solves the average consensus problem as each sensor platform's value approaches the red line. Significant convergence was achieved with five iterations, but the stopping criteria is application dependent.

Table 3.1. Design choices for an example of average consensus with the Perron matrix.

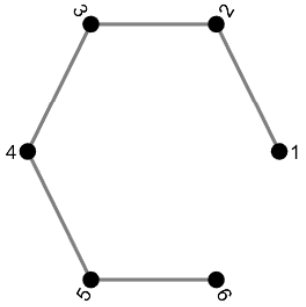
(1) **Consensus Filter** - The state-transition matrix is the Perron matrix
(2) **Consensus Gain** $\epsilon = 1/6$ for the fully connected network $\epsilon = 1/3$ for the partially connected network



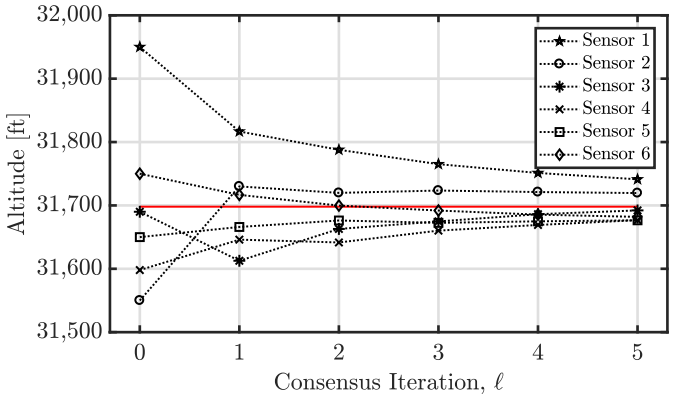
(a) Fully connected network.



(b) Average consensus for the fully connected network.



(c) Partially connected network.



(d) Average consensus for the partially connected network.

Figure 3.2. Average consensus for a target's altitude as observed by a fully and partially connected network of six sensor platforms.

Weighted average consensus with the Perron matrix

A special formulation with two CFs enables weighted average consensus by accounting for measurement uncertainty. Consequently, this methodology works with a

sensor network that contains sensor platforms with inactive sensors — sensors platforms that do not take measurements but participate in the consensus estimation routine by communicating with neighboring sensor platforms. Note, a sensor platform with an inactive sensor is not a 'repeater' since the sensor platform runs a consensus routine instead of simply relaying information.

Consider a sensor network with N sensors in which the measurements are normally distributed about the observable state $\mathbf{Hx}$ such that the measurement by sensor i is

$$y^i \sim \mathcal{N}\big(\mathbf{Hx}, \Sigma^i\big)\,. \tag{3.12}$$

The objective is to calculate a pseudo measurement as a weighted average of a set of measurements which has a smaller variance than that of each individual measurement. The pseudo measurement is

$$\hat{y} \sim \mathcal{N}\big(\mathbf{Hx}, \mathrm{Var}(\hat{y})\,\big) \tag{3.13}$$

where

$$\hat{y} = \left(\sum_{i=1}^{N} w^i\right)^{-1}\left(\sum_{i=1}^{N} w^i y^i\right), \tag{3.14a}$$

$$\mathrm{Var}(\hat{y}) = \left(\sum_{i=1}^{N} w^i\right)^{-2}\left(\sum_{i=1}^{N} \left(w^i\right)^2 \Sigma^i\right), \tag{3.14b}$$

and w^i is the weight for the i^{th} measurement. As proven by Shahar [28], the minimum variance (MV) estimate $\hat{y}_{\mathrm{MV}}$ occurs when the weight is the inverse of the variance such that

$$\hat{y}_{\mathrm{MV}} = \left(\sum_{i=1}^{N} \left[\Sigma^i\right]^{-1}\right)^{-1}\left(\sum_{i=1}^{N} \left[\Sigma^i\right]^{-1} y^i\right), \tag{3.15a}$$

$$\mathrm{Var}(\hat{y}_{\mathrm{MV}}) = \left(\sum_{i=1}^{N} \left[\Sigma^i\right]^{-1}\right)^{-1}. \tag{3.15b}$$

However, a sensor platform may not have access to all the measurements and variances to compute the summations and require a consensus algorithm instead.

The MV estimate can be calculated at each sensor platform in a network by solving two average consensus problems in parallel. The CFs for the local computation at sensor platform i are

$$a_{\ell+1}^i = a_\ell^i + \epsilon \sum_{j \in \mathcal{N}(i)} \left(a_\ell^j - a_\ell^i \right) , \tag{3.16a}$$

$$b_{\ell+1}^i = b_\ell^i + \epsilon \sum_{j \in \mathcal{N}(i)} \left(b_\ell^j - b_\ell^i \right) , \tag{3.16b}$$

where the initial values are

$$a_0^i = \left[\Sigma^i \right]^{-1} y^i , \tag{3.17a}$$

$$b_0^i = \left[\Sigma^i \right]^{-1} . \tag{3.17b}$$

a_0^i is the weighted information vector and b_0^i is the information matrix. As ℓ approaches infinity, the variables a_ℓ^i and b_ℓ^i approach the network's average of the weighted information vectors and the network's average of the information matrices, respectively. The MV estimate of the pseudo measurement is

$$\hat{y}_{\text{MV}} = \lim_{\ell \to \infty} \left[N b_\ell^i \right]^{-1} \left[N a_\ell^i \right] = \lim_{\ell \to \infty} \left[b_\ell^i \right]^{-1} a_\ell^i , \tag{3.18a}$$

$$\text{Var}(\hat{y}_{\text{MV}}) = \lim_{\ell \to \infty} \left[N b_\ell^i \right]^{-1} . \tag{3.18b}$$

After L iterations of the consensus routine, the pseudo measurement can be recovered at each sensor platform with

$$\hat{y}^i = \left[b_L^i \right]^{-1} a_L^i , \tag{3.19a}$$

$$\text{Var}\left(\hat{y}^i \right) = \left[N b_L^i \right]^{-1} . \tag{3.19b}$$

3.1.2 Distributed Kalman Filter

In a Distributed Kalman Filter (DKF), nodes communicate with neighboring nodes to achieve consensus on a state estimate throughout the entire network. Mahmoud published a survey paper [29] providing an overview of a wide range of DKF-related topics. However, that survey paper does not include the evolution of DKF algorithms as shown in Fig. 3.3 as well as some recent DKF algorithms based on the information form of the state filter. The filters can be decomposed into three groups: centralized, decentralized, and distributed. The difference between a decentralized and distributed solution is that the former requires an all-to-all communication network.

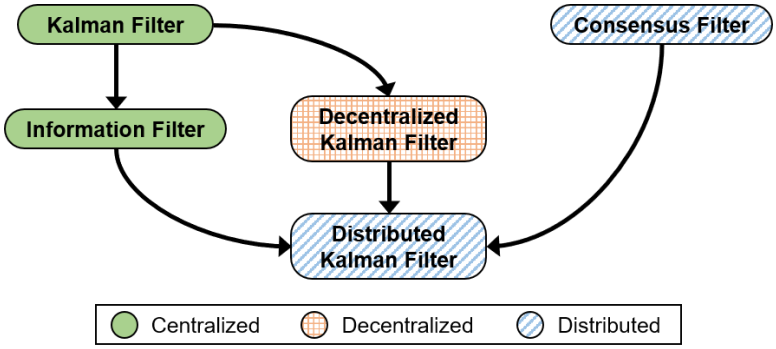


Figure 3.3. Evolution of the Distributed Kalman Filter.

For centralized filters, the Kalman Filter (KF) and Information Filter (IF) were discussed in Section 2.1.1. The first solution for a Decentralized Kalman Filter is credited to Speyer [30] as an extension of the KF for sensor networks. This algorithm contains relations to the Laplacian matrix, thus requiring all-to-all communication, which is not scalable with a communication complexity of $\mathcal{O}(N^2)$ for N nodes.

For distributed filters, the Consensus Filter (CF) was discussed in Section 3.1.1 as a method for achieving consensus on a state throughout a network. The Distributed Kalman Filter (DKF) had numerous variations originating from the CF. Many early DKF algorithms had difficulty obtaining a solution that simultaneously

provided good accuracy, rate of convergence, and scalability [31–38]. In [35], Olfati-Saber demonstrated that the optimal Kalman Consensus Filter (KCF) was not scalable in the numbers of nodes N for communication and computation, which prompted the derivation of a scalable, but sub-optimal KCF. An important development, the Generalized Kalman Consensus Filter (GKCF), added the functionality of handling naïve nodes in sparse communication topologies such as camera networks. A node is considered naïve if that node and its neighbors cannot observe the target. In [38], Kamal et al. identified three problems with the popular, sub-optimal KCF in [35]. To address the shortcomings, the GKCF modified the KCF, but the alterations improved the accuracy at the expense of the message size. A more recent development was the Information Weighted Consensus Filter (IWCF), which can calculate optimal estimates [1, 39, 40]. Kamal et al. provided another algorithm known as the Information Consensus Filter (ICF), which does not require the number of nodes N in the network but produces sub-optimal estimates [1]. Consequently, the ICF and the IWCF are good solutions for the distributed state estimation problem depending on whether N is known. However, Assimakis et al. proved that the fastest algorithm between a KF and an IF depends on the length of the state and measurement vectors [41]. This sentiment can be applied to the state and information form in a DKF. Thus, GKCF can be preferable to reduce the computational complexity at the expense of a biased estimate.

Katragadda et al. provided a table in [42] that compared DKF algorithms and their ability to handle challenges with a non-linear measurement model, naïve nodes, and redundancy in the exchanged information. Similarly, Table 3.2 presents important DKF algorithms and their properties with the addition of an Adaptive DKF (ADKF) algorithm that will be defined later in this chapter. This dissertation explicitly utilizes an ADKF based on the IWCF because the routine can compute optimal state estimates.

Table 3.2. Summary of properties for DKF and ADKF algorithms.

	Acronym	Reference	NL	NN	R	A
DKF	KCF	Olfati-Saber, 2009 [35]	-	-	-	-
	EKCF	Ding, 2012 [36]	✓	-	-	-
	GKCF	Kamal, 2011 [38]	-	✓	-	-
	ICF	Kamal, 2013 [1]	-	✓	-	-
	IWCF	Casbeer, 2009 [39]	-	✓	✓	-
	EICF	Katragadda, 2014 [42]	✓	✓	-	-
	EIWCF	Katragadda, 2014 [42]	✓	✓	✓	-
ADKF	AIWCF		-	✓	✓	✓

Properties: NL - non-linear models, NN - naïve nodes
R - redundant information, A - adaptive models

Information Weighted Consensus Filter

Consider a dynamic process measured by a sensor network defined with the distributed state-space model (DSSM) as

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}\,, \qquad \mathbf{q}_{k-1} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{Q}_{k-1}\big)\,, \tag{3.20a}$$

$$\mathbf{y}_k^i = \mathbf{H}_k^i\mathbf{x}_k + \mathbf{r}_k^i\,, \qquad \mathbf{r}_k^i \sim \mathcal{N}\big(\mathbf{0}, \mathbf{R}_k^i\big)\,, \tag{3.20b}$$

or with the probabilistic notation as

$$p\big(\mathbf{x}_k \mid \mathbf{x}_{k-1}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \mathbf{A}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}\big)\,, \tag{3.21a}$$

$$p\big(\mathbf{y}_k^i \mid \mathbf{x}_k\big) = \mathcal{N}\big(\mathbf{y}_k^i \mid \mathbf{H}_k^i\mathbf{x}_k, \mathbf{R}_k^i\big)\,, \tag{3.21b}$$

for the time instance $k \in [\,1\,, \cdots, T\,]$ and sensor $i \in [\,1\,, \cdots, N\,]$. For the process dynamics, $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector, $\mathbf{A}_k$ is the process matrix, and $\mathbf{q}_k$ is the process noise with covariance $\mathbf{Q}_k$. For the measurement model, $\mathbf{y}_k^i \in \mathbb{R}^m$ is the measurement,

$\mathbf{H}_k^i$ is the measurement matrix, and $\mathbf{r}_k^i$ is the measurement noise with covariance $\mathbf{R}_k^i$. The noise terms are zero-mean white Gaussian noise such that

$$\mathbb{E}\big[\mathbf{q}_k(\mathbf{q}_\ell)^\top\big] = \mathbf{Q}_k\delta_{k\ell}\,, \tag{3.22a}$$

$$\mathbb{E}\big[\mathbf{r}_k^i(\mathbf{r}_\ell^j)^\top\big] = \mathbf{R}_k^i\delta_{k\ell}\delta_{ij}\,, \tag{3.22b}$$

where $\delta_{k\ell}$ and δ_{ij} are the Kronecker delta function.

The state is hidden, but the measurements are observed locally. The IWCF consists of three components: consensus, update, and predict [1]. The filtering and predictive distributions are denoted as

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}^{1:N}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \bar{\mathbf{x}}_k, \bar{\mathbf{P}}_k\big)\,, \tag{3.23a}$$

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k}^{1:N}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \widetilde{\mathbf{x}}_k, \widetilde{\mathbf{P}}_k\big)\,. \tag{3.23b}$$

The *a priori* state estimate and covariance are $\bar{\mathbf{x}}_k^i \in \mathbb{R}^n$ and $\bar{\mathbf{P}}_k^i \in \mathbb{R}^{n\times n}$, respectively. The *a posteriori* state estimate and covariance are $\widetilde{\mathbf{x}}_k^i \in \mathbb{R}^n$ and $\widetilde{\mathbf{P}}_k^i \in \mathbb{R}^{n\times n}$, respectively. However, the IWCF often utilizes the information form in which the transformation was provided in Eqs. (2.10) and (2.11). In the prediction step, the *a priori* statistics are often more convenient in the mixed notation

$$\bar{\mathbf{x}}_k^i = \mathbf{A}_{k-1}\widetilde{\mathbf{x}}_{k-1}^i\,, \tag{3.24a}$$

$$\bar{\mathbf{Z}}_k^i = \left(\mathbf{A}_{k-1}\left[\widetilde{\mathbf{Z}}_{k-1}^i\right]^{-1}\mathbf{A}_{k-1}^\top + \mathbf{Q}_{k-1}\right)^{-1}\,, \tag{3.24b}$$

where the information matrix is $\bar{\mathbf{Z}}_k^i \in \mathbb{R}^{n\times n}$. In the update step, the *a posteriori* statistics are computed with the weighted average consensus routine. At time instance k,

the CF has L iterations to try and achieve consensus. The CFs for the local computation at sensor platform i are

$$\mathbf{w}_{k,\ell+1}^{i} = \mathbf{w}_{k,\ell}^{i} + \epsilon \sum_{j \in \mathcal{N}(i)} \left[\mathbf{w}_{k,\ell}^{j} - \mathbf{w}_{k,\ell}^{i} \right] , \tag{3.25a}$$

$$\mathbf{W}_{k,\ell+1}^{i} = \mathbf{W}_{k,\ell}^{i} + \epsilon \sum_{j \in \mathcal{N}(i)} \left[\mathbf{W}_{k,\ell}^{j} - \mathbf{W}_{k,\ell}^{i} \right] , \tag{3.25b}$$

where the initial values are

$$\mathbf{w}_{k,0}^{i} = \frac{1}{N} \bar{\mathbf{Z}}_{k}^{i} \bar{\mathbf{x}}_{k}^{i} + \mathbf{i}_{k}^{i} , \tag{3.26a}$$

$$\mathbf{W}_{k,0}^{i} = \frac{1}{N} \bar{\mathbf{Z}}_{k}^{i} + \mathbf{I}_{k}^{i} , \tag{3.26b}$$

and the local information is

$$\mathbf{i}_{k}^{i} = \left[\mathbf{H}_{k}^{i} \right]^{\top} \left[\mathbf{R}_{k}^{i} \right]^{-1} \mathbf{y}_{k}^{i} , \tag{3.27a}$$

$$\mathbf{I}_{k}^{i} = \left[\mathbf{H}_{k}^{i} \right]^{\top} \left[\mathbf{R}_{k}^{i} \right]^{-1} \mathbf{H}_{k}^{i} . \tag{3.27b}$$

The intuition behind Eq. (3.26) is that the information vector and matrix are updated with new information that is inversely proportional to the number of sensors in the network N. Consequently, $\mathbf{w}_{k,0}^{i}$ is the updated weighted information vector and $\mathbf{W}_{k,0}^{i}$ is the updated information matrix. Note, $\mathbf{w}_{k,0}^{i}$ and $\mathbf{W}_{k,0}^{i}$ are a measure of the amount of information of a state estimate while a_{0}^{i} and b_{0}^{i} are a measure of the amount of information of a measurement. As ℓ approaches infinity, the variables $\mathbf{w}_{k,\ell}^{i}$ and $\mathbf{W}_{k,\ell}^{i}$ approach the network's average of the updated weighted information vectors and the network's average of the updated information matrices, respectively. The state estimate can be recovered at each sensor platform after L consensus iterations with

$$\widetilde{\mathbf{x}}_{k}^{i} = \left[\mathbf{W}_{k,L}^{i} \right]^{-1} \mathbf{w}_{k,L}^{i} , \tag{3.28a}$$

$$\widetilde{\mathbf{Z}}_{k}^{i} = N \mathbf{W}_{k,L}^{i} , \tag{3.28b}$$

where the information matrix is $\widetilde{\mathbf{Z}}_k^i \in \mathbb{R}^{n\times n}$.

Example of the Information Weighted Consensus Filter

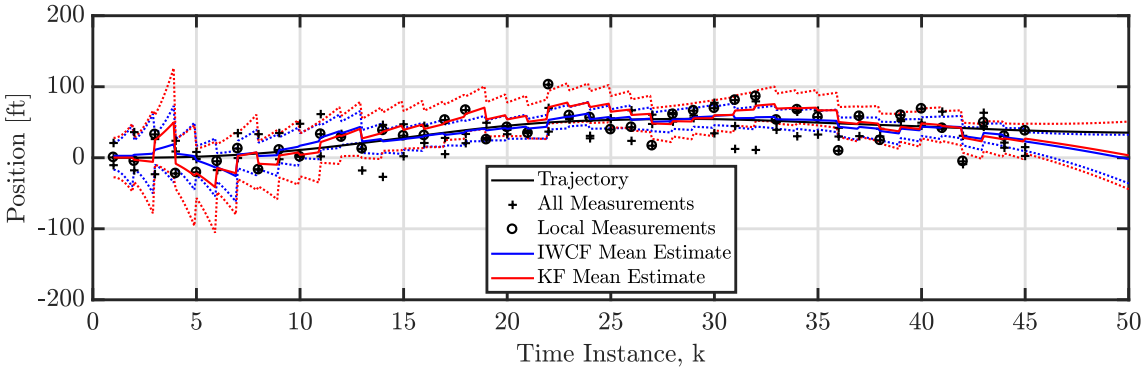
To demonstrate the implementation of the IWCF, an example is provided for estimating the position of a target. The data set was a random sample of the position for the NCAM in Eq. (2.58) with the sampling rate $\Delta t_k = 1$ s and power spectral density $q_c = 0.01$ ft^2/s^5. The measurement model noise for sensor i had covariance $\mathbf{R}_k^i = [\sigma_n^i]^2$ where the standard deviation of the measurement noise was $\sigma_n^i = 20$ ft for all sensors. The design choices for the solution are summarized in Table 3.3 for a fully connected network with three sensors. This solution assumed perfect knowledge of the constants in the dynamical model. The distributed state estimator was the IWCF and RTSS. The CF had the consensus gain $\epsilon = 1/4$ and $L = 10$ iterations for achieving consensus at each time instance k. For comparison, the centralized state estimator was also run locally at each sensor platform with the KF and RTSS.

Table 3.3. Design choices for an example of the Information Weighted Consensus Filter.

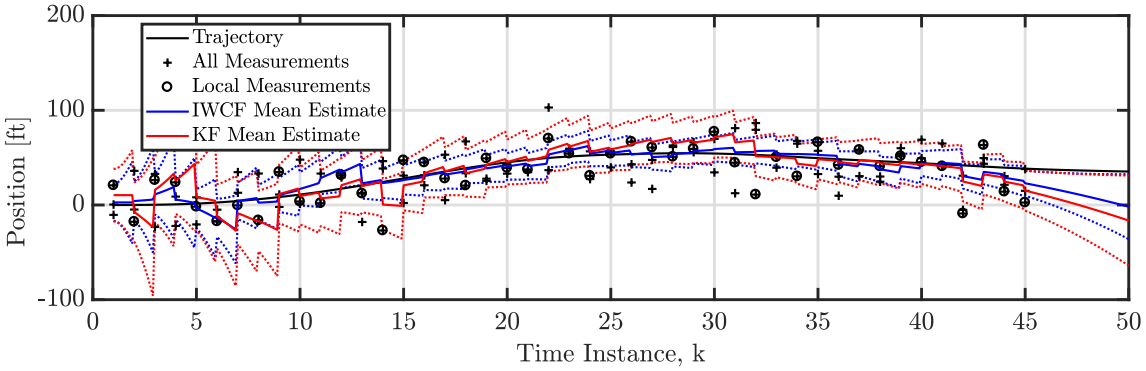
(1) **State-Space Model** - NCAM
(2) **Constants** - $\Delta t_k = 1$ s, $q_c = 0.01$ ft^2/s^5, and $\sigma_n^i = 20$ ft
(3) **State Estimation** (a) IWCF and RTSS (b) KF and RTSS
(4) **Consensus Filter** - Perron matrix with $\epsilon = 1/4$ and $L = 10$

The filtered results are displayed in Fig. 3.4 while the smoothed results are displayed in Fig. 3.5. The black line is the target's true trajectory. The crosses are all the measurements of the sensor network while the circles are the sensor's local measurements. The blue line is the mean estimate of the IWCF with the 95% confidence region represented by the blue dotted line. The red line is the mean estimate of the

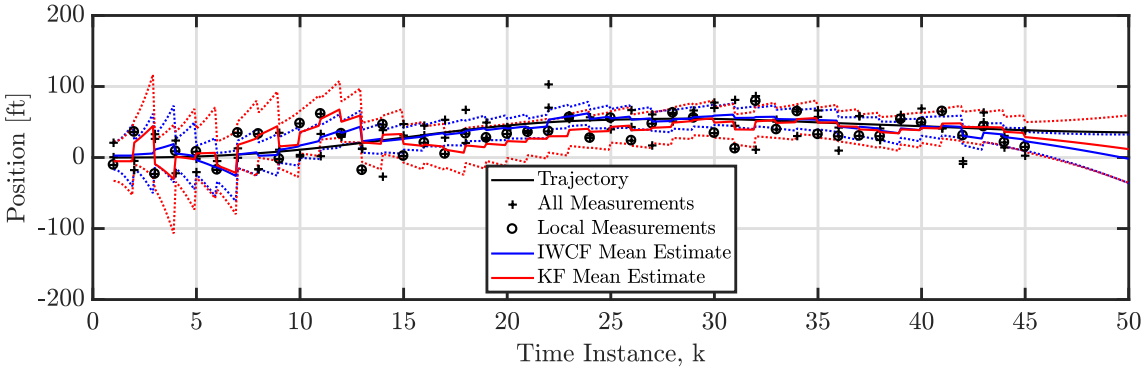
KF with the 95% confidence region represented by the red dotted line. With IWCF, each sensor platform calculates the same filtered estimate. However, the KF utilizes local information which results in different filtered estimates at each sensor platform. These same trends exist after the RTSS with the smoothed estimates. Furthermore, the smoothed estimates near $k = [\,45\,,\,50\,]$ demonstrate that a highly maneuverable target can be difficult to predict in the near future.



(a) Filtered estimates at sensor platform 1.

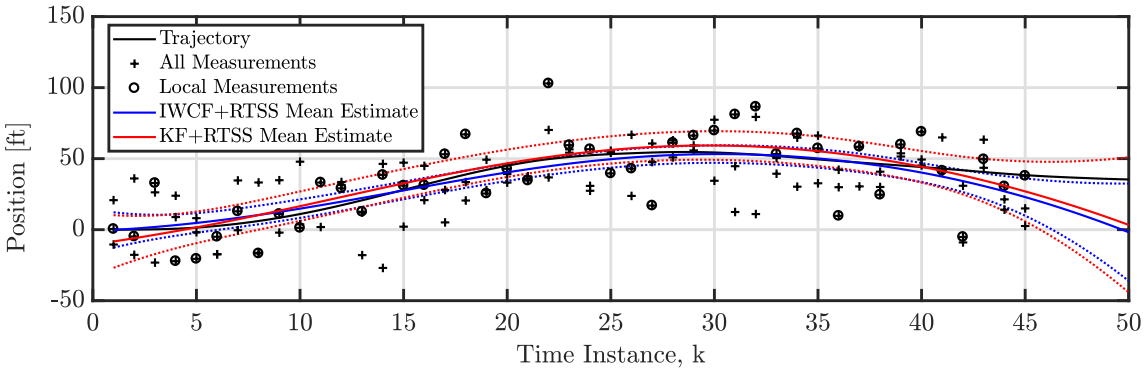


(b) Filtered estimates at sensor platform 2.

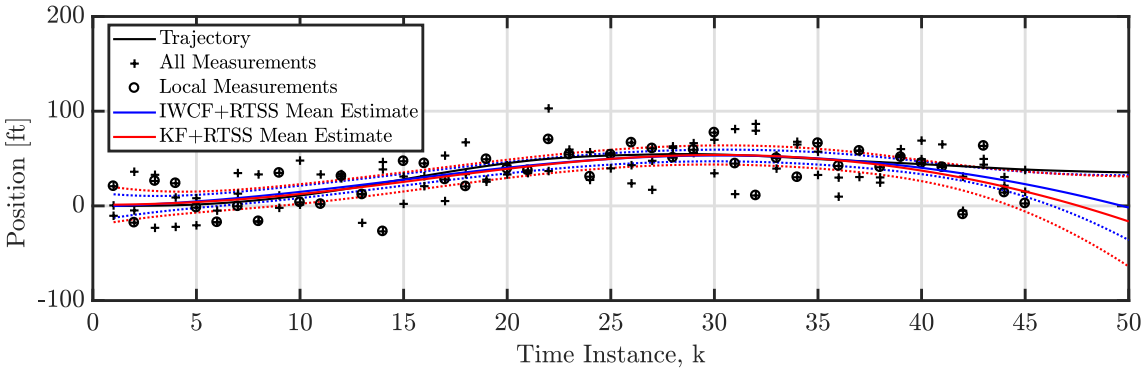


(c) Filtered estimates at sensor platform 3.

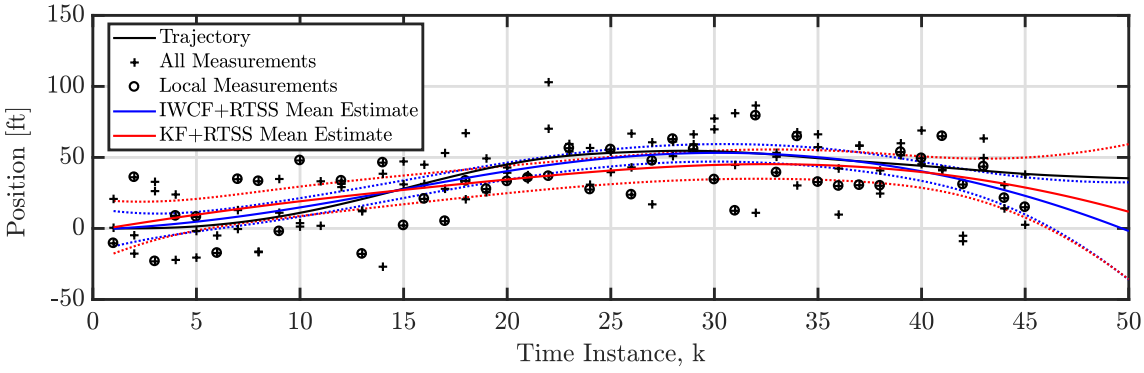
Figure 3.4. Filtered estimates for the Information Weighted Consensus Filter and Kalman Filter.



(a) Smoothed estimates at sensor platform 1.



(b) Smoothed estimates at sensor platform 2.



(c) Smoothed estimates at sensor platform 3.

Figure 3.5. Smoothed estimates for the Information Weighted Consensus Filter and Kalman Filter with a Rauch-Tung-Striebel Smoother.

3.2 Parameterized Distributed State-Space Model

Consider a dynamic process measured by a sensor network defined with the parameterized distributed state-space model (PDSSM) as

$$\mathbf{x}_k = \mathbf{A}_{k-1}(\boldsymbol{\theta})\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1}\,, \qquad \mathbf{q}_{k-1} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{Q}_{k-1}(\boldsymbol{\theta})\big)\,, \tag{3.29a}$$

$$\mathbf{y}_k^i = \mathbf{H}_k^i(\boldsymbol{\theta})\,\mathbf{x}_k + \mathbf{r}_k^i\,, \qquad \mathbf{r}_k^i \sim \mathcal{N}\big(\mathbf{0}, \mathbf{R}_k^i(\boldsymbol{\theta})\big)\,, \tag{3.29b}$$

or with the probabilistic notation as

$$p\big(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \mathbf{A}_{k-1}(\boldsymbol{\theta})\,\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}(\boldsymbol{\theta})\big)\,, \tag{3.30a}$$

$$p\big(\mathbf{y}_k^i \mid \mathbf{x}_k, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{y}_k^i \mid \mathbf{H}_k^i(\boldsymbol{\theta})\,\mathbf{x}_k, \mathbf{R}_k^i(\boldsymbol{\theta})\big)\,, \tag{3.30b}$$

for the time instance $k \in [\,1\,,\,\cdots\,,\,T\,]$, sensor $i \in [\,1\,,\,\cdots\,,\,N\,]$, and parameter $\boldsymbol{\theta} \in \mathbb{R}^p$. For the process model, $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector, $\mathbf{A}_k(\boldsymbol{\theta})$ is the process matrix, and $\mathbf{q}_k$ is the process noise with covariance $\mathbf{Q}_k(\boldsymbol{\theta})$. For the measurement model, $\mathbf{y}_k^i \in \mathbb{R}^m$ is the measurement, $\mathbf{H}_k^i(\boldsymbol{\theta})$ is the measurement matrix, and $\mathbf{r}_k^i$ is the measurement noise with covariance $\mathbf{R}_k^i(\boldsymbol{\theta})$. The noise terms are zero-mean white Gaussian noise such that

$$\mathbb{E}\big[\mathbf{q}_k(\mathbf{q}_\ell)^\top\big] = \mathbf{Q}_k(\boldsymbol{\theta})\,\delta_{k\ell}\,, \tag{3.31a}$$

$$\mathbb{E}\big[\mathbf{r}_k^i(\mathbf{r}_\ell^j)^\top\big] = \mathbf{R}_k^i(\boldsymbol{\theta})\,\delta_{k\ell}\delta_{ij}\,, \tag{3.31b}$$

where $\delta_{k\ell}$ and δ_{ij} are the Kronecker delta function. The PDSSM has the following two properties: Markov property of states and conditional independence of measurements.

Definition 3.2.1 *The states $\mathbf{x}_{0:T}$ are a Markov sequence, which means the next state $\mathbf{x}_{k+1}$ given the parameter $\boldsymbol{\theta}$ and the current state $\mathbf{x}_k$ is independent of the previous states and measurements such that*

$$p\big(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}^{1:N}, \boldsymbol{\theta}, \mathbf{x}_{0:k}\big) = p\big(\mathbf{x}_{k+1} \mid \boldsymbol{\theta}, \mathbf{x}_k\big)\,. \tag{3.32}$$

Definition 3.2.2 *For sensor i, the measurements $\mathbf{y}^i_{1:T}$ are conditionally independent,*

$$p\big(\mathbf{y}^i_{1:T} \mid \boldsymbol{\theta}, \mathbf{x}_{0:T}\big) = \prod_{k=1}^{T} p\big(\mathbf{y}^i_k \mid \boldsymbol{\theta}, \mathbf{x}_k\big)\,, \tag{3.33}$$

which means the current measurement $\mathbf{y}^i_k$ given the current state $\mathbf{x}_k$ is conditionally independent of the past measurements $\mathbf{y}^i_{1:k-1}$ such that

$$p\big(\mathbf{y}^i_k \mid \mathbf{y}^i_{1:k-1}, \boldsymbol{\theta}, \mathbf{x}_{0:k}\big) = p\big(\mathbf{y}^i_k \mid \boldsymbol{\theta}, \mathbf{x}_k\big)\,. \tag{3.34}$$

3.3 Approach

The states and parameters are hidden while the measurements are observed locally. The sensor platforms broadcast and receive messages in order to maintain consensus on state and parameter estimates. The communication network is a simple connected graph — a graph that is undirected, unweighted, and connected with no self-loops or multiple-edges. The difficulty with the consensus estimation is determining which data to share considering potential limitations of the communication network with the messages size and broadcast rate. The message could explicitly contain raw measurements, state estimates, or parameter estimates. Otherwise, the message could include this data in a transformation such as in the IWCF with the information vector and information matrix.

Figure 3.6 displays the indices for the parameter, state, and consensus estimation routines for a set of measurements. The measurement process consists of the sensors taking the measurements $\mathbf{y}^{1:N}_{1:T}$. This research does not address the real-time implementation by considering future measurements (i.e., $T \to T+1$). The parameter estimation utilizes the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm as a gradient-based optimization routine, which converges with a function count of D. The parameter is a design variable and is denoted independent of the indice for the function call $d \in [\,0\,,\,\cdots\,,\,D-1\,]$ without losing any meaning. The parameter estimation requires filtered or smoothed state estimates depending on the

objective function in the training phase and the desired type of estimate in the prediction phase. For the state estimation, a filter makes a forward pass over the T measurements while a smoother makes a backward pass conditioning all of the state estimates on all of the measurements. This research considers using CFs to maintain consensus on the weighted average of the measurements (i.e., pseudo measurements) and the weighted average of the state estimates. At time instance k, the communication network is physically limited to L iterations to achieve consensus.

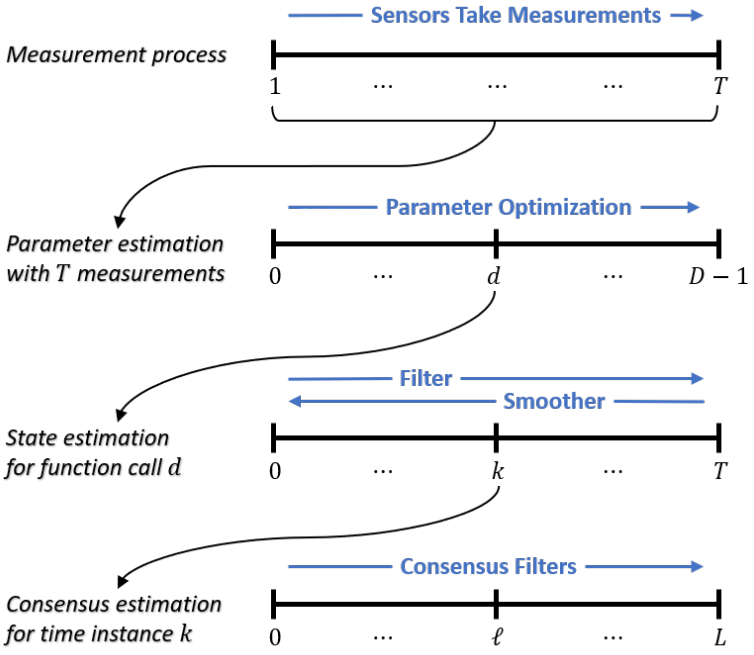


Figure 3.6. Indices of the parameter, state, and consensus estimation routines for a given set of measurements.

The state and parameter estimation routines could utilize pseudo measurements instead of explicitly using all of the measurements. At time instance k, the pseudo measurement for the PDSSM based on the MV estimation method is

$$\begin{aligned}\hat{\mathbf{y}}_{k,\mathrm{MV}} &= \left(\sum_{i=1}^{N} \left[\mathbf{R}_k^i(\boldsymbol{\theta})\right]^{-1} \right)^{-1} \sum_{i=1}^{N} \left[\mathbf{R}_k^i(\boldsymbol{\theta})\right]^{-1} \mathbf{y}_k^i \,, \\ \widehat{\mathbf{R}}_k^i(\boldsymbol{\theta}) &= \left(\sum_{i=1}^{N} \left[\mathbf{R}_k^i(\boldsymbol{\theta})\right]^{-1} \right)^{-1} . \end{aligned} \tag{3.35}$$

Sensor platform $i \in [\,1\,,\,\cdots\,,\,N\,]$ can utilize two CFs to calculate the pseudo measurement where the pseudo measurement model is

$$p\big(\hat{\mathbf{y}}_k^i \mid \mathbf{x}_k, \boldsymbol{\theta}\big) = \mathcal{N}\big(\hat{\mathbf{y}}_k^i \mid \mathbf{H}_k^i(\boldsymbol{\theta})\,\mathbf{x}_k, \widehat{\mathbf{R}}_k^i(\boldsymbol{\theta})\,\big)\,. \tag{3.36}$$

Consequently, the PDSSM also has the following two properties: Markov property of states and conditional independence of pseudo measurements.

Definition 3.3.1 *The states $\mathbf{x}_{0:T}$ are a Markov sequence, which means the next state $\mathbf{x}_{k+1}$ given the parameter $\boldsymbol{\theta}$ and the current state $\mathbf{x}_k$ is independent of the previous states and the pseudo measurements such that*

$$p\big(\mathbf{x}_{k+1} \mid \hat{\mathbf{y}}_{1:k}^i, \boldsymbol{\theta}, \mathbf{x}_{0:k}\big) = p\big(\mathbf{x}_{k+1} \mid \boldsymbol{\theta}, \mathbf{x}_k\big)\,. \tag{3.37}$$

Definition 3.3.2 *For sensor i, the pseudo measurements $\hat{\mathbf{y}}_{1:T}^i$ are conditionally independent,*

$$p\big(\hat{\mathbf{y}}_{1:T}^i \mid \boldsymbol{\theta}, \mathbf{x}_{0:T}\big) = \prod_{k=1}^{T} p\big(\hat{\mathbf{y}}_k^i \mid \boldsymbol{\theta}, \mathbf{x}_k\big)\,, \tag{3.38}$$

which means the current pseudo measurement $\hat{\mathbf{y}}_k^i$ given the current state $\mathbf{x}_k$ is conditionally independent of the past pseudo measurements $\hat{\mathbf{y}}_{1:k-1}^i$ such that

$$p\big(\hat{\mathbf{y}}_k^i \mid \hat{\mathbf{y}}_{1:k-1}^i, \boldsymbol{\theta}, \mathbf{x}_{0:k}\big) = p\big(\hat{\mathbf{y}}_k^i \mid \boldsymbol{\theta}, \mathbf{x}_k\big)\,. \tag{3.39}$$

However, the consensus estimation is nested inside the state estimation, which is nested inside the parameter estimation. The communication cost can be significantly reduced if the consensus routines for the pseudo measurements are not rerun if near-optimal state and parameter estimates can still be computed.

The following content introduces and compares two solutions to solve the distributed state and parameter estimation problem. In the first method, each sensor platform runs an AKF with pseudo measurements that are based on all of the network's measurements. This solution is referred to as an Adaptive Centralized Kalman Filter (ACKF) in order to differentiate the technique from utilizing an AKF with local measurements. The second method is an adaptive formulation of a DKF and more specifically the IWCF. This method is called an Adaptive Distributed Kalman Filter (ADKF). These two methods were selected to highlight the difference between utilizing a KF and DKF for the state estimator. The following derivation for each algorithm starts with the state and parameter estimation, which can be viewed as a centralized or decentralized solution in that all the necessary information is available for the computation. Then, the distributed solution can be determined by incorporating the consensus estimation routine to achieve the same result through communications with neighboring sensor platforms. In this work, every summation is simply replaced with a CF and scaled by the number of sensors N.

3.4 Adaptive Centralized Kalman Filter

The strategy for the Adaptive Centralized Kalman Filter (ACKF) consists of running an AKF at each sensor platform with the pseudo measurement for each time instance. Recall, the pseudo measurement is a weighted average of a set of measurements which has a smaller variance than that of each individual measurement. Figure 3.7 depicts a flow diagram of the ACKF at one sensor platform. This process occurs at each sensor platform, with the intention that all sensor platforms in the network reach a consensus on state and parameter estimates.

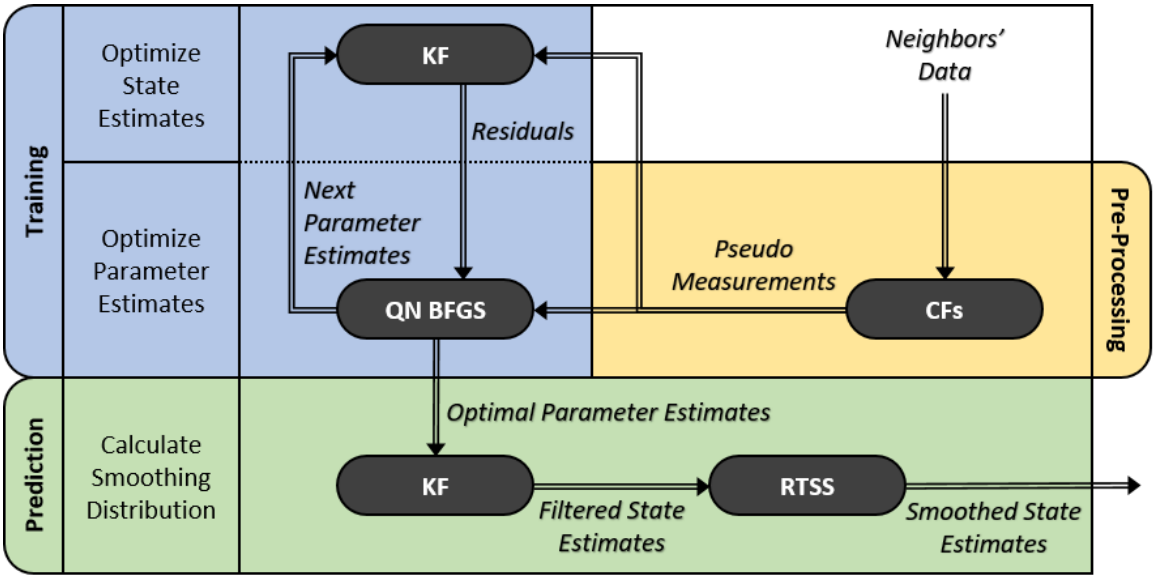


Figure 3.7. Flow Diagram for an Adaptive Centralized Kalman Filter at one sensor platform.

In the pre-processing, two CFs can calculate the pseudo measurement with the method in Section 3.1.1. After the pre-processing, the state and parameter estimation occurs by training a model and calculating the predicted values at target locations. During the training phase, the algorithm alternates between optimizing state and parameter estimates. The state estimation depends upon a KF calculating the predictive and filtering distributions. Determining the maximum a posteriori (MAP) parameter estimate depends upon maximizing the marginal posterior distribution of parameters. For gradient-based optimization with the quasi-Newton BFGS algorithm, the objective function is a byproduct of the KF while the gradient can be determined via termwise differentiation [2, 43]. With the MAP parameter estimate, the prediction phase involves a KF and RTSS calculating the smoothing distribution at target locations. The following subsections will explain the state, parameter, and consensus estimation routines.

3.4.1 State estimation

The state estimator calculates the predictive and filtering distributions as the Gaussian distributions

$$p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \bar{\mathbf{x}}^i_k(\boldsymbol{\theta}), \bar{\mathbf{P}}^i_k(\boldsymbol{\theta})\big), \tag{3.40a}$$

$$p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}^i_{1:k}, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \widetilde{\mathbf{x}}^i_k(\boldsymbol{\theta}), \widetilde{\mathbf{P}}^i_k(\boldsymbol{\theta})\big). \tag{3.40b}$$

The algorithm is recursive; the filtering distribution appears in the predictive distribution and vice versa. The recursion starts from the prior mean and covariance

$$\mathbf{x}_0 \sim \mathcal{N}\big(\widetilde{\mathbf{x}}^i_0(\boldsymbol{\theta}), \widetilde{\mathbf{P}}^i_0(\boldsymbol{\theta})\big). \tag{3.41}$$

In the predict step, the predictive distribution is calculated by marginalizing out the previous state $\mathbf{x}_{k-1}$ from the joint distribution of the current state $\mathbf{x}_k$ and the previous state $\mathbf{x}_{k-1}$ such that

$$p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big) = \int p\big(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big)\mathrm{d}\mathbf{x}_{k-1}. \tag{3.42}$$

Given the Markov property of states in Definition 3.3.1, the predictive distribution can be rewritten as

$$p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big) = \int p\big(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big) p\big(\mathbf{x}_{k-1} \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big)\mathrm{d}\mathbf{x}_{k-1} \tag{3.43a}$$

$$= \int p\big(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}\big) p\big(\mathbf{x}_{k-1} \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big)\mathrm{d}\mathbf{x}_{k-1}. \tag{3.43b}$$

The *a priori* statistics can then be calculated by evaluating the predictive distribution. According to Lemma A.2.1 in the appendix, the joint distribution of $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$ is

$$\begin{bmatrix} \mathbf{x}_{k-1} \\ \mathbf{x}_k \end{bmatrix} \sim \mathcal{N}\big(\boldsymbol{\mu}_{\text{temp\_1}}, \boldsymbol{\Sigma}_{\text{temp\_1}}\big), \tag{3.44}$$

where

$$\boldsymbol{\mu}_{\text{temp\_1}} = \begin{bmatrix} \widetilde{\mathbf{x}}_{k-1}(\boldsymbol{\theta}) \\ \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{x}}_{k-1}(\boldsymbol{\theta}) \end{bmatrix} , \tag{3.45a}$$

$$\boldsymbol{\Sigma}_{\text{temp\_1}} = \begin{bmatrix} \widetilde{\mathbf{P}}_{k-1}(\boldsymbol{\theta}) & \widetilde{\mathbf{P}}_{k-1}(\boldsymbol{\theta})\, \mathbf{A}_{k-1}^{\top}(\boldsymbol{\theta}) \\ \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{P}}_{k-1}(\boldsymbol{\theta}) & \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{P}}_{k-1}(\boldsymbol{\theta})\, \mathbf{A}_{k-1}^{\top}(\boldsymbol{\theta}) + \mathbf{Q}_{k-1}(\boldsymbol{\theta}) \end{bmatrix} . \tag{3.45b}$$

According to Lemma A.2.2 in the appendix, the predictive distribution is the marginal distribution of $\mathbf{x}_k$ denoted as

$$p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}_{1:k-1}^{i}, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \bar{\mathbf{x}}_k^i(\boldsymbol{\theta}) \,, \bar{\mathbf{P}}_k^i(\boldsymbol{\theta})\big) \,, \tag{3.46}$$

where

$$\bar{\mathbf{x}}_k^i(\boldsymbol{\theta}) = \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{x}}_{k-1}^i(\boldsymbol{\theta}) \ , \tag{3.47a}$$

$$\bar{\mathbf{P}}_k^i(\boldsymbol{\theta}) = \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{P}}_{k-1}^i(\boldsymbol{\theta})\, \mathbf{A}_{k-1}^{\top}(\boldsymbol{\theta}) + \mathbf{Q}_{k-1}(\boldsymbol{\theta}) \ . \tag{3.47b}$$

In the update step, the filtering distribution is calculated with Bayes' theorem as

$$p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}_{1:k}^{i}, \boldsymbol{\theta}\big) = \frac{p\big(\hat{\mathbf{y}}_k^i \mid \mathbf{x}_k, \boldsymbol{\theta}\big) p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}_{1:k-1}^{i}, \boldsymbol{\theta}\big)}{p\big(\hat{\mathbf{y}}_k^i \mid \hat{\mathbf{y}}_{1:k-1}^{i}, \boldsymbol{\theta}\big)} \,. \tag{3.48}$$

According to Lemma A.2.1 in the appendix, the joint distribution of $\mathbf{x}_k$ and $\hat{\mathbf{y}}_k^i$ is

$$\begin{bmatrix} \mathbf{x}_k \\ \hat{\mathbf{y}}_k^i \end{bmatrix} \sim \mathcal{N}\big(\boldsymbol{\mu}_{\text{temp\_2}}, \boldsymbol{\Sigma}_{\text{temp\_2}}\big) \,, \tag{3.49}$$

where

$$\boldsymbol{\mu}_{\text{temp\_2}} = \begin{bmatrix} \bar{\mathbf{x}}_k(\boldsymbol{\theta}) \\ \mathbf{H}_k^i(\boldsymbol{\theta})\, \bar{\mathbf{x}}_k(\boldsymbol{\theta}) \end{bmatrix} , \tag{3.50a}$$

$$\boldsymbol{\Sigma}_{\text{temp\_2}} = \begin{bmatrix} \bar{\mathbf{P}}_k(\boldsymbol{\theta}) & \bar{\mathbf{P}}_k(\boldsymbol{\theta})\, [\mathbf{H}_k^i(\boldsymbol{\theta})]^\top \\ \mathbf{H}_k^i(\boldsymbol{\theta})\, \bar{\mathbf{P}}_k(\boldsymbol{\theta}) & \mathbf{H}_k^i(\boldsymbol{\theta})\, \bar{\mathbf{P}}_k(\boldsymbol{\theta})\, [\mathbf{H}_k^i(\boldsymbol{\theta})]^\top + \widehat{\mathbf{R}}_k^i(\boldsymbol{\theta}) \end{bmatrix} . \tag{3.50b}$$

According to Lemma A.2.2 in the appendix, the filtering distribution is the conditional distribution of $\mathbf{x}_k$ denoted as

$$p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}_{1:k}^i, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \widetilde{\mathbf{x}}_k^i(\boldsymbol{\theta}) \,, \widetilde{\mathbf{P}}_k^i(\boldsymbol{\theta})\big) \,, \tag{3.51}$$

where

$$\mathbf{v}_k^i(\boldsymbol{\theta}) = \hat{\mathbf{y}}_k^i - \mathbf{H}_k^i(\boldsymbol{\theta})\, \bar{\mathbf{x}}_k^i(\boldsymbol{\theta}) \ , \tag{3.52a}$$

$$\mathbf{S}_k^i(\boldsymbol{\theta}) = \mathbf{H}_k^i(\boldsymbol{\theta})\, \bar{\mathbf{P}}_k^i(\boldsymbol{\theta})\, \big[\mathbf{H}_k^i(\boldsymbol{\theta})\big]^\top + \widehat{\mathbf{R}}_k^i(\boldsymbol{\theta}) \ , \tag{3.52b}$$

$$\mathbf{K}_k^i(\boldsymbol{\theta}) = \bar{\mathbf{P}}_k^i(\boldsymbol{\theta})\, \big[\mathbf{H}_k^i(\boldsymbol{\theta})\big]^\top \big[\mathbf{S}_k^i(\boldsymbol{\theta})\big]^{-1} \ , \tag{3.52c}$$

$$\widetilde{\mathbf{x}}_k^i(\boldsymbol{\theta}) = \bar{\mathbf{x}}_k^i(\boldsymbol{\theta}) + \mathbf{K}_k^i(\boldsymbol{\theta})\, \mathbf{v}_k^i(\boldsymbol{\theta}) \ , \tag{3.52d}$$

$$\widetilde{\mathbf{P}}_k^i(\boldsymbol{\theta}) = \bar{\mathbf{P}}_k^i(\boldsymbol{\theta}) - \mathbf{K}_k^i(\boldsymbol{\theta})\, \mathbf{H}_k^i(\boldsymbol{\theta})\, \bar{\mathbf{P}}_k^i(\boldsymbol{\theta}) \ . \tag{3.52e}$$

3.4.2 Parameter estimation

The marginal posterior of parameters is calculated by marginalizing out the states $\mathbf{x}_{0:T}$ from the joint distribution of the parameters $\boldsymbol{\theta}$ and the states $\mathbf{x}_{0:T}$ such that

$$p\big(\boldsymbol{\theta} \mid \hat{\mathbf{y}}_{1:T}^i\big) = \int p\big(\boldsymbol{\theta}, \mathbf{x}_{0:T} \mid \hat{\mathbf{y}}_{1:T}^i\big)\, \mathrm{d}\mathbf{x}_{0:T} \,. \tag{3.53}$$

The MAP parameter estimate is

$$\hat{\boldsymbol{\theta}}_{\text{MAP}}^i = \arg\max_{\boldsymbol{\theta}} \big[p\big(\boldsymbol{\theta} \mid \hat{\mathbf{y}}_{1:T}^i\big)\big] \ . \tag{3.54}$$

The marginal posterior is computed with Bayes' theorem as

$$p\big(\boldsymbol{\theta} \mid \hat{\mathbf{y}}^i_{1:T}\big) = \frac{p\big(\hat{\mathbf{y}}^i_{1:T} \mid \boldsymbol{\theta}\big)p\big(\boldsymbol{\theta}\big)}{p\big(\hat{\mathbf{y}}^i_{1:T}\big)}\,, \tag{3.55}$$

where the terms are defined:

$$\begin{aligned} &\text{marginal posterior} && p\big(\boldsymbol{\theta} \mid \hat{\mathbf{y}}^i_{1:T}\big)\,, \\ &\text{marginal likelihood} && p\big(\hat{\mathbf{y}}^i_{1:T} \mid \boldsymbol{\theta}\big)\,, \\ &\text{prior} && p\big(\boldsymbol{\theta}\big)\,, \\ &\text{normalization term} && p\big(\hat{\mathbf{y}}^i_{1:T}\big)\,. \end{aligned} \tag{3.56}$$

The normalization term may be neglected to provide the relationship for the unnormalized marginal posterior

$$p\big(\boldsymbol{\theta} \mid \hat{\mathbf{y}}^i_{1:T}\big) \propto p\big(\hat{\mathbf{y}}^i_{1:T} \mid \boldsymbol{\theta}\big)p\big(\boldsymbol{\theta}\big)\,. \tag{3.57}$$

An advantage of the PDSSM in Eq. (3.30) is an analytic solution for calculating the marginal likelihood as a byproduct of the state estimator. The marginal likelihood can be factorized such that

$$p\big(\hat{\mathbf{y}}^i_{1:T} \mid \boldsymbol{\theta}\big) = p\big(\hat{\mathbf{y}}^i_1 \mid \boldsymbol{\theta}\big) \prod_{k=2}^{T} p\big(\hat{\mathbf{y}}^i_k \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big)\,. \tag{3.58}$$

The factorized marginal likelihood appeared in the denominator of the filtering distribution in Eq. (3.48) as a normalization constant but is utilized as part of the objective function for the parameter estimation. This term is calculated by marginalizing out the current state $\mathbf{x}_k$ from the joint distribution of the current state $\mathbf{x}_k$ and the pseudo measurement $\hat{\mathbf{y}}^i_k$ such that

$$p\big(\hat{\mathbf{y}}^i_k \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big) = \int p\big(\hat{\mathbf{y}}^i_k, \mathbf{x}_k \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big)\mathrm{d}\mathbf{x}_k\,. \tag{3.59}$$

Given the conditional independence of the pseudo measurements in Definition 3.3.2, the joint distribution simplifies to

$$p\big(\hat{\mathbf{y}}_k^i, \mathbf{x}_k \mid \hat{\mathbf{y}}_{1:k-1}^i, \boldsymbol{\theta}\big) = p\big(\hat{\mathbf{y}}_k^i \mid \mathbf{x}_k, \hat{\mathbf{y}}_{1:k-1}^i, \boldsymbol{\theta}\big) p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}_{1:k-1}^i, \boldsymbol{\theta}\big) \tag{3.60a}$$

$$= p\big(\hat{\mathbf{y}}_k^i \mid \mathbf{x}_k, \boldsymbol{\theta}\big) p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}_{1:k-1}^i, \boldsymbol{\theta}\big)\,. \tag{3.60b}$$

Consequently, the marginal likelihood is

$$p\big(\hat{\mathbf{y}}_k^i \mid \hat{\mathbf{y}}_{1:k-1}^i, \boldsymbol{\theta}\big) = \int p\big(\hat{\mathbf{y}}_k^i \mid \mathbf{x}_k, \boldsymbol{\theta}\big) p\big(\mathbf{x}_k \mid \hat{\mathbf{y}}_{1:k-1}^i, \boldsymbol{\theta}\big) \mathrm{d}\mathbf{x}_k\,. \tag{3.61}$$

The joint distribution of $\mathbf{x}_k$ and $\hat{\mathbf{y}}_k^i$ was found previously as part of the state estimation in Eq. (3.49). According to Lemma A.2.2 in the appendix, the marginal likelihood distribution is the marginal distribution of $\hat{\mathbf{y}}_k^i$ denoted as

$$p\big(\hat{\mathbf{y}}_k^i \mid \hat{\mathbf{y}}_{1:k-1}^i, \boldsymbol{\theta}\big) = \mathcal{N}\big(\hat{\mathbf{y}}_k^i \mid \mathbf{H}_k^i(\boldsymbol{\theta})\, \bar{\mathbf{x}}_k^i(\boldsymbol{\theta})\,, \mathbf{S}_k^i(\boldsymbol{\theta})\big)\,, \tag{3.62}$$

where the residual vector $\mathbf{v}_k^i$ and covariance matrix $\mathbf{S}_k^i$ were defined in Eqs. (3.52a) and (3.52b).

Energy function

The energy function enables the calculation of the MAP parameter estimate without actually calculating the marginal posterior. The MAP estimate is the argument that minimizes the energy function φ_T such that

$$\hat{\boldsymbol{\theta}}_{\mathrm{MAP}}^i = \arg\min_{\boldsymbol{\theta}} \left[\varphi_T^i(\boldsymbol{\theta})\right]\,, \tag{3.63}$$

where the energy function is the negative log of the un-normalized marginal posterior

$$\varphi_T^i(\boldsymbol{\theta}) = -\log\left[p\big(\hat{\mathbf{y}}_{1:T}^i \mid \boldsymbol{\theta}\big) p\big(\boldsymbol{\theta}\big)\right]\,. \tag{3.64}$$

The substitution is valid as the maximum of the marginal posterior in Eq. (3.55) and the minimum of the negative log of the un-normalized marginal posterior in Eq. (3.57) are both the MAP estimate $\hat{\boldsymbol{\theta}}^i_{\text{MAP}}$. Applying the factorization in Eq. (3.58) to the energy function yields

$$\varphi^i_T(\boldsymbol{\theta}) = -\log p\big(\boldsymbol{\theta}\big) - \log p\big(\hat{\mathbf{y}}^i_{1:T} \mid \boldsymbol{\theta}\big)\,, \tag{3.65a}$$

$$= -\log p\big(\boldsymbol{\theta}\big) - \log p\big(\hat{\mathbf{y}}^i_1 \mid \boldsymbol{\theta}\big) - \sum_{k=2}^{T} \log p\big(\hat{\mathbf{y}}^i_k \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big)\,. \tag{3.65b}$$

Substituting the probability density function of the multivariate Gaussian distribution,

$$p\big(\hat{\mathbf{y}}^i_k \mid \hat{\mathbf{y}}^i_{1:k-1}, \boldsymbol{\theta}\big) = \frac{1}{\sqrt{|2\pi \mathbf{S}^i_k(\boldsymbol{\theta})\,|}} \exp\!\left(-\frac{1}{2}\left[\mathbf{v}^i_k(\boldsymbol{\theta})\right]^\top \left[\mathbf{S}^i_k(\boldsymbol{\theta})\right]^{-1} \mathbf{v}^i_k(\boldsymbol{\theta})\right)\,, \tag{3.66}$$

yields the energy function as

$$\varphi^i_T(\boldsymbol{\theta}) = -\log p\big(\boldsymbol{\theta}\big) + \frac{1}{2}\sum_{k=1}^{T}\Big[\log|2\pi \mathbf{S}^i_k(\boldsymbol{\theta})\,| + \left[\mathbf{v}^i_k(\boldsymbol{\theta})\right]^\top \left[\mathbf{S}^i_k(\boldsymbol{\theta})\right]^{-1} \mathbf{v}^i_k(\boldsymbol{\theta})\Big]\,. \tag{3.67}$$

This residual and residual covariance can be calculated recursively with the KF as shown in Appendix B Algorithm 5 with the initial value

$$\varphi^i_0(\boldsymbol{\theta}) = -\log p\big(\boldsymbol{\theta}\big)\,. \tag{3.68}$$

Sensitivity equations

A gradient based optimization routine requires the termwise differentiation of the KF and energy function. The derivatives of the energy function are called the sensi-

tivity equations [2,43,44]. Using the identities for matrix derivatives in Appendix A.3, the derivative of the energy function is

$$\begin{aligned}\frac{\partial \varphi_T^i(\boldsymbol{\theta})}{\partial \theta_j} = -\frac{\partial \log p\big(\boldsymbol{\theta}\big)}{\partial \theta_j} + \frac{1}{2}\sum_{k=1}^{T}\Bigg[&\mathrm{Tr}\Big(\big[\mathbf{S}_k^i(\boldsymbol{\theta})\big]^{-1}\frac{\partial \mathbf{S}_k^i(\boldsymbol{\theta})}{\partial \theta_j}\Big)\\ &+ 2\big[\mathbf{v}_k^i(\boldsymbol{\theta})\big]^{\top}\big[\mathbf{S}_k^i(\boldsymbol{\theta})\big]^{-1}\frac{\partial \mathbf{v}_k^i(\boldsymbol{\theta})}{\partial \theta_j}\\ &- \big[\mathbf{v}_k^i(\boldsymbol{\theta})\big]^{\top}\big[\mathbf{S}_k^i(\boldsymbol{\theta})\big]^{-1}\frac{\partial \mathbf{S}_k^i(\boldsymbol{\theta})}{\partial \theta_j}\big[\mathbf{S}_k^i(\boldsymbol{\theta})\big]^{-1}\mathbf{v}_k^i(\boldsymbol{\theta})\Bigg]\,. \end{aligned} \tag{3.69}$$

The derivative of the residual and residual covariance can be calculated recursively with the derivative of the KF as shown in Appendix B Algorithm 6 with the initial value

$$\frac{\partial \varphi_0^i(\boldsymbol{\theta})}{\partial \theta_j} = -\frac{\partial \log p\big(\boldsymbol{\theta}\big)}{\partial \theta_j}\,. \tag{3.70}$$

3.4.3 Consensus estimation

For time instance k, the sensor network takes independent measurements $\mathbf{y}_k^{1:N}$. The pseudo measurement can be calculated with two CFs as discussed in Section 3.1.1. However, the measurement variance $\mathbf{R}_k^i(\boldsymbol{\theta})$ is unavailable when the calculation is performed in the pre-processing. Consequently, the unknown variance requires a guess as a replacement denoted as $\boldsymbol{\Sigma}_k^i$ such that the measurement model is assumed to be

$$\mathbf{y}_k^i \sim \mathcal{N}\big(\mathbf{H}_k^i(\boldsymbol{\theta})\,\mathbf{x}_k, \boldsymbol{\Sigma}_k^i\big)\,. \tag{3.71}$$

A couple of methods for selecting $\boldsymbol{\Sigma}_k^i$ are discussed in the next chapter in the context of civil aviation.

The MV estimate of the mean at time instance k is the weighted average with the inverse-variance weights:

$$\hat{\mathbf{y}}_{k,\mathrm{MV}} = \left(\sum_{i=1}^{N} \left[\boldsymbol{\Sigma}_k^i \right]^{-1} \right)^{-1} \left(\sum_{i=1}^{N} \left[\boldsymbol{\Sigma}_k^i \right]^{-1} \mathbf{y}_k^i \right), \tag{3.72a}$$

$$\mathrm{Var}(\hat{\mathbf{y}}_{k,\mathrm{MV}}) = \left(\sum_{i=1}^{N} \left[\boldsymbol{\Sigma}_k^i \right]^{-1} \right)^{-1}. \tag{3.72b}$$

However, a sensor platform may not have access to all the measurements and the variances and require a consensus algorithm. The weighted average consensus requires solving two average consensus problems. The consensus routines at sensor platform i are

$$\mathbf{a}_{k,\ell+1}^i = \mathbf{a}_{k,\ell}^i + \epsilon \sum_{j \in \mathcal{N}(i)} \left(\mathbf{a}_{k,\ell}^j - \mathbf{a}_{k,\ell}^i \right), \tag{3.73a}$$

$$\mathbf{b}_{k,\ell+1}^i = \mathbf{b}_{k,\ell}^i + \epsilon \sum_{j \in \mathcal{N}(i)} \left(\mathbf{b}_{k,\ell}^j - \mathbf{b}_{k,\ell}^i \right), \tag{3.73b}$$

where the initial values are

$$\mathbf{a}_{k,0}^i = \left[\boldsymbol{\Sigma}_k^i \right]^{-1} \mathbf{y}_k^i, \tag{3.74a}$$

$$\mathbf{b}_{k,0}^i = \left[\boldsymbol{\Sigma}_k^i \right]^{-1}. \tag{3.74b}$$

The MV estimate of the pseudo measurement is

$$\hat{\mathbf{y}}_{k,\mathrm{MV}} = \lim_{\ell \to \infty} \left[\mathbf{b}_{k,\ell}^i \right]^{-1} \mathbf{a}_{k,\ell}^i, \tag{3.75a}$$

$$\mathrm{Var}(\hat{\mathbf{y}}_{k,\mathrm{MV}}) = \lim_{\ell \to \infty} \left[N \mathbf{b}_{k,\ell}^i \right]^{-1}. \tag{3.75b}$$

After L iterations of the consensus routine, the pseudo measurement can be recovered at each sensor platform with

$$\hat{\mathbf{y}}_k^i = [\mathbf{b}_{k,L}^i]^{-1}\mathbf{a}_{k,L}^i\,, \tag{3.76}$$

$$\mathrm{Var}\big(\hat{\mathbf{y}}_k^i\big) = [N\mathbf{b}_{k,L}^i]^{-1}\,. \tag{3.77}$$

3.5 Adaptive Distributed Kalman Filter

The strategy for the Adaptive Distributed Kalman Filter (ADKF) is similar to the ACKF, but utilizes a DKF instead of a KF for the state estimation. Figure 3.8 depicts a flow diagram for the ADKF at one sensor platform. This process occurs at each sensor platform, with the intention that all sensor platforms in the network reach a consensus on state and parameter estimates.

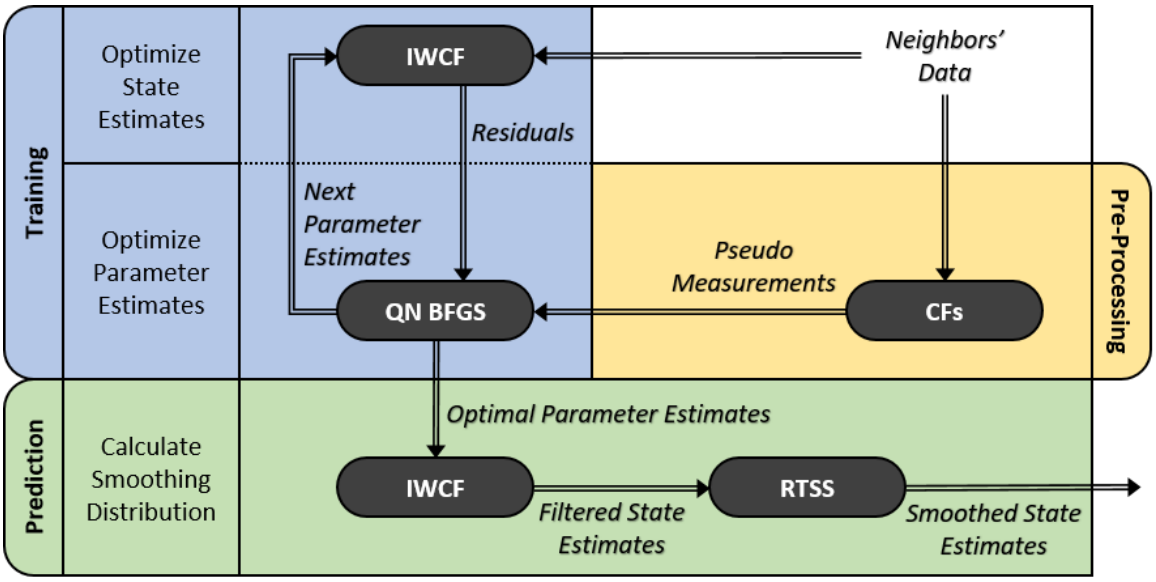


Figure 3.8. Flow Diagram for an Adaptive Distributed Kalman Filter at one sensor platform.

As in the ACKF, the pre-processing consists of two CFs calculating the pseudo measurements. However, the pseudo measurements are utilized in the parameter es-

timation, but not the state estimation. The DKF utilizes two CFs to maintain a consensus on the state estimates. After the pre-processing, the state and parameter estimation occurs by training a model and calculating the predicted values at target locations. During the training phase, the algorithm alternates between optimizing state and parameter estimates. The state estimation depends upon a DKF calculating the predictive and filtering distributions. Specifically, the flow diagram depicts the IWCF, which will be used in the following derivations and analysis. Determining the MAP parameter estimate depends upon maximizing the marginal posterior distribution of parameters. For gradient-based optimization with the quasi-Newton BFGS algorithm, the objective function is a byproduct of the DKF while the gradient can be determined via termwise differentiation [2, 43]. With the MAP parameter estimate, the prediction phase involves a DKF and a RTSS calculating the smoothing distribution. The following subsections will explain the state, parameter, and consensus estimation routines.

3.5.1 State estimation

The state estimator calculates the predictive and filtering distributions as the Gaussian distributions

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \bar{\mathbf{x}}_k^i(\boldsymbol{\theta}), \bar{\mathbf{P}}_k^i(\boldsymbol{\theta})\big), \tag{3.78a}$$

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k}^{1:N}, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \widetilde{\mathbf{x}}_k^i(\boldsymbol{\theta}), \widetilde{\mathbf{P}}_k^i(\boldsymbol{\theta})\big). \tag{3.78b}$$

The algorithm is recursive; the filtering distribution appears in the predictive distribution and vice versa. The recursion starts from the prior mean and covariance

$$\mathbf{x}_0 \sim \mathcal{N}\big(\widetilde{\mathbf{x}}_0^i(\boldsymbol{\theta}), \widetilde{\mathbf{P}}_0^i(\boldsymbol{\theta})\big). \tag{3.79}$$

In the predict step, the predictive distribution is calculated by marginalizing out the previous state $\mathbf{x}_{k-1}$ from the joint distribution of the current state $\mathbf{x}_k$ and the previous state $\mathbf{x}_{k-1}$ such that

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big) = \int p\big(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big) \mathrm{d}\mathbf{x}_{k-1} \,. \tag{3.80}$$

Given the Markov property of states in Definition 3.2.1, the predictive distribution can be rewritten as

$$\begin{aligned} p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big) &= \int p\big(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big) p\big(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big) \mathrm{d}\mathbf{x}_{k-1} && \text{(3.81a)} \\ &= \int p\big(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}\big) p\big(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big) \mathrm{d}\mathbf{x}_{k-1} \,. && \text{(3.81b)} \end{aligned}$$

The *a priori* statistics can then be calculated by evaluating the predictive distribution. According to Lemma A.2.1 in the appendix, the joint distribution of $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$ is

$$\begin{bmatrix} \mathbf{x}_{k-1} \\ \mathbf{x}_k \end{bmatrix} \sim \mathcal{N}\big(\boldsymbol{\mu}_{\text{temp\_3}}, \boldsymbol{\Sigma}_{\text{temp\_3}}\big) \,, \tag{3.82}$$

where

$$\boldsymbol{\mu}_{\text{temp\_3}} = \begin{bmatrix} \widetilde{\mathbf{x}}_{k-1}(\boldsymbol{\theta}) \\ \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{x}}_{k-1}(\boldsymbol{\theta}) \end{bmatrix} , \tag{3.83a}$$

$$\boldsymbol{\Sigma}_{\text{temp\_3}} = \begin{bmatrix} \widetilde{\mathbf{P}}_{k-1}(\boldsymbol{\theta}) & \widetilde{\mathbf{P}}_{k-1}(\boldsymbol{\theta})\, \mathbf{A}_{k-1}^{\top}(\boldsymbol{\theta}) \\ \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{P}}_{k-1}(\boldsymbol{\theta}) & \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{P}}_{k-1}(\boldsymbol{\theta})\, \mathbf{A}_{k-1}^{\top}(\boldsymbol{\theta}) + \mathbf{Q}_{k-1}(\boldsymbol{\theta}) \end{bmatrix} . \tag{3.83b}$$

According to Lemma A.2.2 in the appendix, the predictive distribution is the marginal distribution of $\mathbf{x}_k$ denoted as

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \bar{\mathbf{x}}_k^i(\boldsymbol{\theta}) \,, \bar{\mathbf{P}}_k^i(\boldsymbol{\theta})\big) \,, \tag{3.84}$$

where

$$\bar{\mathbf{x}}_k^i(\boldsymbol{\theta}) = \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{x}}_{k-1}^i(\boldsymbol{\theta}) \;, \tag{3.85a}$$

$$\bar{\mathbf{P}}_k^i(\boldsymbol{\theta}) = \mathbf{A}_{k-1}(\boldsymbol{\theta})\, \widetilde{\mathbf{P}}_{k-1}^i(\boldsymbol{\theta})\, \mathbf{A}_{k-1}^\top(\boldsymbol{\theta}) + \mathbf{Q}_{k-1}(\boldsymbol{\theta}) \;. \tag{3.85b}$$

The derivation for the information form is provided in Appendix C. After applying the transformation for the information form, the *a priori* statistics are

$$\mathbf{M}_k^i(\boldsymbol{\theta}) = \left[\mathbf{A}_{k-1}^\top(\boldsymbol{\theta})\right]^{-1} \widetilde{\mathbf{Z}}_{k-1}^i(\boldsymbol{\theta}) \left[\mathbf{A}_{k-1}(\boldsymbol{\theta})\right]^{-1} \;, \tag{3.86a}$$

$$\boldsymbol{\Sigma}_k^i(\boldsymbol{\theta}) = \mathbf{M}_k^i(\boldsymbol{\theta}) + \left[\mathbf{Q}_{k-1}(\boldsymbol{\theta})\right]^{-1} \;, \tag{3.86b}$$

$$\bar{\mathbf{Z}}_k^i(\boldsymbol{\theta}) = \mathbf{M}_k^i(\boldsymbol{\theta}) - \mathbf{M}_k^i(\boldsymbol{\theta}) \left[\boldsymbol{\Sigma}_k^i(\boldsymbol{\theta})\right]^{-1} \mathbf{M}_k^i(\boldsymbol{\theta}) \;, \tag{3.86c}$$

$$\bar{\mathbf{z}}_k^i(\boldsymbol{\theta}) = \bar{\mathbf{Z}}_k^i(\boldsymbol{\theta})\, \mathbf{A}_{k-1}(\boldsymbol{\theta}) \left[\widetilde{\mathbf{Z}}_{k-1}^i(\boldsymbol{\theta})\right]^{-1} \widetilde{\mathbf{z}}_{k-1}^i(\boldsymbol{\theta}) \;. \tag{3.86d}$$

A mixed form of the state estimate and information matrix is often more convenient.

In the update step, the filtering distribution is calculated with Bayes' theorem as

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k}^{1:N}, \boldsymbol{\theta}\big) = \frac{p\big(\mathbf{y}_k^{1:N} \mid \mathbf{x}_k, \boldsymbol{\theta}\big) p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big)}{p\big(\mathbf{y}_k^{1:N} \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big)} \;. \tag{3.87}$$

According to Lemma A.2.1 in the appendix, the joint distribution of $\mathbf{x}_k$ and $\boldsymbol{\mathcal{Y}}_k$ is

$$\begin{bmatrix} \mathbf{x}_k \\ \boldsymbol{\mathcal{Y}}_k \end{bmatrix} \sim \mathcal{N}\big(\boldsymbol{\mu}_{\text{temp\_4}}, \boldsymbol{\Sigma}_{\text{temp\_4}}\big) \;, \tag{3.88}$$

where

$$\boldsymbol{\mu}_{\text{temp\_4}} = \begin{bmatrix} \bar{\mathbf{x}}_k(\boldsymbol{\theta}) \\ \boldsymbol{\mathcal{H}}_k(\boldsymbol{\theta})\, \bar{\mathbf{x}}_k(\boldsymbol{\theta}) \end{bmatrix} \;, \tag{3.89a}$$

$$\boldsymbol{\Sigma}_{\text{temp\_4}} = \begin{bmatrix} \bar{\mathbf{P}}_k(\boldsymbol{\theta}) & \bar{\mathbf{P}}_k(\boldsymbol{\theta})\, \boldsymbol{\mathcal{H}}_k^\top(\boldsymbol{\theta}) \\ \boldsymbol{\mathcal{H}}_k(\boldsymbol{\theta})\, \bar{\mathbf{P}}_k(\boldsymbol{\theta}) & \boldsymbol{\mathcal{H}}_k(\boldsymbol{\theta})\, \bar{\mathbf{P}}_k(\boldsymbol{\theta})\, \boldsymbol{\mathcal{H}}_k^\top(\boldsymbol{\theta}) + \boldsymbol{\mathcal{R}}_k(\boldsymbol{\theta}) \end{bmatrix} \;. \tag{3.89b}$$

Here, the column vector $\boldsymbol{\mathcal{Y}}_k \in \mathbb{R}^{mN}$ is the collection of measurement vectors, the column block matrix $\boldsymbol{\mathcal{H}}_k \in \mathbb{R}^{mN \times n}$ is the collection of measurement matrices, and the block diagonal matrix $\boldsymbol{\mathcal{R}}_k \in \mathbb{R}^{mN \times mN}$ is the collection of measurement noise matrices such that

$$\boldsymbol{\mathcal{Y}}_k = \operatorname{col}\big(\mathbf{y}_k^1 , \mathbf{y}_k^2 , \cdots , \mathbf{y}_k^N \big) \ , \tag{3.90a}$$

$$\boldsymbol{\mathcal{H}}_k(\boldsymbol{\theta}) = \operatorname{col}\big(\mathbf{H}_k^1(\boldsymbol{\theta}) \ , \mathbf{H}_k^2(\boldsymbol{\theta}) \ , \cdots , \mathbf{H}_k^N(\boldsymbol{\theta}) \big) \ , \tag{3.90b}$$

$$\boldsymbol{\mathcal{R}}_k(\boldsymbol{\theta}) = \operatorname{Diag}\big(\mathbf{R}_k^1(\boldsymbol{\theta}) \ , \mathbf{R}_k^2(\boldsymbol{\theta}) \ , \cdots , \mathbf{R}_k^N(\boldsymbol{\theta}) \big) \ . \tag{3.90c}$$

According to Lemma A.2.2 in the appendix, the filtering distribution is the conditional distribution of $\mathbf{x}_k$ denoted as

$$p\big(\mathbf{x}_k \mid \mathbf{y}_{1:k}^{1:N}, \boldsymbol{\theta}\big) = \mathcal{N}\big(\mathbf{x}_k \mid \widetilde{\mathbf{x}}_k^i(\boldsymbol{\theta}) , \widetilde{\mathbf{P}}_k^i(\boldsymbol{\theta}) \big) \, , \tag{3.91}$$

where

$$\boldsymbol{\mathcal{V}}_k^i(\boldsymbol{\theta}) = \boldsymbol{\mathcal{Y}}_k - \boldsymbol{\mathcal{H}}_k(\boldsymbol{\theta}) \, \bar{\mathbf{x}}_k^i(\boldsymbol{\theta}) \, , \tag{3.92a}$$

$$\boldsymbol{\mathcal{S}}_k^i(\boldsymbol{\theta}) = \boldsymbol{\mathcal{H}}_k(\boldsymbol{\theta}) \, \bar{\mathbf{P}}_k^i(\boldsymbol{\theta}) \, \boldsymbol{\mathcal{H}}_k^\top(\boldsymbol{\theta}) + \boldsymbol{\mathcal{R}}_k(\boldsymbol{\theta}) \ , \tag{3.92b}$$

$$\boldsymbol{\mathcal{K}}_k^i(\boldsymbol{\theta}) = \bar{\mathbf{P}}_k^i(\boldsymbol{\theta}) \, \boldsymbol{\mathcal{H}}_k^\top(\boldsymbol{\theta}) \, \big[\boldsymbol{\mathcal{S}}_k^i(\boldsymbol{\theta})\big]^{-1} \ , \tag{3.92c}$$

$$\widetilde{\mathbf{x}}_k^i(\boldsymbol{\theta}) = \bar{\mathbf{x}}_k^i(\boldsymbol{\theta}) + \boldsymbol{\mathcal{K}}_k^i(\boldsymbol{\theta}) \, \boldsymbol{\mathcal{V}}_k^i(\boldsymbol{\theta}) \ , \tag{3.92d}$$

$$\widetilde{\mathbf{P}}_k^i(\boldsymbol{\theta}) = \bar{\mathbf{P}}_k^i(\boldsymbol{\theta}) - \boldsymbol{\mathcal{K}}_k^i(\boldsymbol{\theta}) \, \boldsymbol{\mathcal{H}}_k(\boldsymbol{\theta}) \, \bar{\mathbf{P}}_k^i(\boldsymbol{\theta}) \ . \tag{3.92e}$$

After applying the transformation in Eq. (2.10) for the information form, the *a posteriori* statistics are

$$\widetilde{\mathbf{x}}_k^i(\boldsymbol{\theta}) = \big[\bar{\mathbf{Z}}_k^i(\boldsymbol{\theta}) + \boldsymbol{\mathcal{I}}_k(\boldsymbol{\theta})\big]^{-1} \big[\bar{\mathbf{z}}_k^i(\boldsymbol{\theta}) + \boldsymbol{\imath}_k(\boldsymbol{\theta})\big] \ , \tag{3.93a}$$

$$\widetilde{\mathbf{P}}_k^i(\boldsymbol{\theta}) = \big[\bar{\mathbf{Z}}_k^i(\boldsymbol{\theta}) + \boldsymbol{\mathcal{I}}_k(\boldsymbol{\theta})\big]^{-1} \ , \tag{3.93b}$$

where

$$\boldsymbol{i}_k(\boldsymbol{\theta}) = \boldsymbol{\mathcal{H}}_k^{\top}(\boldsymbol{\theta}) \left[\boldsymbol{\mathcal{R}}_k(\boldsymbol{\theta})\right]^{-1} \boldsymbol{\mathcal{Y}}_k \,, \tag{3.94a}$$

$$\boldsymbol{\mathcal{I}}_k(\boldsymbol{\theta}) = \boldsymbol{\mathcal{H}}_k^{\top}(\boldsymbol{\theta}) \left[\boldsymbol{\mathcal{R}}_k(\boldsymbol{\theta})\right]^{-1} \boldsymbol{\mathcal{H}}_k(\boldsymbol{\theta}) \,. \tag{3.94b}$$

Given the structure of the matrices in Eq. (3.90), the *a posteriori* statistics can be simplified to the form

$$\widetilde{\mathbf{x}}_k^i(\boldsymbol{\theta}) = \left[\bar{\mathbf{Z}}_k^i(\boldsymbol{\theta}) + \sum_{j=1}^{N} \mathbf{I}_k^j(\boldsymbol{\theta})\right]^{-1} \left[\bar{\mathbf{z}}_k^i(\boldsymbol{\theta}) + \sum_{j=1}^{N} \mathbf{i}_k^j(\boldsymbol{\theta})\right], \tag{3.95a}$$

$$\widetilde{\mathbf{P}}_k^i(\boldsymbol{\theta}) = \left[\bar{\mathbf{Z}}_k^i(\boldsymbol{\theta}) + \sum_{j=1}^{N} \mathbf{I}_k^j(\boldsymbol{\theta})\right]^{-1}, \tag{3.95b}$$

where the local information vector and matrix are

$$\mathbf{i}_k^j(\boldsymbol{\theta}) = \left[\mathbf{H}_k^j(\boldsymbol{\theta})\right]^{\top} \left[\mathbf{R}_k^j(\boldsymbol{\theta})\right]^{-1} \mathbf{y}_k^j \,, \tag{3.96a}$$

$$\mathbf{I}_k^j(\boldsymbol{\theta}) = \left[\mathbf{H}_k^j(\boldsymbol{\theta})\right]^{\top} \left[\mathbf{R}_k^j(\boldsymbol{\theta})\right]^{-1} \mathbf{H}_k^j(\boldsymbol{\theta}) \,. \tag{3.96b}$$

If consensus was achieved for the *a posteriori* statistics at time instance k, then $\bar{\mathbf{z}}_k^1(\boldsymbol{\theta}) = \bar{\mathbf{z}}_k^2(\boldsymbol{\theta}) = \cdots = \bar{\mathbf{z}}_k^N(\boldsymbol{\theta})$ and $\bar{\mathbf{Z}}_k^1(\boldsymbol{\theta}) = \bar{\mathbf{Z}}_k^2(\boldsymbol{\theta}) = \cdots = \bar{\mathbf{Z}}_k^N(\boldsymbol{\theta})$. Consequently, the *a posteriori* statistics can be rewritten as

$$\widetilde{\mathbf{x}}_k^i(\boldsymbol{\theta}) = \left[\sum_{j=1}^{N} \frac{1}{N} \bar{\mathbf{Z}}_k^j(\boldsymbol{\theta}) + \mathbf{I}_k^j(\boldsymbol{\theta})\right]^{-1} \left[\sum_{j=1}^{N} \frac{1}{N} \bar{\mathbf{z}}_k^j(\boldsymbol{\theta}) + \mathbf{i}_k^j(\boldsymbol{\theta})\right], \tag{3.97a}$$

$$\widetilde{\mathbf{P}}_k^i(\boldsymbol{\theta}) = \left[\sum_{j=1}^{N} \frac{1}{N} \bar{\mathbf{Z}}_k^j(\boldsymbol{\theta}) + \mathbf{I}_k^j(\boldsymbol{\theta})\right]^{-1}. \tag{3.97b}$$

After applying the transformation for the information form, the *a posteriori* statistics are

$$\widetilde{\mathbf{z}}_k^i(\boldsymbol{\theta}) = \sum_{j=1}^{N} \frac{1}{N} \bar{\mathbf{z}}_k^j(\boldsymbol{\theta}) + \mathbf{i}_k^j(\boldsymbol{\theta}) \ , \tag{3.98a}$$

$$\widetilde{\mathbf{Z}}_k^i(\boldsymbol{\theta}) = \sum_{j=1}^{N} \frac{1}{N} \bar{\mathbf{Z}}_k^j(\boldsymbol{\theta}) + \mathbf{I}_k^j(\boldsymbol{\theta}) \ . \tag{3.98b}$$

However, a sensor platform may not have access to all the information and require a consensus algorithm. This update step is a MV estimate, which can be computed with two CFs.

An additional assumption for the substitution of the *a posteriori* statistics is that each sensor platform in the network maintains a consensus on the parameter $\boldsymbol{\theta}$ throughout the parameter optimization. In other words, each sensor platform in the network utilizes the same value for the design variable for each function call (e.g., the computation for the state estimator, objective function, and gradient. This implies that the optimization routine at every sensor platforms utilizes the same consensus gain to determine the next value of the design variable. A future analysis of this method should consider differences in the design variable at sensor platforms i and j such that $\boldsymbol{\theta}^i \neq \boldsymbol{\theta}^j$. Furthermore, the analysis should consider differences in the *a posteriori* statistics such that $\bar{\mathbf{z}}_k^i\big(\boldsymbol{\theta}^i\big) \neq \bar{\mathbf{z}}_k^j\big(\boldsymbol{\theta}^j\big)$ and $\bar{\mathbf{Z}}_k^i\big(\boldsymbol{\theta}^i\big) \neq \bar{\mathbf{Z}}_k^j\big(\boldsymbol{\theta}^j\big)$.

3.5.2 Parameter estimation

The marginal posterior of parameters is calculated by marginalizing out the states $\mathbf{x}_{0:T}$ from the joint distribution of the parameters $\boldsymbol{\theta}$ and the states $\mathbf{x}_{0:T}$ such that

$$p\big(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}^{1:N}\big) = \int p\big(\boldsymbol{\theta}, \mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}^{1:N}\big) \, \mathrm{d}\mathbf{x}_{0:T} \, . \tag{3.99}$$

The MAP parameter estimate is

$$\hat{\boldsymbol{\theta}}^{i}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \left[p\big(\boldsymbol{\theta} \mid \mathbf{y}^{1:N}_{1:T}\big) \right] . \tag{3.100}$$

The marginal posterior is computed with Bayes' theorem as

$$p\big(\boldsymbol{\theta} \mid \mathbf{y}^{1:N}_{1:T}\big) = \frac{p\big(\mathbf{y}^{1:N}_{1:T} \mid \boldsymbol{\theta}\big) p\big(\boldsymbol{\theta}\big)}{p\big(\mathbf{y}^{1:N}_{1:T}\big)} , \tag{3.101}$$

where the terms are defined:

marginal posterior	$p\big(\boldsymbol{\theta} \mid \mathbf{y}^{1:N}_{1:T}\big)$,
marginal likelihood	$p\big(\mathbf{y}^{1:N}_{1:T} \mid \boldsymbol{\theta}\big)$,
prior	$p\big(\boldsymbol{\theta}\big)$,
normalization term	$p\big(\mathbf{y}^{1:N}_{1:T}\big)$.

The normalization term may be neglected to provide the relationship for the unnormalized marginal posterior

$$p\big(\boldsymbol{\theta} \mid \mathbf{y}^{1:N}_{1:T}\big) \propto p\big(\mathbf{y}^{1:N}_{1:T} \mid \boldsymbol{\theta}\big) p\big(\boldsymbol{\theta}\big) . \tag{3.102}$$

An advantage of the PDSSM in Eq. (3.30) is an analytic solution for calculating the marginal likelihood with a state estimator. The marginal likelihood can be factorized such that

$$p\big(\mathbf{y}^{1:N}_{1:T} \mid \boldsymbol{\theta}\big) = p\big(\mathbf{y}^{1:N}_{1} \mid \boldsymbol{\theta}\big) \prod_{k=2}^{T} p\big(\mathbf{y}^{1:N}_{k} \mid \mathbf{y}^{1:N}_{1:k-1}, \boldsymbol{\theta}\big) . \tag{3.103}$$

The factorized marginal likelihood appeared in the denominator of the filtering distribution in Eq. (3.87) as a normalization constant but is utilized here as part of the objective function for the parameter estimation. This term can be further factorized given the independence of the sensor measurements such that

$$p\big(\mathbf{y}^{1:N}_{k} \mid \mathbf{y}^{1:N}_{1:k-1}, \boldsymbol{\theta}\big) = \prod_{i=1}^{N} p\big(\mathbf{y}^{i}_{k} \mid \mathbf{y}^{i}_{1:k-1}, \boldsymbol{\theta}\big) . \tag{3.104}$$

However, a sensor platform may not have access to all the measurements to compute the marginal likelihood of parameters without CFs. According to Lemma A.2.3 in the appendix, the product of two Gaussian distributions is an un-normalized Gaussian distribution. By extending this result for the N Gaussian distributions of the marginal likelihood, the product is

$$p\big(\mathbf{y}_k^{1:N} \mid \mathbf{y}_{1:k-1}^{1:N}, \boldsymbol{\theta}\big) \propto p\big(\hat{\mathbf{y}}_k^i \mid \hat{\mathbf{y}}_{1:k-1}^i, \boldsymbol{\theta}\big)\,. \tag{3.105}$$

This substitution of the objective function is valid as the pseudo measurement is a convex combination of the measurements [45]. The marginal likelihood for sensor platform i was found previously for the parameter estimation of the ACKF in Eq. (3.62). However, the *a priori* statistics (i.e., $\bar{\mathbf{x}}_k^i(\boldsymbol{\theta})$ and $\bar{\mathbf{P}}_k^i(\boldsymbol{\theta})$) are provided by the IWCF instead of the KF.

As before with the ACKF, the MAP estimate is the argument that minimizes the energy function φ_T. The energy function and sensitivity equations were found previously in Eqs. (3.67) and (3.69), respectively. However, the pseudo measurement noise covariance $\widehat{\mathbf{R}}_k^i(\boldsymbol{\theta})$ is unavailable as an output so the measurement noise covariance $\mathbf{R}_k^i(\boldsymbol{\theta})$ or the inverse variance weight $\boldsymbol{\Sigma}_k^i$ can be utilized in an approximation. Future work will consider a derivation of the energy function based on the factorized marginal likelihood in Eq. (3.103), which would require consensus filters to compute the residuals. The pseudocode for calculating the energy function and sensitivity equations with the IWCF and its derivative are provided in Appendix B with Algorithms 7 and 8.

3.5.3 Consensus estimation

The consensus estimation requires consensus algorithms in the pre-processing for the pseudo measurements and the state estimation for the *a posteriori* statistics. For the pre-processing, calculating the pseudo measurements can be accomplished in

the same manner for ACKF as in Section 3.4.3. For state estimation, the consensus routines at sensor platform i for calculating the *a posteriori* statistics are

$$\mathbf{w}^i_{k,\ell+1}(\boldsymbol{\theta}) = \mathbf{w}^i_{k,\ell}(\boldsymbol{\theta}) + \epsilon \sum_{j\in\mathcal{N}(i)} \left(\mathbf{w}^j_{k,\ell}(\boldsymbol{\theta}) - \mathbf{w}^i_{k,\ell}(\boldsymbol{\theta})\right) , \tag{3.106a}$$

$$\mathbf{W}^i_{k,\ell+1}(\boldsymbol{\theta}) = \mathbf{W}^i_{k,\ell}(\boldsymbol{\theta}) + \epsilon \sum_{j\in\mathcal{N}(i)} \left(\mathbf{W}^j_{k,\ell}(\boldsymbol{\theta}) - \mathbf{W}^i_{k,\ell}(\boldsymbol{\theta})\right) , \tag{3.106b}$$

where the initial values are

$$\mathbf{w}^i_{k,0}(\boldsymbol{\theta}) = \frac{1}{N}\bar{\mathbf{z}}^i_k(\boldsymbol{\theta}) + \mathbf{i}^i_k(\boldsymbol{\theta}) , \tag{3.107a}$$

$$\mathbf{W}^i_{k,0}(\boldsymbol{\theta}) = \frac{1}{N}\bar{\mathbf{Z}}^i_k(\boldsymbol{\theta}) + \mathbf{I}^i_k(\boldsymbol{\theta}) . \tag{3.107b}$$

The MV estimate of the state is

$$\widetilde{\mathbf{x}}_{k,\mathrm{MV}}(\boldsymbol{\theta}) = \lim_{\ell\to\infty} \left[\mathbf{W}^i_{k,\ell}(\boldsymbol{\theta})\right]^{-1} \mathbf{w}^i_{k,\ell}(\boldsymbol{\theta}) , \tag{3.108a}$$

$$\widetilde{\mathbf{P}}_{k,\mathrm{MV}}(\boldsymbol{\theta}) = \lim_{\ell\to\infty} \left[N\mathbf{W}^i_{k,\ell}(\boldsymbol{\theta})\right]^{-1} . \tag{3.108b}$$

After L iterations of the consensus routine, the *a posteriori* state estimate and information matrix can be recovered at each sensor platform with

$$\widetilde{\mathbf{x}}^i_k(\boldsymbol{\theta}) = \left[\mathbf{W}^i_{k,L}(\boldsymbol{\theta})\right]^{-1} \mathbf{w}^i_{k,L}(\boldsymbol{\theta}) , \tag{3.109a}$$

$$\widetilde{\mathbf{Z}}^i_k(\boldsymbol{\theta}) = N\mathbf{W}^i_{k,L}(\boldsymbol{\theta}) . \tag{3.109b}$$

Furthermore, this process needs to be repeated for calculating the derivatives $\partial\widetilde{\mathbf{x}}^i_k/\partial\theta_j$ and $\partial\widetilde{\mathbf{Z}}^i_k/\partial\theta_j$ for all p parameters.

3.6 Comparison of the ACKF and ADKF

The following content addresses the (a) computational complexity, (b) communication cost, (c) optimality and stability, and (d) simulation-based performance of the ACKF and ADKF.

3.6.1 Comparison of the computational complexity

Table 3.4 shows the computational complexity for sensor platform i, which can be analyzed in terms of consensus, state, and parameter estimation. The computational complexity is dominated by the training of the model. The difference in computational complexity between the ACKF and ADKF is practically negligible. The computational complexity of the consensus estimation is negligible. The state estimation runs for T measurements with n as the length of the state vector and m as the length of the measurement vector. Typically, $n > m$ for dynamical systems. At time instance k, the KF has a computational complexity of $\mathcal{O}(4n^3)$ while the IWCF has a computational complexity of $\mathcal{O}(25n^3/3)$. See the appendix for the matrix operations with a cubic computational cost. The primary advantage of the recursive solutions (i.e., KF and IWCF) is a linear computational complexity as a function of T as opposed to the cubic complexity of the batch solution (i.e., GPR). For the parameter estimation, the optimization of p parameters requires running the state estimator once and the derivative of the state estimator p times Consequently, the total runs is approximately $p + 1$ state estimators. The parameter estimation runs for D function counts — the number of calls to the objective function and the gradient.

Table 3.4. The computational complexity for sensor platform i.

Algorithm	**Computational Complexity**
ACKF	$\mathcal{O}(TD(p+1)(4n^3))$
ADKF	$\mathcal{O}\big(TD(p+1)(\frac{25}{3}n^3)\big)$

3.6.2 Comparison of the communication cost

Table 3.5 shows the communication cost for sensor platform i. ACKF and ADKF require the broadcast of the variables $\mathbf{a}_{k,\ell}^i$ and $\mathbf{b}_{k,\ell}^i$ for computing the pseudo measurement $\hat{\mathbf{y}}_k^i$ in Eq. (3.76). Since m is the length of measurement vector, the message size for this vector and matrix is $m^2 + m$. The consensus routine is called for L_1 iterations for each of the T measurements. Only the ADKF requires the broadcast of the variables $\mathbf{w}_{k,\ell}^i$ and $\mathbf{W}_{k,\ell}^i$ for consensus on the *a posteriori* state estimate $\widetilde{\mathbf{x}}_k^i$ and information matrix $\widetilde{\mathbf{Z}}_k^i$ in Eq. (3.109). The gradient-based optimization routine also requires the broadcast of the derivatives $\partial \mathbf{w}_{k,\ell}^i / \partial \theta_j$ and $\partial \mathbf{W}_{k,\ell}^i / \partial \theta_j$ for all p parameters. Since n is the length of the state vector, the message size for each pair is $n^2 + n$ for $p + 1$ pairs. The consensus routine is called for L_2 iterations for each of the T measurements. As this consensus routine is nested inside the training step, the consensus routine is also called in each of the D function calls. The term L_2TD describes this nesting problem in which some CFs are nested inside the state estimation, which is nested inside the parameter estimation. For a communication network with a fixed bandwidth, the ADKF can be orders of magnitude slower than the ACKF. Furthermore, the ADKF is likely to present a significant challenge to realize a communication network with a sufficient broadcast rate.

3.6.3 Comparison of the optimality and stability

The optimality and stability properties are summarized in Table 3.6 for the estimation routines. For weighted average consensus, two CFs using the Perron matrix provided the minimum variance (MV) estimate. The ACKF and ADKF utilize this technique for computing the pseudo measurements, but only the ADKF utilizes this technique in the state estimation. The speed of each CF is at least as fast as the 2nd largest eigenvalue of the Perron matrix [25, 26]. This routine requires the consensus gain ϵ. Selecting a good gain requires requires global information about the network: the number of sensors N. A node counting method for N as well as consensus meth-

Table 3.5. The communication cost for sensor platform i.

Algorithm	**Broadcast Variable**	**Matrix Dimensions**	**Communication Cost**
ACKF	$\mathbf{a}_{k,\ell}^i$	$(m \times 1)$	$L_1 T(m^2 + m)$
	$\mathbf{b}_{k,\ell}^i$	$(m \times m)$	
ADKF	$\mathbf{a}_{k,\ell}^i$	$(m \times 1)$	$L_1 T(m^2 + m) + L_2 TD(n^2 + n)(p + 1)$
	$\mathbf{b}_{k,\ell}^i$	$(m \times m)$	
	$\mathbf{w}_{k,\ell}^i$	$(n \times 1)$	
	$\mathbf{W}_{k,\ell}^i$	$(n \times n)$	
	$\partial \mathbf{w}_{k,\ell}^i / \partial \theta_j$	$(n \times 1)$	
	$\partial \mathbf{W}_{k,\ell}^i / \partial \theta_j$	$(n \times n)$	

Table 3.6. The optimality and stability of the estimation algorithms.

Estimation	**Optimality**	**Stability**
Consensus	Minimum Variance	Globally exponentially stable
State	Minimum Mean Squared Error	Globally asymptotically stable
Parameter	Maximum A Posteriori	Locally stable with superlinear convergence

ods with local information are presented by Garin in [25]. As the number of consensus iterations L is finite, the CF in practice is generally near-optimal.

The state estimation in ACKF and ADKF was derived to calculate the minimum mean squared error (MMSE) estimate, which is globally asymptotically stable. Even though the state estimation is optimal in the sensor of MMSE, the ACKF and ADKF assume different underlying models. For the ACKF, the state estimator was a KF conditioned on the pseudo measurements. At time instance k, the pseudo measurement is a MV estimate of the weighted average of the measurements. The inverse-variance weight $\mathbf{\Sigma}_k^i$ can be the same for all time instances for simplicity. For the ADKF, the state estimator was the IWCF which has CFs embedded within the algorithm. This

enables the *a posteriori* statistics of the IWCF to be conditioned on the network's measurements and state estimates. See the discussions by Kamal et al. in [1, 46] for additional content on the stability of the IWCF. As the number of consensus iterations L is finite, the IWCF in practice is generally near-optimal. Earlier work on DKF often utilized $L = 1$ such as the suboptimal algorithm Kalman-Consensus Filter (KCF) in [35] by Olfati-Saber. This approach represents an important case in which the state estimator at time instance k can still be globally asymptotically stable despite a lack of convergence in the early time instances. Consequently, the quality of the smoothed results would need to be investigated. For target tracking applications, predictive distributions are often more useful than the smoothed distributions.

The maximum a posteriori (MAP) parameter estimate was calculated with a gradient based optimization routine using the energy function and sensitivity equations. Gupta and Mehra discussed the computational difficulties of maximum likelihood estimation with linear dynamical systems in [43]. These complications can be extended to this MAP estimate and include multiple maxima, saddle-points, discontinuities, singular Hessian matrices, slow rates of convergence, and no convergence. If the Hessian matrix satisfies conditions of continuity and boundedness, then the quasi-Newton BFGS algorithm provides a superlinear rate of convergence [47]. The stability of the parameter estimation with quasi-Newton BFGS depends upon the problem (e.g., PSSM, signal-to-noise ratio, and initial values) [7, 43]. The choice of initial values impacts the probability of convergence as well as the probability of determining the globally optimal parameters. The ACKF calculates the MAP parameter estimate given the pseudo measurements from the pre-processing. For an optimal solution, the pseuedo-measurements would need to be updated in the training phase with the pseudo measurement noise covariance $\widehat{\mathbf{R}}_k^i(\boldsymbol{\theta})$. The ADKF calculates the MAP parameter estimate with a substitute for the noise covariance as an approximation. For an optimal solution, the energy function should be based on the factorized marginal likelihood in Eq. (3.103), which would require consensus filters to compute the residuals.

3.6.4 Comparison of the simulation-based performance

A simulation-based performance of ACKF and ADKF can highlight the difference in the state estimation. Consequently, the following simulations depict the state estimation results for one function call in the parameter estimation. In other words, the adaptive component is not necessary to clarify the differences between the KF and IWCF in the two distributed estimation algorithms.

An example with nominal conditions

This example utilized a fully connected network with three sensor platforms under nominal conditions for one function call in ACKF and ADKF. The true states were a random sample of the NCAM in Eq. (2.58) with the sampling rate $\Delta t_k = 1$ s and power spectral density $q_c = 0.01$ ft^2/s^5. The measurement model for sensor i tracking the target's position was a random sample with

$$\mathbf{H}_k^i = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \qquad \text{and} \qquad R_k^i = [\sigma_n^i]^2 \,, \tag{3.110}$$

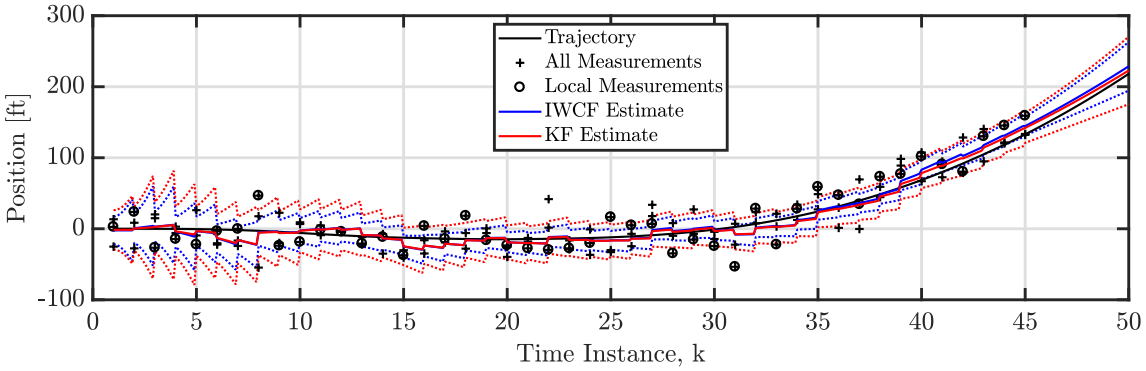
where $\sigma_n^i = 20$ ft for all sensors.

ACKF and ADKF were run for one function call of the parameter estimation in order to highlight the difference in the state estimators. The design choices for the two solutions are summarized in Table 3.7. These solutions assumed perfect knowledge of the constants in the dynamical model (i.e., Δt_k, q_c, and σ_n^i). For the pre-processing, ACKF utilized the guess for the inverse-variance weight $\Sigma_k^i = 20^2$ ft^2 for each sensor. Since all the weights are the same, the value is not particularly important as the weighted average is normalized. The state estimator for the ADKF was the IWCF and RTSS while the state estimator for the ACKF was the KF and RTSS. The CF had $L = 10$ iterations for achieving consensus at each time instance k with the consensus gain $\epsilon = 1/4$.

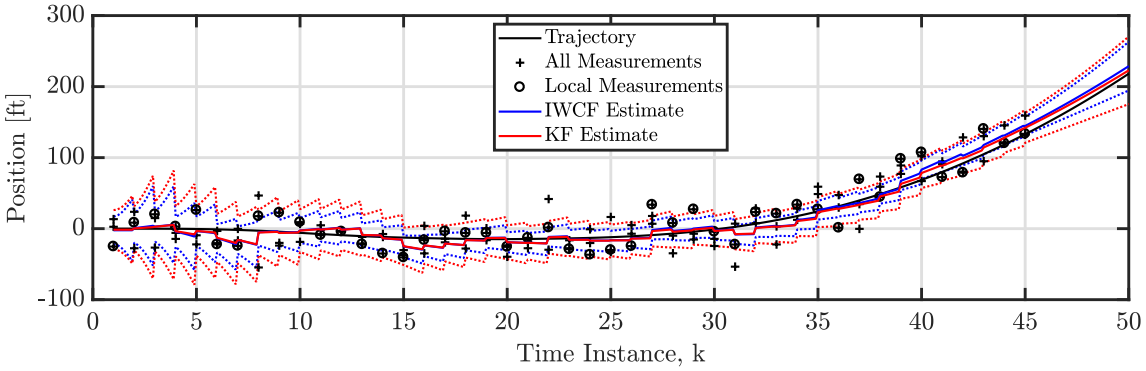
Table 3.7. Design choices for an example of the IWCF and KF.

(1) **State-Space Model** - NCAM
(2) **Constants** - $\Delta t_k = 1$ s, $q_c = 0.01$ ft^2/s^5, and $\sigma_n^i = 20$ ft (b) $\Sigma_k^i = 20^2$ ft^2
(3) **State Estimation** (a) IWCF and RTSS (b) KF and RTSS
(4) **Consensus Filter** - Perron matrix with $\epsilon = 1/4$ and $L = 10$

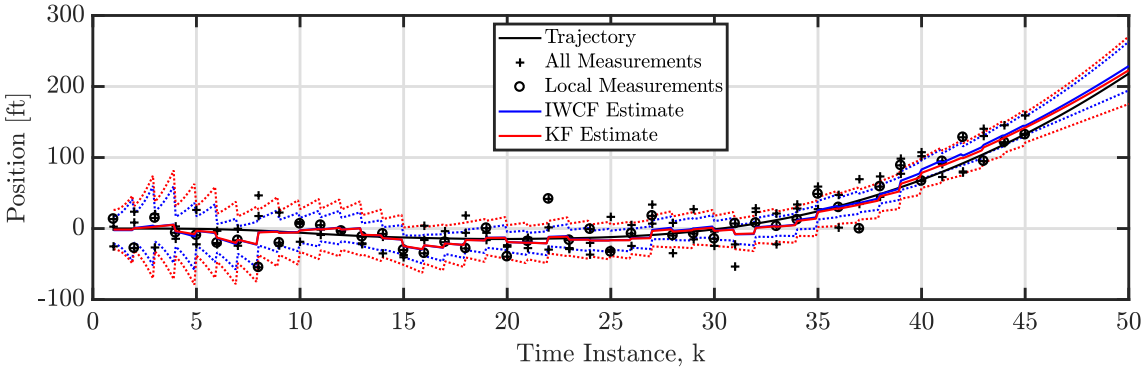
The filtered results are displayed in Fig. 3.9 while the smoothed results are displayed in Fig. 3.10. The black line is the target's true trajectory. The crosses are all the measurements of the sensor network while the circles are the sensor's local measurements. The blue line is the estimated mean for the IWCF while the blue dotted line is the 95% confidence region. The red line is the estimated mean for the KF while the red dotted line is the 95% confidence region. With the IWCF, each sensor platform calculates the same filtered estimate so consensus was achieved. With the KF, each sensor platform calculates the same filtered estimate so consensus was achieved. However, the solutions for the KF and IWCF are slightly different. These conclusions hold after the RTSS with the smoothed results too. This example demonstrates that the state estimators in ACKF an ADKF both provide good estimates with nominal conditions. Next, an example with off-nominal condition is presented, which is more useful to elicit differences in the weighting scheme of the state estimators.



(a) Filtered estimates for sensor platform 1.

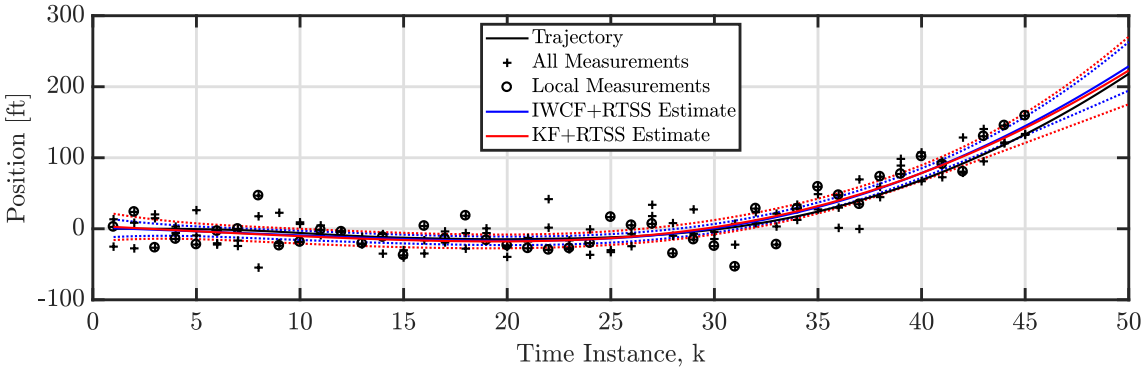


(b) Filtered estimates for sensor platform 2.

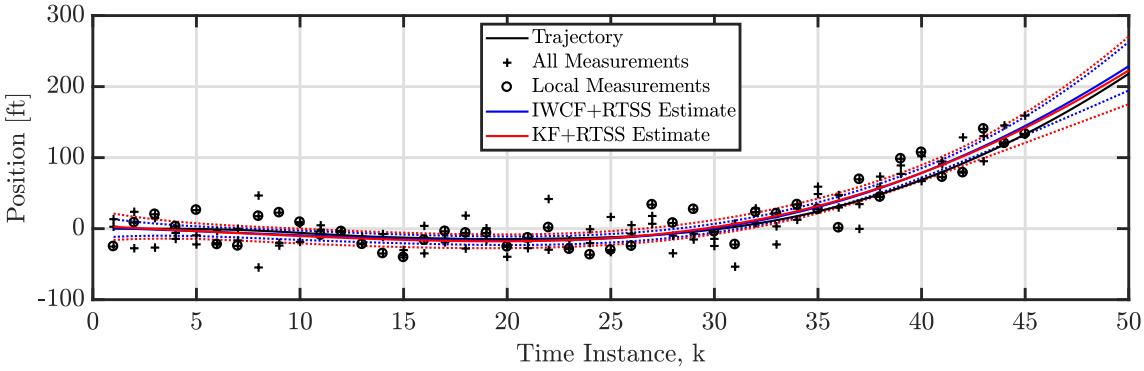


(c) Filtered estimates for sensor platform 3.

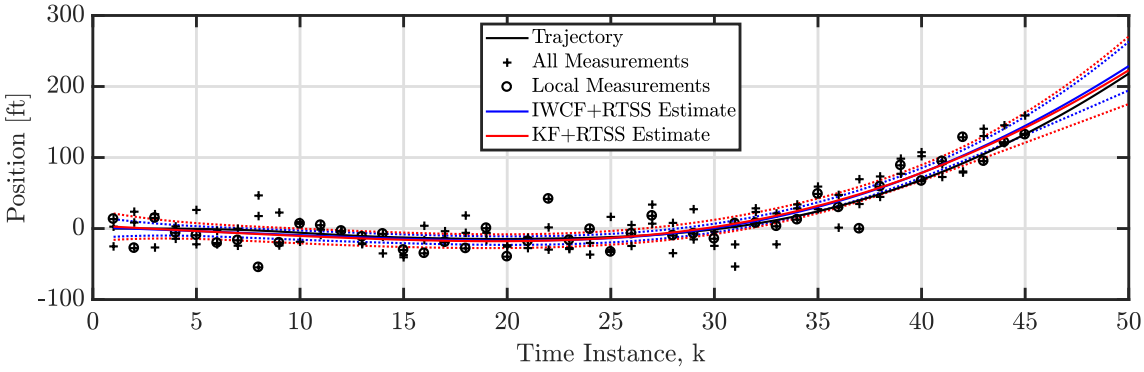
Figure 3.9. Filtered estimates for the Information Weighted Consensus Filter and Kalman Filter.



(a) Smoothed estimates for sensor platform 1.



(b) Smoothed estimates for sensor platform 2.



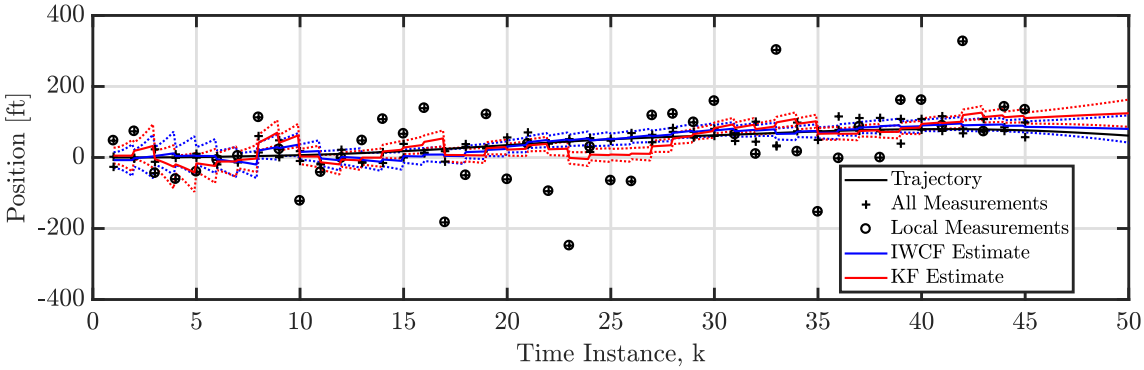
(c) Smoothed estimates for sensor platform 3.

Figure 3.10. Smoothed estimates for the Information Weighted Consensus Filter and Kalman Filter.

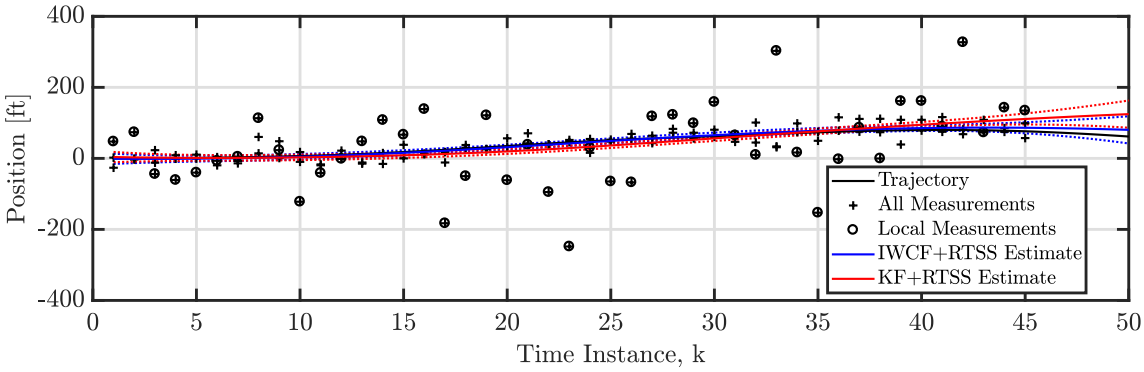
An example with an off-nominal condition

This example considers the same scenario with an off-nominal condition. The true states were another random sample of the NCAM with the sampling rate $\Delta t_k = 1$ s and power spectral density $q_c = 0.01$ ft^2/s^5. However, the network's position measurements were a random sample with $\sigma_n^i = 20$ ft for sensors 1 and 2 and $\sigma_n^i = 100$ ft for sensor 3. Furthermore, the ACKF utilized the guess $\Sigma_k^i = 20^2$ ft^2 for each sensor in the pre-processing not realizing that third sensor's performance was degraded.

The filtered and smoothed results are displayed in Fig. 3.11 for sensor platform 3. The black line is the target's true trajectory. The crosses are all the measurements of the sensor network while the circles are the sensor's local measurements. The blue line is the estimated mean for the IWCF while the blue dotted line is the 95% confidence region. The red line is the estimated mean for the KF while the red dotted line is the 95% confidence region. With the IWCF, each sensor platform calculates the same filtered estimate so consensus was achieved. With the KF, each sensor platform calculates the same filtered estimate so consensus was achieved. These results hold after the RTSS with the smoothed results too. However, the solutions for the KF and IWCF were actually very different, which is easier to see with the zoomed in image in Fig. 3.12. Unlike the KF in the ACKF, the IWCF in the ADKF does not require a guess of the inverse variance weights. For sensor 3, the guess $\Sigma_k^i = 20^2$ ft^2 can be described as 'overconfident' compared to the actual measurement variance $R_k^i(\boldsymbol{\theta}) = [\sigma_n^i]^2 = 100^2$ ft^2. The overconfident guess resulted in bad estimates with the KF and RTSS, which can be seen around $k = [15, 30]$ and $k = [40, 50]$. In other simulations, an under-confident guess resulted in good estimates with the KF and RTSS that were much more similar to the results from the IWCF and RTSS.



(a) Filtered estimates for sensor platform 3.



(b) Smoothed estimates for sensor platform 3.

Figure 3.11. Filtered and smoothed estimates for the Information Weighted Consensus Filter and Kalman Filter for sensor platform 3.

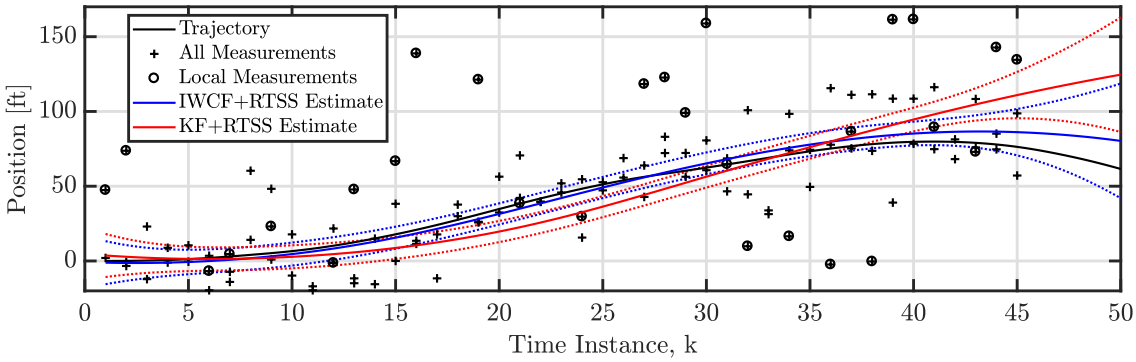


Figure 3.12. Zoom in on the smoothed estimates for sensor platform 3.

3.7 Summary and Contributions

The objective of this chapter was to address how a sensor network could maintain consensus while estimating the state of a dynamical system and the parameters defining a model of that system. To start, a literature review discussed the Consensus Filter (CF) and Distributed Kalman Filter (DKF). A special formulation with two CFs was presented as a desirable routine for maintaining consensus on a pseudo measurement, which works for sensor platforms with inactive sensors. The evolution of DKF algorithms was presented in which the Information Weighted Consensus Filter (IWCF) was highlighted as an optimal estimator. Then, the main problem of this dissertation was defined as the state and parameter estimation of a parameterized distributed state-space model (PDSSM). A point of difficulty resided in how to use the CFs to guarantee consensus with limitations in the communication network.

Two solutions were presented: the Adaptive Centralized Kalman Filter (ACKF) and the Adaptive Distributed Kalman Filter (ADKF). Theses solutions were designed to highlight the difference between utilizing a KF and DKF for the state estimation. As a result, the ACKF utilized CFs to calculate the pseudo measurements while the ADKF utilized CFs to calculate the pseudo measurements and *a posteriori* statistics. The solutions were then compared the (a) computational complexity, (b) communication cost, (c) optimality and stability, and (d) simulation-based performance. The major difference between the ACKF and the ADKF was in the communication cost, in which the ADKF is likely to present a significant challenge to realize a communication network with a sufficient message size and broadcast rate for target tracking applications. The performance of their state estimators provided very similar results with nominal conditions. Then, off-nominal conditions were considered to demonstrate that the KF in the ACKF's can provide poor results with an over confident guess of the true measurement uncertainty. Consequently, the ACKF is preferable to the ADKF because of the reduced communication cost.

4. SAFETY MOTIVATION IN CIVIL AVIATION

Implementing automated technologies for air traffic management in the National Airspace System (NAS) will rely heavily on the availability and quality of estimated aircraft state data. Availability refers to the degree to which a sensor platform can calculate state estimates. Given a probability for a faulty sensor, implementing a sensor network instead of a single sensor is expected to increase the availability by sharing relevant information. A comprehensive understanding of noisy and potentially faulty aircraft state information, its effect on safety within an airspace, and robust conflict mitigation strategies are still challenging problems. Maintaining the safety of an increasingly automated airspace requires technologies for detecting and resolving conflicts between aircraft. Consequently, this chapter analyzes the implementation of the ACKF algorithm in civil aviation in order to improve the safety of the airspace.

Some conflict detection algorithms can output the time until a loss of a minimum separation between the ownship and an intruder. This time value can be utilized as a method for quantifying safety. Section 4.1 includes a literature review on conflict detection methods with an emphasis on two models: Well Clear (WC) and Critical Pair Identification (CPI). In Section 4.2, an analysis demonstrates that WC and CPI, as a result of different underlying kinematic models, can be combined for improved conflict detection capabilities. Section 4.3 presents a safety assessment based on state estimates from the ACKF algorithm. Lastly, Section 4.4 summarizes the contributions of this chapter.

4.1 Literature Review

In 1956, the technical memo [48] by Morrel clarified the difficulty of the collision avoidance problem for aircraft with a detailed analysis of the fundamental problem and potential solutions. In the 1960's, Reich [49–51] determined separation standards for along track, across-track, and vertical dimensions based on the collision risk. Alexander [52] discussed the modeling of aircraft interactions with a gas model as the collision risk depends significantly on aircraft density. The role of air traffic control was discussed in the context of the aircraft density and whether the flight was ordered or unordered. Since then, the capabilities for conflict detection and resolution has grown more complicated than balancing the efficiency and safety of an airspace. A rigid airway structure with in-trail spacing is no longer guaranteed with modern flight given the potential for free flight and changing conditions of an airspace. This has led to a large variety of conflict detection and resolution methodologies for aiding pilots and traffic controllers in traffic management. In 1997, Kuchar provided a survey paper [53] with 33 models assessing five categories (e.g., airspace dimensions, state variables, propagation of the trajectory, uncertainty in state variables, and metrics for conflict detection).

One of the main differences in conflict detection models is whether the model is based on the probability of a collision or on the geometric relationship between aircraft. Lauderdale [54] compared geometric and probabilistic models for improving the robustness of conflict detection and resolution. Some recent probabilistic attempts were demonstrated by Blom and Bakker [55] for evaluating safety in a high traffic airspace and by Sahawneh et al. [56] for an Unmanned Aerial System (UAS) equipped with radar to estimate the collision risk with other UAS. To enhance the situational awareness of air traffic control, hybrid estimation was considered in [57,58] to infer an aircraft's flight plan and flight track. This methodology was based on the Interacting Multiple Models algorithm, which utilizes a bank of Kalman Filters.

For geometric models, Wikle et al. [59] compared four models and calculated the minimum distance required for conflict detection in order for a UAS to maintain separation. In [60], Maki developed a computationally efficient method for analysis of the collision risk of a near mid-air collision (NMAC) for unmanned aircraft based on historical tracks. An NMAC occurs when two aircraft are within 500 ft horizontally and 100 ft vertically of each other. A simple tool utilized by air traffic controllers for conflict detection is Loss of Separation (LoS), which occurs if the positions for a pair of aircraft are within 1,000 ft vertically and 5 NMi horizontally of each other. The Traffic Collision Avoidance System (TCAS) is a family of airborne devices developed to reduce risk of mid-air collisions for a manned aircraft by providing advisories to the pilot [61,62]. TCAS I, the first generation of the device family, provided a Traffic Advisory (TA) to the pilot when a potential threat was detected. TCAS II provides a TA in addition to a recommended conflict resolution maneuver called a Resolution Advisory (RA). The RA maneuver is an adjustment of the vertical speed. The TA provides conflict detection if a conflict develops, while an RA provides a conflict resolution if that conflict significantly worsens.

A recently introduced model called Well Clear (WC) originated from a NASA concept for sense-and-avoid in the context of integrating UAS into the NAS [63,64]. WC extends the TCAS functionality with improved conflict detection as well as conflict resolutions for adjustments in vertical speed, ground speed, and heading. Recently, NASA and Vigilant Aerospace Systems conducted flight tests with a Phantom 4 UAS to develop and certify the commercialized FlightHorizon technology, which is based on the sense-and-avoid concepts [65]. However, uncertainty exists on how to define WC for a smooth integration of UAS into the NAS. Lee and Park performed a preliminary study [66] to investigate the impact of four WC metrics on the safety and efficiency of a fast-time simulation of the NAS with UAS. Then, Cook et al. [67] evaluated three WC candidates according to eight metrics in four modeling and simulation environments. The result was an agreed upon and recommended definition for WC between subject matter experts at NASA, Massachusetts Institute of Technology

(MIT) Lincoln Labs, and Air Force Research Laboratory (AFRL). Many authors have extended the use of WC to small UAS. Weinert et al. [68] provided a definition of WC for small UAS based on airborne collision risk and operational suitability, and found that collision risk was not sensitive to the small UAS's speed, performance, or the manned aircraft's encounter model. Consequently, Weinert et al. suggested a simple hockey puck definition for WC involving simultaneous loss of horizontal and vertical separation. Duffield [69] provided a preliminary study for defining WC for small UAS equipped with an Automatic Dependent Surveillance-Broadcast (ADS-B) device for communication. Kim [70] presented a methodology to calculate conflict risk between small UAS with position uncertainty which noted a significant impact for flow rate, speed, intersection angle, and number of small UAS.

Critical Pair Identification (CPI) is a conflict detection model, which calculates the danger of an NMAC for a pair of aircraft should the ownship make an unexpected maneuver or "blunder" [71–74]. Examples of blunders are unexpected heading, altitude, and speed changes. Jacobs et al. [75] discussed the development of a framework to analyze and assess safety with LoS, TCAS, WC, and CPI. WC can provide the same or improved functionality as LoS and TCAS given similar threshold values. However, WC does not consider blunders in the prediction of an aircraft's future trajectory as is the case with CPI. Consequently, this chapter introduces conflict detection with the combined use of WC and CPI. Both models are defined in terms of a two aircraft scenario with an ownship and an intruder, in which the mathematics consider a local relative coordinate system of the ownship. Outputs include violations and a time response. A violation is a binary decision on whether a violation exists. A time response indicates the numbers of seconds until a collision or loss of a minimum separation. The remainder of the literature review presents the relevant parts of WC and CPI for generating time responses in the horizontal airspace.

4.1.1 Well Clear

The following description of WC summarizes the relevant content for conflict detection by Upchurch et al. in [64]. The problem setup for two aircraft is depicted in Fig. 4.1. The horizontal position and velocity are defined in a two-dimensional (2-D) horizontal airspace. The horizontal relative position is $\mathbf{s} = \mathbf{s}_o - \mathbf{s}_i$ where $\mathbf{s}_o$ and $\mathbf{s}_i$ are the horizontal positions of the ownship and the intruder, respectively. Similarly, the horizontal relative velocity is defined as $\mathbf{v} = \mathbf{v}_o - \mathbf{v}_i$ where $\mathbf{v}_o$ and $\mathbf{v}_i$ are the horizontal velocities of the ownship and the intruder, respectively.

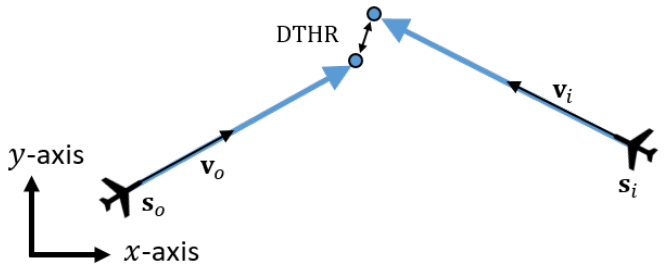


Figure 4.1. The horizontal kinematic model for Well Clear.

WC explored a family of four horizontal time models to extend TCAS II functionality. Considering the work by Jacobs [75] and Upchurch [64], t_{ep} is sufficient to replace the other three models. The time to the entry point (EP) t_{ep} is defined as the time for a horizontal loss of separation for the distance threshold `DTHR`:

$$t_{\text{ep}}(\mathbf{s}, \mathbf{v}) = \begin{cases} \Theta(\mathbf{s}, \mathbf{v}, \texttt{DTHR}, -1) & \text{if } \mathbf{s} \cdot \mathbf{v} < 0 \text{ and } \Delta(\mathbf{s}, \mathbf{v}, \texttt{DTHR}) \geq 0\,, \\ \texttt{NaN} & \text{otherwise}\,, \end{cases} \tag{4.1}$$

where

$$\Theta(\mathbf{s}, \mathbf{v}, D, \epsilon) = \frac{-\mathbf{s} \cdot \mathbf{v} + \epsilon\sqrt{\Delta(\mathbf{s}, \mathbf{v}, D)}}{\mathbf{v}^2}\,, \tag{4.2a}$$

$$\Delta(\mathbf{s}, \mathbf{v}, D) = D^2\mathbf{v}^2 - \left(\mathbf{s} \cdot \mathbf{v}^{\perp}\right)^2\,. \tag{4.2b}$$

MATLAB supports the command '`NaN`' (not-a-number), which is useful for the representation of undefined results. Note, we changed this condition from '-1' in [64] to avoid confusion with negative values of the time variable. The condition $\mathbf{s} \cdot \mathbf{v} < 0$ represents the horizontal convergence of the aircraft, which implies $\mathbf{v} \neq \mathbf{0}$ where the zero vector is $\mathbf{0} = (0, 0)$. For any distance D and the condition $\Delta(\mathbf{s}, \mathbf{v}, D) \geq 0$, the equation for $\Theta(\cdot)$ calculates the time to lose separation when $\epsilon = -1$ and the time to regain separation when $\epsilon = 1$. The condition $\Delta(\cdot) \geq 0$ prevents solutions with imaginary components. For the vector $\mathbf{v} = (v_x, v_y)$, the right-perpendicular vector is $\mathbf{v}^{\perp} = (v_y, -v_x)$. See Upchurch et al. [64] for a more comprehensive discussion on WC models in the context of creating boundary models for UAS.

4.1.2 Critical Pair Identification

The CPI concept considers blunders to produce a metric such that some agent in the system (i.e., human, automatic, or autonomous) may detect and resolve potential NMAC events within a certain time frame and likelihood of occurring. Surakitbanharn [74] included analysis of CPI in air traffic control as a horizontal time variable that determined the worst case heading blunder. Blunders in climb rate and ground speed as well as probabilistic implementations are still under investigation. This chapter considers the simple implementation of the heading blunder, which is depicted in Fig. 4.2. The horizontal time variable t_{nmac} is the time until a future heading blunder by the ownship can cause an NMAC.

The ownship's trajectory starts with a blunder consisting of a turn at the initial position $\mathbf{s}_o(0)$ with heading θ_o, blunder rate $\dot{\psi}$, and speed $\mathbf{v}_o(0)$. At time t_{turn}, the ownship exits the turn at blunder angle ψ and position $\mathbf{s}_o(\mathrm{t}_{\text{turn}})$. The time of the turn is defined as

$$t_{\text{turn}} = \text{sgn}(\psi) \, \frac{\psi}{\dot{\psi}} \, , \tag{4.3}$$

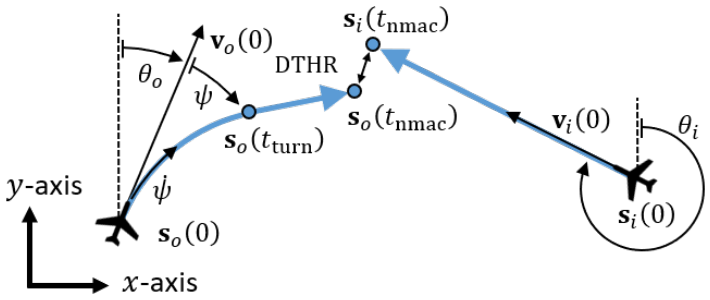


Figure 4.2. The horizontal kinematic model for Critical Pair Identification.

where $\dot{\psi} > 0$ for clockwise and counterclockwise turns. For the time $0 \leq t \leq t_{\text{turn}}$, the horizontal position $\mathbf{s}_o(t)$ and velocity $\mathbf{v}_o(t)$ of the ownship is defined as

$$\mathbf{s}_o(t) = \mathbf{s}_o(0) + \text{sgn}(\psi)\,\frac{\|\mathbf{v}_o(0)\|}{\dot{\psi}}\begin{bmatrix}\cos(\theta_o) - \cos\Big(\theta_o + \text{sgn}(\psi)\,\dot{\psi}t\Big) \\ \sin\Big(\theta_o + \text{sgn}(\psi)\,\dot{\psi}t\Big) - \sin(\theta_o)\end{bmatrix}^{\top}, \tag{4.4a}$$

$$\mathbf{v}_o(t) = \|\mathbf{v}_o(0)\|\begin{bmatrix}\sin\Big(\theta_o + \text{sgn}(\psi)\,\dot{\psi}t\Big) \\ \cos\Big(\theta_o + \text{sgn}(\psi)\,\dot{\psi}t\Big)\end{bmatrix}^{\top}, \tag{4.4b}$$

assuming the ownship travels at a constant ground speed. The horizontal position and velocity of the ownship for the time $t > t_{\text{turn}}$ is defined as

$$\mathbf{s}_o(t) = \mathbf{s}_o(t_{\text{turn}}) + (t - t_{\text{turn}})\,\mathbf{v}_o(t)\ , \tag{4.5a}$$

$$\mathbf{v}_o(t) = \mathbf{v}_o(t_{\text{turn}})\ , \tag{4.5b}$$

assuming the ownship travels at a constant ground speed. If the an NMAC exists at time t_{nmac}, then the ownship experiences that NMAC at position $\mathbf{s}_o(t_{\text{nmac}})$.

An intruder is predicted to travel on a straight path starting at position $\mathbf{s}_i(0)$ with heading θ_i and velocity $\mathbf{v}_i(0)$. At time t_{nmac}, the intruder experiences an NMAC at

position $\mathbf{s}_i(t_{\text{nmac}})$. For time t, the horizontal position $\mathbf{s}_i(t)$ and velocity $\mathbf{v}_i(t)$ of the intruder is

$$\mathbf{s}_i(t) = \mathbf{s}_i(0) + t\mathbf{v}_i(0) \ , \tag{4.6a}$$

$$\mathbf{v}_i(t) = \mathbf{v}_i(0) \ . \tag{4.6b}$$

Since CPI is actually an 'entry point' model that utilizes the NMAC condition, t_{nmac} can be defined similarly to t_{ep}. Given ψ and $\dot{\psi}$, the time until an NMAC after the ownship's blunder (i.e., $t > t_{\text{turn}}$) is defined as

$$t_{\text{nmac}}\big(\mathbf{s}(t_{\text{turn}}) \, , \mathbf{v}(t_{\text{turn}}) \big) = t_{\text{turn}} + t_{\text{ep}}\big(\mathbf{s}(t_{\text{turn}}) \, , \mathbf{v}(t_{\text{turn}}) \big) \, . \tag{4.7}$$

At time t, the relative position is $\mathbf{s}(t) = \mathbf{s}_o(t) - \mathbf{s}_i(t)$ and the relative velocity is $\mathbf{v}(t) = \mathbf{v}_o(t) - \mathbf{v}_i(t)$. This computation requires Eqs. (4.1), (4.3), (4.4), and (4.6). However, the future projection of the ownship and intruder for $t > t_{\text{turn}}$ with Eq. (4.5) is not necessary for computing t_{nmac}, but is included for completeness of defining the ownship's kinematics. In relation to previous publications on the CPI concept, this definition of t_{nmac} uses the compact notation defining WC by Upchurch et al. in [64].

Previous publications [71–75] with CPI defined the horizontal time variable t_{nmac} as an optimization routine in terms of blunder angles $\psi = [-90, +90]$ deg and blunder rates $\dot{\psi} = [0, +4.5]$ deg/s. For initial configurations in this research, t_{nmac} was calculated with a grid search using increments for blunder angles and blunder rates of $\Delta\psi = 0.1$ deg and $\Delta\dot{\psi} = 0.1$ deg/s, respectively. These increments were determined via simulations as sufficient for capturing t_{nmac} within a time frame of 100 s. Clearly, the search methodology could be optimized. Theunissen et al. [76] demonstrated the impact of modeling a constant turn rate versus an instantaneous turn. The main takeaway for the purposes of this chapter is that an aircraft with a larger blunder rate will arrive at a position in less time than an aircraft with a smaller blunder rate. Consequently, the grid search for t_{nmac} only needs to consider the blunder rate $\dot{\psi} = 4.5$ deg/s for the blunder angles $\psi = [-90, +90]$ deg. Further improvements for

the search methodology were not investigated as this research focuses on the comparison of conflict detection capabilities and not on design optimization.

4.2 Analysis with a Parameter Sweep of a Pairwise Conflict

Early research in conflict detection and resolution aimed to prevent conflicts by providing pilots and air traffic controllers useful information such as separation standards. The increasing aircraft density from civil aviation and the realization of new aerial concepts like Urban Air Mobility (UAM) and UAS Traffic Management (UTM) only exacerbates the problem with traffic management. Consequently, a Parameter Sweep of a Pairwise Conflict or PSPC analysis of conflict detection and resolution algorithms can enable an intuitive visual exploration of a large design space and verification of the defining characteristics. Interesting components of the output may include general trends, asymptotes, undefined regions, and discontinuities in the domain space. Furthermore, the method can enable a comparison of algorithms whose relationship is unclear.

To provide a deeper understanding of the conflict detection capabilities of WC and CPI, we are interested in applying the PSPC analysis by creating a response plot over a variety of initial aircraft states in the horizontal airspace. As depicted in Fig. 4.3, the pairwise conflict can be defined as a function of three variables: the relative approach angle θ, distance offset d, and radius r. This methodology originated from our earlier work [77] that assessed safety for aircraft with self-separation devices under the presence of faults, but the work did not analyze the fundamentals of the PSPC analysis. For analyzing potential conflicts in a horizontal airspace, we assume the aircraft maintain co-altitude, the same constant ground speed, and zero climb-rate. For $d > 0$, AC1 is always 'leading' AC2 to the intersection of their trajectories, which is denoted with a red marker in Fig. 4.3. Note, the notation of AC1 and AC2 does not determine which aircraft is the ownship or intruder. For WC and CPI, the conflict

detection models utilize a pair of aircraft in which the notation ACx/y means ACx is the ownship and ACy is the intruder.

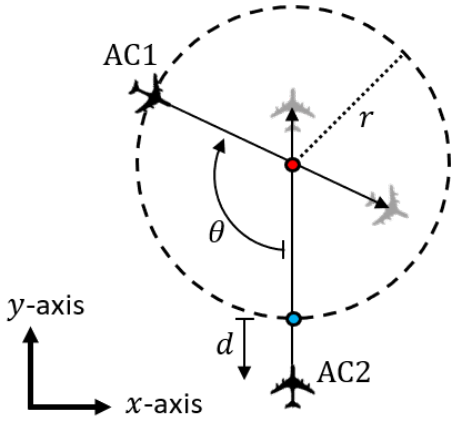


Figure 4.3. A pairwise conflict defined in terms of the relative approach angle θ, distance offset d, and radius r.

Upchurch et al. [64] presented a response plot of four different time variables for one pairwise conflict. The 2-D plot depicted the time responses plotted against the simulation time, which enabled a visualization of how the conflict evolves. Temporally evolving scenarios explored by Upchurch et al. only included some instantiations of initial state data. This type of analysis is suitable for replicating actual traffic scenarios in an airspace, but insufficient for a comprehensive view of the domain space of the conflict detection algorithm. For a PSPC analysis, the design space is translated from the state data to the variables r, d, and θ (see Fig. 4.3). The PSPC analysis allows for a more effective visualization by depicting the responses of conflicts with a large variety of initial aircraft states. This approach provides a 3-D plot if one of the three independent variables is held constant. For example, the metric's time or violation response can be plotted against relative approach angles $\theta = [0, 180]$ deg and distance offsets $d = [0, 10]$ NMi for the radius $r = 4$ NMi. As such, the PSPC analysis focuses on the properties of the conflict detection algorithm itself, rather than conclusions on several randomly initialized simulations. Our research indicates that

comprehensive conclusions can be reached from the PSPC analysis, when compared to exhaustive, randomly initialized simulation based studies. Additionally, the PSPC analysis combined with simulation-based approaches that include dynamics for the aircraft and environment can further enhance conclusions that can be made about conflict detection and resolution algorithms.

Here, we aim to create a graphical comparison by combining WC and CPI for a 'merged' time response, which is the minimum of t_{ep} and t_{nmac}. Figure 4.4 depicts a contour plot of the merged horizontal time response in seconds for AC1/2. Each aircraft had a ground speed of 350 knots. The time response was a function of the relative approach angle $\theta = [0, 180]$ deg, the offset $d = [0, 10]$ NMi, and the radius r fixed at 4 NMi. The values of the relative approach angle θ and offset d were selected to create a realistic traffic scenario. The responses for t_{ep} and t_{nmac} contain line symmetry at $\theta = 180$ deg so displaying the full domain of 0 to 360 deg is unnecessary. The WC code was downloaded from NASA's github repository[1] while CPI was programmed as a wrapper function around that code. The distance threshold for WC was `DTHR` = 1.3 NMi, which was based on the TCAS TA threshold values corresponding to an ownship altitude of 20,000 to 42,000 ft [62]. The distance threshold for CPI was `DTHR` = 500 ft, which was based on the NMAC definition.

A smaller value of t_{ep} or t_{nmac} for a given pair of aircraft means less time to detect and resolve the conflict and therefore represents a "riskier" situation. The merged response is significantly dependent on both time variables. The response values are quite different for WC and CPI as a result of the different underlying kinematic models. WC contains a larger distance threshold (`DTHR` = 1.3 NMi) and provides better detection for smaller d values. CPI contains a small distance threshold (`DTHR` = 500 ft), but includes the blunder kinematics. As a result, CPI provides better conflict detection around $30 < \theta < 120$ deg. If CPI utilized a larger distance threshold such as that in WC, then t_{nmac} would be too conservative for efficient use of the airspace. Another inefficient use of the airspace is the current separation standards. As described in

[1]Available at `https://github.com/nasa/WellClear`.

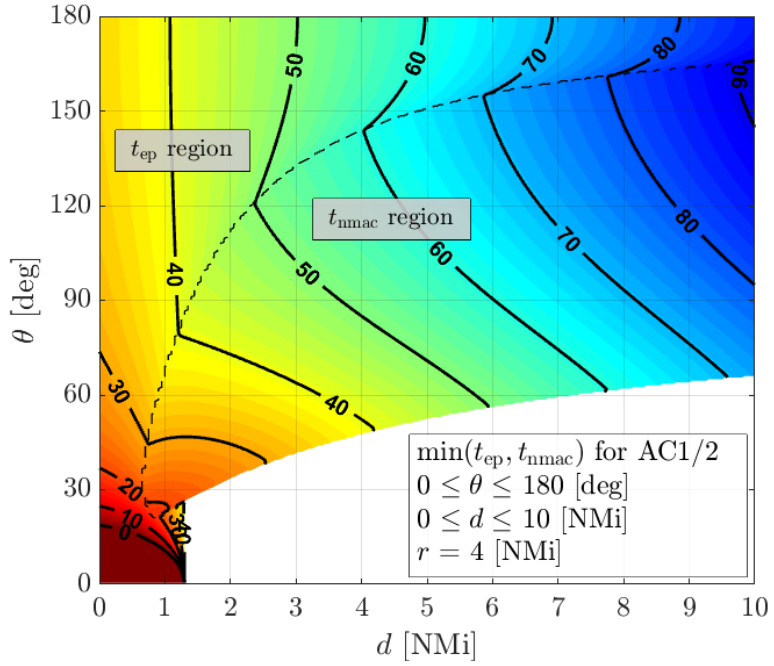


Figure 4.4. Contour plot of the merged time response based on WC's t_{ep} and CPI's t_{nmac} for AC1/2.

Section 8.7.3 of ICAO Doc 4444 [78], the minimum horizontal separation based on air traffic service (ATS) surveillance systems is 5 NMi. The separation minimum can be altered depending on the scenario, such as the availability and quality of estimated aircraft state data, wake turbulence, and area of operation (e.g. en-route, arrival, departure, and approach). For example, the minimum horizontal separation can be reduced to 3 NMi by an ATS authority when surveillance systems' capabilities permit. WC and CPI provide significantly different insight into safety-critical aircraft encounters than these simple geometric separation standards. Consequently, WC and CPI can improve the fidelity for separation minima with their kinematic models of the aircraft trajectories.

The merged time response combines WC's functionality for separation assurance with CPI's functionality for detecting NMAC events due to a blunder. The merged time response can benefit air traffic controllers as a decision support tool. A time

value enables the controller to make decisions with an understanding of how many critical pairs exist and the severity of those conflicts. The time responses can be converted to a violation model [64,75], which is appropriate for use in a dedicated on-board system in an UAS. Utilizing these merged responses for designing or assessing conflict resolution technologies is still an open problem.

4.3 Improving Safety in Civil Aviation

For applications in civil aviation within the near-future, distributed and adaptive target tracking can improve the availability and quality of estimated aircraft state data, and thus the safety and efficiency of an airspace. Here, the availability of state data refers to the degree to which aircraft and air traffic control stations can calculate state estimates of each aircraft for traffic management. Before presenting an example with the ACKF, the following subsections discuss how to quantify safety, correct for an airspace with noisy position measurements, and select an inverse-variance weight.

4.3.1 Correction for the estimated position uncertainty

The time response in Fig. 4.4 assumed AC1 had perfect knowledge of the intruder and itself. A safety assessment needs to account for uncertainty in all state data, but this example will focus on just the uncertainty in position. Automatic Dependent Surveillance - Broadcast (ADS-B) is a surveillance technology that periodically broadcasts the state information of the equipped aircraft. The broadcast includes a navigation accuracy category for position ($\mathrm{NAC_P}$) value in Table 4.1, which corresponds to the estimated position uncertainty (EPU) with a 95% accuracy bound.

Figure 4.5 depicts a correction for the EPU of the ownship and intruder denoted as EPU_o and EPU_i, respectively. The 'worst-case' position from a safety standpoint is obtained by shifting the AC's position to the boundary of the circle along the shortest distance vector to the intruder. Since the safety assessment utilizes the

Table 4.1. The navigation accuracy category for position and the corresponding estimated position uncertainty values.

$\mathbf{NAC_P}$	**EPU**
0	$\geq$ 10 NMi
1	$<$ 10 NMi
2	$<$ 4 NMi
3	$<$ 2 NMi
4	$<$ 1 NMi
5	$<$ 0.5 NMi
6	$<$ 0.3 NMi
7	$<$ 0.1 NMi
8	$<$ 0.05 NMi
9	$<$ 30 m
10	$<$ 10 m
11	$<$ 3 m

relative position, adjusting only the intruder's position by the EPU of both aircraft is a valid method as defined by

$$\mathbf{s}_{i,adj} = \mathbf{s}_i + \frac{\mathbf{s}_o - \mathbf{s}_i}{\|\mathbf{s}_o - \mathbf{s}_i\|} \left(EPU_o + EPU_i\right) , \tag{4.8}$$

where $\mathbf{s}_o$ is the ownship's position, $\mathbf{s}_i$ is the intruder's position, and $\mathbf{s}_{i,adj}$ is the intruder's adjusted position. Thus, the safety assessment was calculated with $\mathbf{s}_o$ and $\mathbf{s}_{i,adj}$.

A smaller time response indicates a more severe scenario. For the initial positions $\mathbf{s}_o$ and $\mathbf{s}_i$, the scenario that accounts for the EPU will have a smaller time response value than the scenario with the true position. This logic can be extended to the intuitive conclusion that a larger uncertainty of $EPU_o + EPU_i$ results in a smaller time response. In relation to Fig. 4.4, this conclusion can be reached by plotting and comparing each scenario with the relative approach angle θ, distance offset d, and radius r. The significance of target tracking algorithms is related to the availability

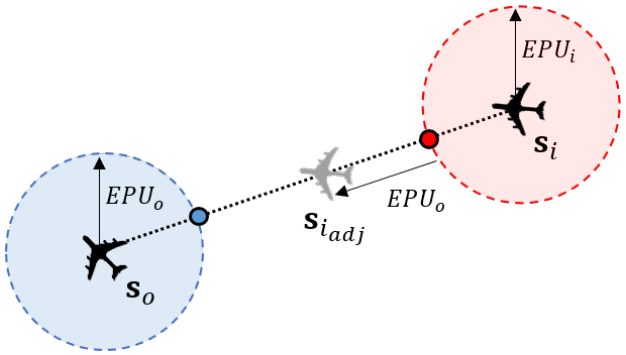


Figure 4.5. A correction for the estimated position uncertainty in the ownship and the intruder.

and quality of estimated aircraft state data. A filter utilizes the temporal history to calculate accurate state data, which enables a higher traffic density in the NAS. Furthermore, distributed target tracking algorithms (e.g., ACKF and ADKF) improve the availability of that estimated aircraft state data.

4.3.2 Selection of an inverse-variance weight

Two obvious options exist for selecting the inverse-variance weight $\mathbf{\Sigma}_k^i$ for calculating the pseudo measurement $\hat{\mathbf{y}}_k^i$ in the pre-processing at each sensor platform. Recall, the pseudo measurement is a weighted average of a set of measurements which has a smaller variance than that of each individual measurement. The first option is to utilize uncertainty quantification data like the EPU values such that the MV estimate is

$$\hat{\mathbf{y}}_{k,\mathrm{MV}} = \left(\sum_{i=1}^{N} \left[\mathbf{\Sigma}_k^i \right]^{-1} \right)^{-1} \left(\sum_{i=1}^{N} \left[\mathbf{\Sigma}_k^i \right]^{-1} \mathbf{y}_k^i \right), \tag{4.9a}$$

$$\mathrm{Var}(\hat{\mathbf{y}}_{k,\mathrm{MV}}) = \left(\sum_{i=1}^{N} \left[\mathbf{\Sigma}_k^i \right]^{-1} \right)^{-1}. \tag{4.9b}$$

If $\mathbf{\Sigma}_k^i$ is the same for all measurements, then the MV estimate simplifies to

$$\hat{\mathbf{y}}_{k,\mathrm{MV}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_k^i\,, \tag{4.10a}$$

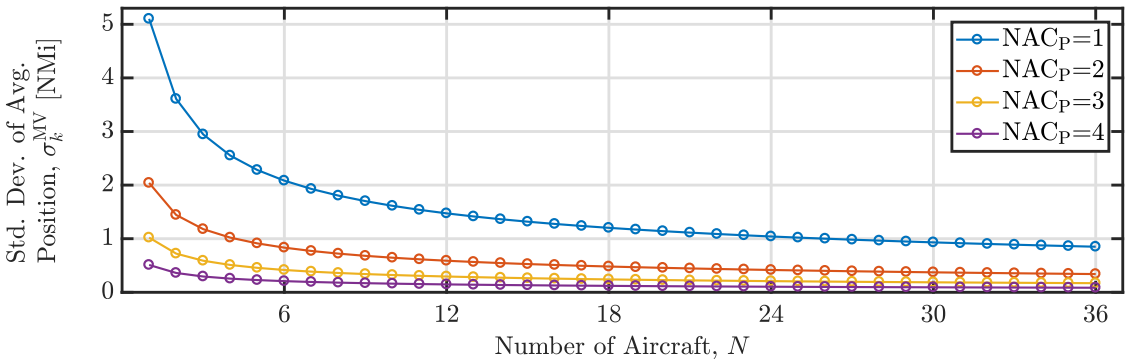
$$\mathrm{Var}(\hat{\mathbf{y}}_{k,\mathrm{MV}}) = \frac{1}{N}\mathbf{\Sigma}_k^i\,. \tag{4.10b}$$

This leads to the second option which is to classify the sensor platforms with $\mathbf{\Sigma}_k^i = 1$ for active sensors or $\mathbf{\Sigma}_k^i = \infty$ for inactive sensors such that

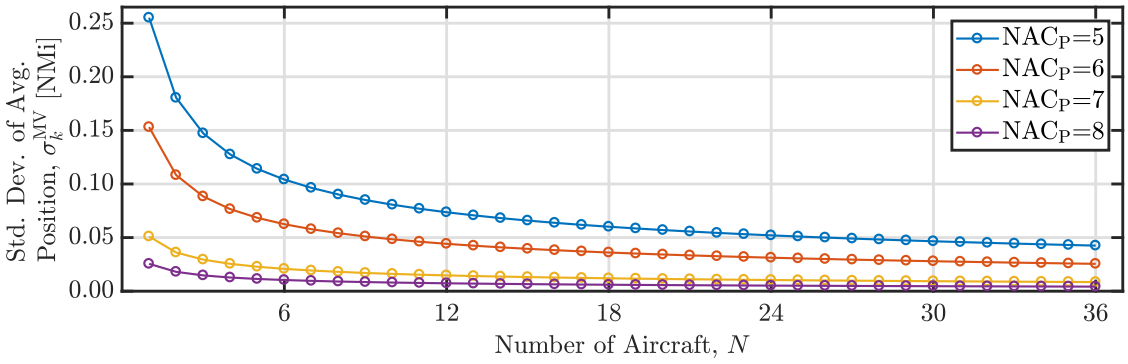
$$\hat{\mathbf{y}}_{k,\mathrm{MV}} = \frac{1}{N_a}\sum_{i=1}^{N_a}\mathbf{y}_k^i\,, \tag{4.11}$$

where N_a is the number of active sensors. For either option, incorporating an additional measurement reduces the variance of the MV estimate.

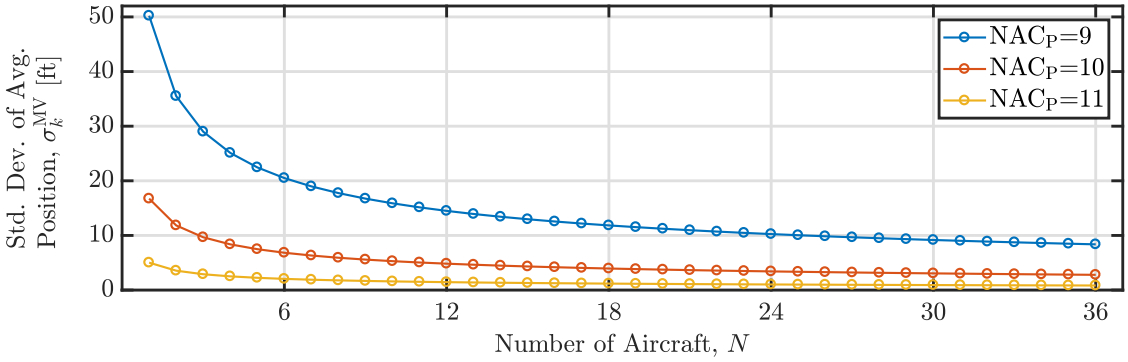
Consider a scenario where all N aircraft have the same $\mathrm{NAC_P}$ value, and thus the same EPU value. The standard deviation of the pseudo measurement for the position in Eq. (4.10b) is $\hat{\sigma}_{k,\mathrm{MV}} = \mathrm{sqrt}(\mathrm{Var}(\hat{\mathbf{y}}_{k,\mathrm{MV}}))$. In Fig. 4.6, each line corresponds to the scenario for different $\mathrm{NAC_P}$ values. A human air traffic controller can handle 12 to 18 aircraft in a sector referred to as the 1x and 1.5x traffic levels, respectively [73,74]. Upgrades for the Next Generation Air Transportation System (NextGen) is expected to triple the capacity of the airspace, which is why the plot range includes 36 aircraft (or the 3x traffic level). However, there are diminishing gains as 4 aircraft reduces the standard deviation by 50% while 16 aircraft reduces the standard deviation by 75%. Utilizing 4 to 8 aircraft is a realistic implementation to improve the availability and quality of estimated aircraft state data. Furthermore, the independently sensed data can enable additional fault detection and mitigation strategies.



(a) $\text{NAC}_\text{P} = \{1, 2, 3, 4\}$.



(b) $\text{NAC}_\text{P} = \{5, 6, 7, 8\}$.



(c) $\text{NAC}_\text{P} = \{9, 10, 11\}$.

Figure 4.6. The standard deviation of the average position as a function of the number of aircraft for a network in which all aircraft have the same NAC_P values.

4.3.3 Assessing safety for a horizontal airspace with the ACKF

This section presents an example in order to demonstrate the impact of distributed target tracking for assessing the safety of a horizontal airspace. The simulation includes running the ACKF and the AKF in order to highlight the difference in sensor platform i conditioning the state estimates on all of the pseudo measurements $\hat{\mathbf{y}}^i_{1:T}$ instead of the local measurements $\mathbf{y}^i_{1:T}$.

A common problem setup to test the performance of a conflict detection or resolution technology is to simulate multiple aircraft converging on the same region of an airspace. Figure 4.7 depicts such a scenario with six aircraft operating under nominal conditions. The dynamics of each aircraft were a random sample of the Nearly Constant Acceleration Model (NCAM) for a duration of 100 s. Each aircraft takes a position measurement of every aircraft with a standard deviation of 200 m. The objective is to assess the safety of AC1 in relation to the other five aircraft, which requires AC1 to estimate the trajectory of every aircraft (including itself).

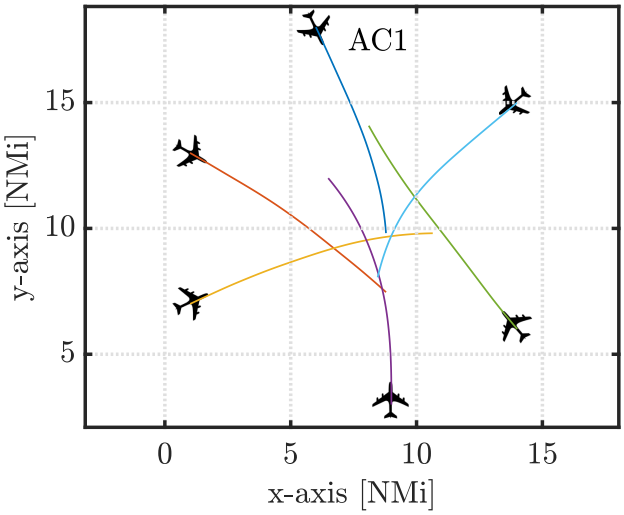


Figure 4.7. A six aircraft scenario generated with the NCAM.

The design choices for the AKF and ACKF are summarized in Table 4.2. The model was a zero-mean GP with the squared exponential covariance function

$$k_{\text{SE}}(t_i, t_j) = \sigma_{\text{SE}}^2 \exp\left(-\frac{\tau^2}{2\ell_{\text{SE}}^2}\right), \tag{4.12}$$

where $\tau = |t_i - t_j|$ and a measurement model that had additive Gaussian noise with covariance $R_k^i(\boldsymbol{\theta}) = [\sigma_n^i]^2$. The model was then converted to a discrete PSSM. The state estimator for both algorithms was the KF and RTSS while the parameter optimization utilized the quasi-Newton BFGS algorithm. The difference in the state estimator was that the ACKF used the pseudo measurements with the inverse-variance weight $\Sigma_k^i = 1$ while the AKF used the local measurements. Here, the ACKF was implemented with a fully connected communication network in which each Consensus Filter (CF) had the consensus gain $\epsilon = 1/7$ and $L = 1$ iterations for achieving consensus at each time instance k.

Table 4.2. Design choices for an example of the AKF and ACKF.

(1) **State-Space Model** - Squared Exponential Model
(2) **State Estimation** - KF and RTSS
(3) **Parameter Optimization** - Quasi-Newton BFGS
(4) **Parameters** $\boldsymbol{\theta} = [\,\sigma_n^2\,,\,\ell_{\text{SE}}\,,\,\sigma_{\text{SE}}^2\,]^\top$
(5) **Initial Values for Parameters** $\boldsymbol{\theta}_0 = [\,1\,,\,100\,,\,5000\,]^\top$
(6) **Prior Distributions** - Log-Uniform Distributions $p\big(\log\theta_j\big) \propto 1 \quad \forall j$
(7) **Consensus Filter** - Perron matrix with $\epsilon = 1/7$ and $L = 1$

For the safety assessment, Fig. 4.8 depicts the minimum time until a conflict will occur: $\min(t_{\text{ep}}, t_{\text{nmac}})$. Furthermore, any negative time values were set equal to zero as the conflict already exists. Five time responses exist for each algorithm, which

corresponds to the safety assessment of AC1 relative to the other five AC. The merged time response is not guaranteed to be continuous since the time variables are defined for different domains of the initial relative position and velocity in the horizontal airspace. For example, a jump discontinuity exists with the AKF at $k = 14$ and the ACKF at $k = 10$, which is the transition from t_{nmac} to t_{ep}. A jump discontinuity can only exist for a transition between a domain where t_{nmac} and t_{ep} are both defined and a domain where only one of these time variables is defined. Future research should focus on a merged response with a continuous transition.

A safety assessment based on the state estimates from a centralized or distributed target tracking algorithm provides a more accurate representation of safety than a safety assessment based on the raw noisy measurements. The ACKF provides a slightly more accurate representation of safety than the AKF because of the improvements in the state estimates. Generally, this translates to a difference of a few seconds for the predicted time until a conflict will occur. Further improvements in the quality of the estimated aircraft state data are expected with filtering of velocity and heading measurements and accounting for their uncertainty like the EPU correction.

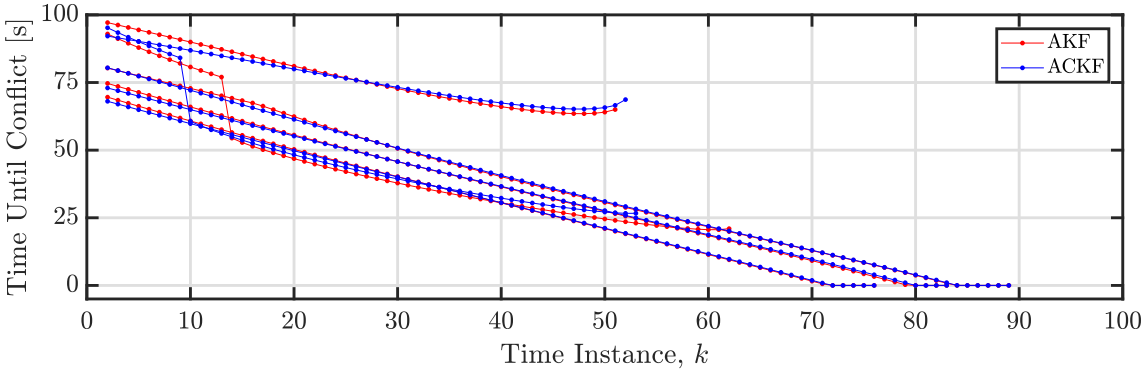


Figure 4.8. Minimum time until a loss of a separation minima using t_{ep} and t_{nmac} from the perspective of AC1.

Simulating an individual traffic scenario is exploratory work. There are two scenarios that should be investigated as parameter sweeps or Monte Carlo simulations

in order to provide a more rigorous conclusion about the exact benefits of the ACKF for civil aviation. The primary interest is a quantification of the improvement in the availability of the aircraft state estimates. The first scenario should consider the ability to assess safety for an off-nominal condition in which a fault exists in the communication or surveillance system. A system with independent data sources provides an opportunity for fault detection and mitigation, which can improve the robustness and resilience of the target tracking operations. However, this simulation is beyond the scope of this work as it requires a thorough analysis of the fault space such as in [77] by Sudarsanan et al. The second scenario that should be considered is a sensor network that contains sensors platforms with inactive sensors. Figure 4.9 depicts the communication network for a scenario with stationary sensor platforms and two aircraft with a line of sight that is obstructed by buildings. The use of CFs would enable consensus on state and parameter estimates despite each aircraft not being able to communicate directly with the other aircraft or measure the other aircraft's position. A similar scenario could arise in a dense airspace as communication devices have a fixed number of slots to communicate with other devices as well as signal acquisition problems with significant noise. However, such a simulation is also beyond the scope of this work as it requires analysis of an evolving scenario, which requires additional alterations to the target tracking and safety assessment algorithms.

4.3.4 Remarks on the ACKF and the ADKF for Civil Aviation

The two major differences between the ACKF and the ADKF are the communication cost and the performance. For applications where a good guess for the inverse-variance weight $\mathbf{\Sigma}_k^i$ can be determined, the ACKF is preferable to the ADKF because of the smaller communication cost. However, both algorithms are dependent on the inverse-variance weight for the parameter estimation. In order to make the ADKF a more viable option than the ACKF, the parameter estimation routine should

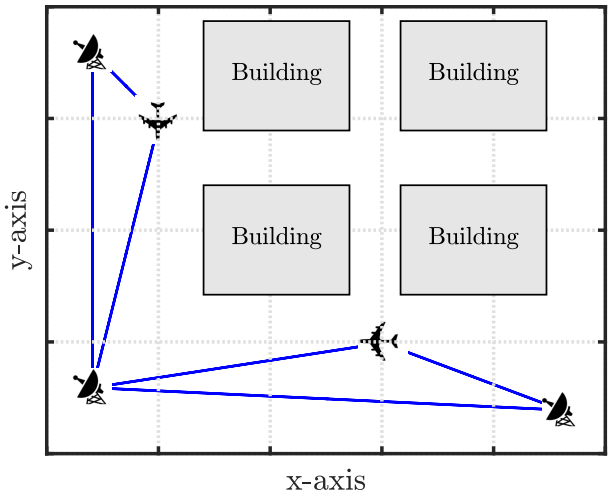


Figure 4.9. A communication network with stationary sensor platforms and two aircraft with an obstructed line of sight.

not be based on the pseudo measurements. Consequently, the ADKF would then be completely independent of the guessing the inverse-variance weights.

Constructing a communication network to implement the ACKF or the ADKF would require changes to ADS-B in terms of the message content, broadcast rate, and broadcast range. Instead of broadcasting the ownship's state data, ADS-B would need to broadcast the relevant message content (see Table 3.5) in order for each aircraft to maintain a consensus on the track of every aircraft. Some relevant research for altering the ADS-B performance characteristics was discussed by Duffield and McLain in [69] in the context of small Unmanned Aerial System (UAS).

For a real-world application, the difficulty of implementing either algorithm is designing a communication network with a sufficient response time. Limits can be placed on the function count D for the parameter optimization, the number of measurements T, and the number of aircraft N in the consensus routine. Additionally, the network connectivity impacts the number of consensus iterations L that are needed to maintain consensus. Most aircraft would fly at a high enough altitude to maintain

a fully connected network under nominal conditions. A fully connected network can achieve consensus in one round of communication with the appropriate value for the consensus gain ϵ. However, future applications like Urban Air Mobility (UAM) and UAS Traffic Management (UTM) will have aircraft flying in a city environment, which could result in a sparsely connected network and thus more consensus iterations L.

4.4 Summary and Contributions

This chapter demonstrated the role of the Adaptive Centralized Kalman Filter (ACKF) in civil aviation for improving a safety-critical airspace. A literature review summarized the time variable models of Well Clear (WC) and Critical Pair Identification (CPI) in a horizontal airspace. Although a literature review, the CPI description included two contributions: an improved grid search and mathematical notation that was consistent with the WC notation. Then, the parameter sweep of a pairwise conflict (PSPC) analysis was defined for a conflict with two aircraft over a variety of initial aircraft states. Using the PSPC method, merged responses for WC and CPI were computed, which revealed that conflict detection depended significantly on CPI's t_{nmac} and WC's t_{ep}. Intuitively, this result make sense as WC has a much larger horizontal distance threshold while CPI includes the blunder kinematics.

For a given scenario, a larger position uncertainty translates to a more severe scenario as quantified by the safety assessment. An example demonstrated that the safety assessment was improved by utilizing the ACKF over the AKF as a result of improved state and parameter estimates. The ACKF utilized the pseudo measurements $\hat{\mathbf{y}}^i_{1:T}$ while the AKF utilized the local measurements $\mathbf{y}^i_{1:T}$. The ACKF improved the quality of the state estimates, but the primary benefit for civil aviation is improving the availability of the state estimates at aircraft and air traffic control stations. However, more sophisticated simulations are required to quantify the benefits for the availability of state estimates for an airspace with faulty communication and surveillance devices. Since the ACKF and the ADKF both depend upon the

inverse-variance weights to calculate the pseudo measurements, the ACKF is currently the recommended algorithm based on the significantly smaller communication cost. If the parameter estimation routine for the ADKF was no longer dependent on the pseudo measurements, then the ADKF could be a more robust solution than the ACKF at the expense of a higher communication cost.

5. CONCLUSION

This dissertation analyzed a distributed estimation problem for a sensor network tracking an unknown target. The problem consisted of estimating the state of a dynamical system and the parameters defining a model of that system through local communications between sensor platforms. The sensors platforms broadcast messages to neighbors with the intent of maintaining consensus on state and parameter estimates. The primary objective was to answer the two following questions. *How can a sensor network maintain consensus while estimating the state of a dynamical system and the parameters defining a model of that system? Can an adaptive formulation of a Distributed Kalman Filter (DKF) guarantee desirable results for real-time implementation?*

Chapter 2 focused on the selection of an Adaptive Kalman Filter (AKF) and the underlying system as a parameterized state-space model (PSSM). The PSSM was compatible with machine learning models from Adaptive Gaussian Process Regression (AGPR) and the dynamical models from target tracking theory. A few example problems demonstrated the applicability of these models for target tracking applications. The squared exponential covariance function provided comparable results to the target tracking models with potential computational benefits. Otherwise, the joint use of these models can provide enhanced capabilities in aerospace applications such as with fault detection and surrogate modeling of physical processes.

Chapter 3 defined the target dynamics and sensor network's measurement process as a parameterized distributed state-space model (PDSSM). The Adaptive Centralized Kalman Filter (ACKF) and the Adaptive Distributed Kalman Filter (ADKF) were presented as solutions for the distributed estimation problem in terms of state, parameter, and consensus estimation. These two solutions were designed to highlight the difference between utilizing a KF and DKF as the state estimator. The

contributions included the derivations as well as an analysis of their performance characteristics: computational complexity, communication cost, optimality, stability, and simulation-based performance. The ADKF has a much larger communication cost the ACKF and is likely to present a significant challenge to realize a communication network with a sufficient broadcast rate for target tracking applications. Under nominal conditions, the state estimators for the ACKF and the ADKF provided very similar results. However, only the state estimator for the ACKF requires a guess of the inverse-variance weights in order to calculate the pseudo measurements. Recall, the pseudo measurement is a weighted average of a set of measurements which has a smaller variance than that of each individual measurement. As a result, the state estimator in the ACKF can provide poor results under off-nominal conditions.

Chapter 4 applied the ACKF towards civil aviation as a method for improving the availability and quality of estimated aircraft state data for a safety-critical airspace. A safety assessment for a pairwise conflict was found to significantly depend on the horizontal time models t_{ep} and t_{nmac} from Well Clear (WC) and Critical Pair Identification (CPI), respectively. An example demonstrated that the safety assessment was improved by utilizing the ACKF over the AKF as a result of improved state and parameter estimates. For civil aviation, an important benefit of the ACKF is the improvement in the availability of the estimated aircraft state data at the end users. Recall, the availability of state data refers to the degree to which aircraft and air traffic control stations can calculate state estimates of each aircraft for traffic management. More sophisticated simulations are required to quantify the benefits for the availability of state estimates for an airspace with faulty communication and surveillance devices. Currently, the ACKF is the recommended algorithm over the ADKF based on the significantly smaller communication cost for similar results. The ACKF and the ADKF both require guessing the inverse-variance weights in order to calculate the pseudo measurements. In order to make the ADKF a more viable option than the ACKF, the parameter estimation routine should be based on the network's residuals and not the pseudo measurements. The ADKF would then be completely

independent of guessing the inverse-variance weights and a more robust solution for civil aviation.

5.1 Significance

The advantages of a distributed system over a centralized system can include cost, robustness, resilience, accuracy, coverage, and scalability. Not all benefits are guaranteed, but a sensor network can be more desirable than a single high-end platform. In order to improve safety-critical systems, implementing distributed functions like target tracking are necessary to maintain accessible and accurate information, which in turn improves a system's robustness and resilience. The two algorithms discussed in this dissertation presented the capability for distributed state and parameter estimation with a PDSSM. The objective of these solutions was to determine an algorithm with desirable performance characteristics for on-line computations. Only the ACKF algorithm advanced the capabilities for distributed state and parameter estimation towards real-world applications in safety-critical systems. With adjustments to the parameter estimation routine, the ADKF could become a more robust solution than ACKF at the expense of a higher communication cost. However, the maximum a posteriori (MAP) parameter estimation methodology still presents difficulties for guaranteeing fast and accurate solutions in a large variety of scenarios.

Distributed estimation algorithms are necessary to improve the availability and quality of estimated state data. For civil aviation, the upgrade of legacy systems is underway in NextGen with increased automation of communication, surveillance, and navigation technologies. The flight-tracking industry is providing new predictive capabilities for flight operations and resource allocation. Further improvements to the robustness and resilience of target tracking capabilities are necessary to increase the efficiency and safety of the airspace as well as seamlessly integrate new aerial concepts like Urban Air Mobility (UAM) and Unmanned Aircraft System Traffic Management (UTM). Similar target tracking capabilities are necessary with military

applications for tracking ground, aerial, naval, and space targets. With continued automation, these distributed systems can be intelligent enough to self-govern, and minimize the necessity for a human-in-the-loop and occurrences of bottlenecks in the information flow.

5.2 Future Work

The long term objective is to determine a robust solution for the distributed estimation of states and parameters for the on-line implementation in real sensor networks. The following topics are potential extensions of this research.

- **Improving the Robustness of the ADKF over the ACKF**: The objective function and gradient for the parameter estimation routine is dependent on the pseudo measurements for both solutions. For the ADKF to be a more robust solution, the parameter estimation routine should be based on the network's residuals instead of the pseudo measurements. Then, the parameter estimates would be based most up-to-date measurement noise variance $\mathbf{R}_k^i(\boldsymbol{\theta})$ instead of the initial guesses for the inverse-variance weight $\boldsymbol{\Sigma}_k^i$.

- **Improving the Computational Complexity**: The computational complexity can be reduced by altering the method for calculating an objective function and the gradient. Since the sensitivity equations utilized termwise differentiation, approximately $p + 1$ state estimators were required for p parameters. Alternatively, Fisher's identity allows the gradient of the energy function to be defined in terms of the smoothed estimates from the RTSS. The method is computationally lighter for linear state-space models allowing approximately two state estimators. The term $p+1$ can be reduced to 2 by utilizing the Fisher's identity [2, 79–81].

- **Improving the Consensus Estimation**: The best consensus methods are still open research areas, but are dependent on the information each node knows

about the network. An alternative weighting scheme could increase the rate of convergence, which would be extremely useful for a bandwidth limited communication network.

- **Additional Functionality in Distributed Estimation**: ACKF provides a base framework to incorporate additional statistical tools (e.g., deep learning, Gaussian mixtures, classification) and estimation problems (e.g., track association, data discrimination, and sensor-target allocation). A particular area of interest for civil aviation is to incorporate waypoint information using a method based on Interacting Multiple Models as demonstrated by Hwang [82].

- **Applications with Machine Learning and Dynamical Models**: The machine learning models have a wide range of applications, but their joint application with dynamical models in physical systems still warrants exploration. This dissertation explored a couple potential aerospace applications, which indicated potential usage in fault detection of state data as well as surrogate modeling of physical processes with deep learning.

- **Evolution of Technologies in Civil Aviation**: With the NextGen architecture, the difficulty of performing distributed estimation and fault detection techniques is related to a lack of independent sensors and the availability of their measurements at the end users. More research is needed on the cost and benefits of the required modifications to the architecture of the surveillance and communication operations to enable these distributed operations.

REFERENCES

REFERENCES

[1] Ahmed T. Kamal, Jay A. Farrell, and Amit K. Roy-Chowdhury. Information weighted consensus filters and their application in distributed camera networks. *IEEE Transactions on Automatic Control*, 58(12):3112–3125, Dec 2013.

[2] Simo Särkkä. *Bayesian filtering and smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, Cambridge, U.K., 2013.

[3] NASA. Urban Air Mobility Grand Challenge. `https://www.nasa.gov/uamgc`, November 2018.

[4] NASA. Unmanned Aircraft System (UAS) Traffic Management (UTM). `https://www.utm.arc.nasa.gov/index.shtml`, September 2018.

[5] Raman K. Mehra. Approaches to adaptive filtering. In *1970 IEEE Symposium on Adaptive Processes (9th) Decision and Control*, pages 141–141, Dec 1970.

[6] Chee-Yee Chong, David Garren, and Timothy P. Grayson. Ground target tracking-a historical perspective. In *2000 IEEE Aerospace Conference. Proceedings (Cat. No.00TH8484)*, volume 3, pages 433–448, 2000.

[7] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and tracking : principles, techniques, and software*. Artech House, Boston, 1993.

[8] X. Rong Li and Vesselin P. Jilkov. Survey of maneuvering target tracking: dynamic models. In *AeroSense 2000*, volume 4048, pages 212–235, July 2000.

[9] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2006.

[10] David Kristjanson Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, Pembroke College, University of Cambridge, 2014.

[11] Rudolf E. Kálmán. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, Mar 1960.

[12] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, Aug 1965.

[13] Jarno Vanhatalo, Jaakko Riihimäki, Jouni Hartikainen, Pasi Jylänki, Ville Tolvanen, and Aki Vehtari. Gpstuff: Bayesian modeling with gaussian processes. *Journal of Machine Learning Research*, 14:1175–1179, April 2013.

[14] Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal gaussian process regression models. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pages 379–384, Aug 2010.

[15] Steven Reece and Stephen Roberts. An introduction to gaussian processes for the kalman filter expert. In *2010 13th International Conference on Information Fusion*, pages 1–9, July 2010.

[16] Simo Särkkä and Jouni Hartikainen. Infinite-dimensional kalman filtering approach to spatio-temporal gaussian process regression. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 22, pages 993–1001, 2012.

[17] Simo Särkkä, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional bayesian filtering and smoothing: A look at gaussian process regression through kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, July 2013.

[18] Roger Frigola, Fredrik Lindsten, Thomas B. Schön, and Carl E. Rasmussen. Identification of gaussian process state-space models with particle stochastic approximation em. December 2013.

[19] Arno Solin and Simo Särkkä. Explicit link between periodic covariance functions and state space models. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 33, pages 904–912, 2014.

[20] S. Reece and S. Roberts. The near constant acceleration gaussian process kernel for tracking. *IEEE Signal Processing Letters*, 17(8):707–710, Aug 2010.

[21] Robert A. Singer. Estimating optimal tracking filter performance for manned maneuvering targets. *IEEE Transactions on Aerospace and Electronic Systems*, AES-6(4):473–483, July 1970.

[22] Ladislav Král, Jakub Prüher, and Miroslav Šimandl. Gaussian process based dual adaptive control of nonlinear stochastic systems. In *22nd Mediterranean Conference on Control and Automation*, pages 1074–1079, June 2014.

[23] Sandeep Katragadda, Juan C. SanMiguel, and Andrea Cavallaro. The costs of fusion in smart camera networks. In *Proceedings of the International Conference on Distributed Smart Cameras*, Nov 2014.

[24] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9(1):427–438, Feb 2013.

[25] Federica Garin and Luca Schenato. *A Survey on Distributed Estimation and Control Applications Using Linear Consensus Algorithms*, volume 406, pages 75–107. Springer, London, 2010.

[26] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, Jan 2007.

[27] Lin Xiao, Stephen Boyd, and Sanjay Lall. A scheme for robust distributed sensor fusion based on average consensus. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 63–70, April 2005.

[28] Doron J. Shahar. Minimizing the variance of a weighted average. *Open Journal of Statistics*, 7(2):216–224, April 2017.

[29] Magdi S. Mahmoud and Haris M. Khalid. Distributed kalman filtering: a bibliographic review. *IET Control Theory Applications*, 7(4):483–501, March 2013.

[30] Jason L. Speyer. Computation and transmission requirements for a decentralized linear-quadratic-gaussian control problem. *IEEE Transactions on Automatic Control*, 24(2):266–269, April 1979.

[31] Reza Olfati-Saber and Richard M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, Sept 2004.

[32] Reza Olfati-Saber. Distributed kalman filter with embedded consensus filters. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 8179–8184, Dec 2005.

[33] Reza Olfati-Saber. Distributed kalman filtering for sensor networks. In *2007 46th IEEE Conference on Decision and Control*, pages 5492–5498, Dec 2007.

[34] Reza Olfati-Saber and Nils F. Sandell. Distributed tracking in sensor networks with limited sensing range. In *2008 American Control Conference*, pages 3157–3162, June 2008.

[35] Reza Olfati-Saber. Kalman-consensus filter : Optimality, stability, and performance. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 7036–7042, Dec 2009.

[36] Chong Ding, Bi Song, Akshay Morye, Jay A. Farrell, and Amit K. Roy-Chowdhury. Collaborative sensing in a distributed ptz camera network. *IEEE Transactions on Image Processing*, 21(7):3282–3295, July 2012.

[37] Bi Song, Ahmed T. Kamal, Cristian Soto, Chong Ding, Jay A. Farrell, and Amit K. Roy-Chowdhury. Tracking and activity recognition through consensus in distributed camera networks. *IEEE Transactions on Image Processing*, 19(10):2564–2579, Oct 2010.

[38] Ahmed T. Kamal, Chong Ding, Bi Song, Jay A. Farrell, and Amit K. Roy-Chowdhury. A generalized kalman consensus filter for wide-area video networks. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 7863–7869, Dec 2011.

[39] David W. Casbeer and Randy Beard. Distributed information filtering using consensus filters. In *2009 American Control Conference*, pages 1882–1887, June 2009.

[40] Xie Li, Huang Caimou, and Hu Haoji. Distributed filter with consensus strategies for sensor networks. *Journal of Applied Mathematics*, 2013, 2013.

[41] Nicholas Assimakis, Maria Adam, and Anargyros Douladiris. Information filter and kalman filter comparison: Selection of the faster filter. *International Journal of Information Engineering*, 2(1):1–5, Mar 2012.

[42] Sandeep Katragadda, Juan C. SanMiguel, and Andrea Cavallaro. Consensus protocols for distributed tracking in wireless camera networks. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8, July 2014.

[43] Narendra K. Gupta and Raman K. Mehra. Computational aspects of maximum likelihood estimation and reduction in sensitivity function calculations. *IEEE Transactions on Automatic Control*, 19(6):774–783, Dec 1974.

[44] Oliver Cappé, Eric Moulines, and Tobias Rydén. *Inference in hidden Markov models*. Springer series in statistics. Springer, London, U.K., 2005.

[45] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[46] Ahmed T. Kamal, Jay A. Farrell, and Amit K. Roy-Chowdhury. Information weighted consensus. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 2732–2737, Dec 2012.

[47] Jasbir S. Arora. *Introduction to Optimum Design (Third Edition)*. Academic Press, Waltham, MA, third edition edition, 2012.

[48] J. S. Morrel. Fundamental physics of the aircraft collision problem. Technical Report 465-1016-39, Bendix Aviation Corporation, 1956.

[49] P. G. Reich. Analysis of long-range air traffic systems: Separation standardsi. *Journal of Navigation*, 19(1):88–98, 1966.

[50] P. G. Reich. Analysis of long-range air traffic systems: Separation standardsii. *Journal of Navigation*, 19(2):169–186, 1966.

[51] P. G. Reich. Analysis of long-range air traffic systems: Separation standardsiii. *Journal of Navigation*, 19(3):331–347, 1966.

[52] B. Alexander. Aircraft density and midair collision. *Proceedings of the IEEE*, 58(3):377–381, 1970.

[53] J.K. Kuchar and L.C. Yang. Survey of conflict detection and resolution modeling methods. In *Guidance, Navigation, and Control Conference*, pages 1388–1397. AIAA, 1997.

[54] T.A. Lauderdale. Probabilistic conflict detection for robust detection and resolution. In *12th AIAA Aviation Technology, Integration and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012.

[55] Henk A. P. Blom and G. J. Bakker. Safety evaluation of advanced self-separation under very high en route traffic demand. *Journal of Aerospace Information Systems*, 12(6):413–427, 2015.

[56] Laith R. Sahawneh, James Mackie, Jonathan Spencer, Randal W. Beard, and Karl F. Warnick. Airborne radar-based collision detection and risk estimation for small unmanned aircraft systems. *Journal of Aerospace Information Systems*, 12(12):756–766, 2015.

[57] Kwangyeon Kim and Inseok Hwang. Intent-based detection and characterization of aircraft maneuvers in en route airspace. *Journal of Aerospace Information Systems*, 15(2):72–91, 2018.

[58] I. Hwang, H. Balakrishnan, and C. Tomlin. State estimation for hybrid systems: Applications to aircraft tracking. *IEEE Proceedings - Control Theory and Applications*, 153(5):556–566, Sept 2006.

[59] Jared K. Wikle, Timothy W. McLain, Randal W. Beard, and Laith R. Sahawneh. Minimum required detection range for detect and avoid of unmanned aircraft systems. *Journal of Aerospace Information Systems*, 14(7):351–372, 2017.

[60] Evan Maki, Andrew Weinert, and Mykel Kochenderfer. Efficiently estimating ambient near mid-air collision risk for unmanned aircraft. In *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*. AIAA, 2010.

[61] Introduction to tcas ii version 7.1. Technical report, U.S. Department of Transportation FAA, 2011.

[62] C. Muñoz, A. Narkawicz, and J. Chamberlain. A tcas-ii resolution advisory detection algorithm. In *Proceedings of the AIAA Guidance Navigation, and Control Conference and Exhibit 2013*, 2013.

[63] A. J. Narkawicz, C. A. Muñoz, J. M. Upchurch, J. P. Chamberlain, and M. C. Consiglio. A well-clear volume based on time to entry point. Technical Memo NASA/TM2014-218155, NASA, Langley Research Center, Hampton, VA, 2014.

[64] J. M. Upchurch, C. Muñoz, A. J. Narkawicz, J. P. Chamberlain, and M. C. Consiglio. Analysis of well-clear boundary models for the integration of uas in the nas. Technical Memo NASA/TM2014218280, NASA, Langley Research Center, Hampton, VA, 2014.

[65] Ricardo Arteaga, Mike Dandachy, Hong Truong, Arun Aruljothi, Mihir Vedantam, Kraettli Epperson, and Reed McCartney. ads-b detect and avoid flight tests on phantom 4 unmanned aircraft system. In *AIAA Science and Technology Forum and Exposition (SciTech)*, 2018.

[66] S. Man Lee, C. Park, M.A. Johnson, and E.R. Mueller. Investigating effects of "well clear" definitions on uas sense-and-avoid operations. In *Aviation Technology, Integration, and Operations Conference*. AIAA, 2013.

[67] S. P. Cook, D. Brooks, R. Cole, D. Hackenberg, and V. Raska. Defining well clear for unmanned aircraft systems. In *AIAA Infotech at Aerospace*. AIAA, 2015.

[68] Andrew Weinert, Scot Campbell, Adan Vela, Dieter Schuldt, and Joel Kurucar. Well-clear recommendation for small unmanned aircraft systems based on unmitigated collision risk. *Journal of Air Transportation*, 26(3):113–122, 2018.

[69] M. O. Duffield and T. W. McLain. A well clear recommendation for small uas in high-density, ads-b-enabled airspace. In *AIAA Information Systems-AIAA Infotech at Aerospace*. AIAA, 2017.

[70] S. H. Kim. Conflict risk analysis of small unmanned aircraft systems. *Journal of Aerospace Information Systems*, 15(12):684–695, 2018.

[71] S. J. Landry. Intensity control: a concept for automated separation assurance safety and function allocation in nextgen. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSM*, 2012.

[72] S. J. Landry and Z. Zhang. The critical pair problem as a measure of the safety of separation assurance systems in air traffic control. In *Human Factors and Ergonomics Society (HFES) Annual Meeting*, 2017.

[73] C. A. Surakitbanharn. Evaluating intensity as a controller function for nextgen scenarios with increased capacity. Master's thesis, School of Industrial Engineering, Purdue University, 2014.

[74] C. A. Surakitbanharn. *Analyzing Critical Pair Identification as a Human-Controlled Function in Air Traffic Control*. PhD thesis, School of Industrial Engineering, Purdue University, 2017.

[75] M. Jacobs, V. Sudarsanan, S. Subramanian, D. DeLaurentis, Z. Zhang, and S. Landry. Measuring the impact of avionics faults with a set of safety metrics. In *IEEE International Conference on System, Man, and Cybernetics (SMC) 2017*, 2017.

[76] E. Theunissen, B. Suarez, , and F. Kunzi. Well clear recovery for detect and avoid. In *2016 IEEE/AIAA 35th Digital Avionics Syst. Conf. (DASC)*, 2016.

[77] V. Sudarsanan, M. Jacobs, A. Dervisevic, and D. DeLaurentis. Ads-b and cpdlc fault modeling for safety assessment in a distributed environment. In *IEEE Aerospace Conference*, 2018.

[78] Procedures for air navigation services, air traffic management. Technical Report Doc 4444, 16th ed., International Civil Aviation Organization, 2016.

[79] M. Segal and E. Weinstein. A new method for evaluating the log-likelihood gradient, the hessian, and the fisher information matrix for linear dynamic systems. *IEEE Transactions on Information Theory*, 35(3):682–687, May 1989.

[80] R. K. Olsson, K. B. Petersen, and T. Lehn-Schiler. State-space models: From the em algorithm to a gradient approach. *Neural Computation*, 19(4):1097–1111, April 2007.

[81] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the em algorithm. *Journal of Time Series Analysis*, 3(4):253–26, July 1982.

[82] I. Hwang, H. Balakrishnan, and C. Tomlin. State estimation for hybrid systems: applications to aircraft tracking. *IEEE Proceedings - Control Theory and Applications*, 153(5):556–566, Sept 2006.

APPENDICES

A. PROPERTIES

This appendix covers relevant concepts regarding Bayes' Theorem, Gaussian distribution, matrix derivatives, continuous and discrete consensus algorithms, and the computational cost of various algorithms.

A.1 Bayes' Theorem

Theorem A.1.1 *Bayes's Theorem for Parameter Estimation: The posterior distribution for the parameter* $\boldsymbol{\theta}$ *given the measurements* $\mathbf{y}_{1:T}$ *via Bayes theorem is*

$$p\big(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}\big) = \frac{p\big(\boldsymbol{\theta}\big) \cdot p\big(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}\big)}{p\big(\mathbf{y}_{1:T}\big)}\,, \tag{A.1}$$

where the terms are defined as:

posterior	$p\big(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}\big)\,,$
prior	$p\big(\boldsymbol{\theta}\big)\,,$
likelihood	$p\big(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}\big)\,,$
normalization term	$p\big(\mathbf{y}_{1:T}\big)\,.$

Proof The joint probability for A and B may be defined as

$$p\big(A, B\big) = p\big(A\big) \cdot p\big(B \mid A\big) \tag{A.2}$$

or

$$p\big(A, B\big) = p\big(B\big) \cdot p\big(A \mid B\big)\,. \tag{A.3}$$

Setting Eq. (A.2) equal to Eq. (A.3) gives

$$p\big(A\big) \cdot p\big(B \mid A\big) = p\big(B\big) \cdot p\big(A \mid B\big)\,. \tag{A.4}$$

Thus, the posterior for B conditioned on A is

$$p\big(B \mid A\big) = \frac{p\big(B\big) \cdot p\big(A \mid B\big)}{p\big(A\big)}\,, \tag{A.5}$$

where the terms are defined as:

posterior	$p\big(B \mid A\big)\,,$
prior	$p\big(B\big)\,,$
likelihood	$p\big(A \mid B\big)\,,$
normalization term	$p\big(A\big)\,.$

■

Theorem A.1.2 *Bayes's Theorem for Parameter and State Estimation: The posterior distribution for the parameter* $\boldsymbol{\theta}$ *and states* $\mathbf{x}_{0:T}$ *given the measurements* $\mathbf{y}_{1:T}$ *via Bayes theorem is*

$$p\big(\boldsymbol{\theta}, \mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}\big) = \frac{p\big(\boldsymbol{\theta}\big) \cdot p\big(\mathbf{x}_{0:T} \mid \boldsymbol{\theta}\big) \cdot p\big(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}, \mathbf{x}_{0:T}\big)}{p\big(\mathbf{y}_{1:T}\big)}\,, \tag{A.6}$$

where the terms are defined as:

joint posterior	$p\big(\boldsymbol{\theta}, \mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}\big)\,,$
prior	$p\big(\boldsymbol{\theta}\big)\,,$
process model	$p\big(\mathbf{x}_{0:T} \mid \boldsymbol{\theta}\big)\,,$
measurement model	$p\big(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}, \mathbf{x}_{0:T}\big)\,,$
normalization term	$p\big(\mathbf{y}_{1:T}\big)\,.$

Proof The joint probability for A, B, and C may be defined as

$$p\big(A, B, C\big) = p\big(A\big) \cdot p\big(B, C \mid A\big) \tag{A.7}$$

or

$$p\big(A, B, C\big) = p\big(B\big) \cdot p\big(A, C \mid B\big)\,. \tag{A.8}$$

Expanding the likelihood in Eq. (A.8) yields

$$p\big(A, C \mid B\big) = p\big(C \mid B\big) \cdot p\big(A \mid B, C\big)\,. \tag{A.9}$$

Setting Eq. (A.7) equal to Eq. (A.8) with the expanded likelihood in Eq. (A.9) gives

$$p\big(A\big) \cdot p\big(B, C \mid A\big) = p\big(B\big) \cdot p\big(C \mid B\big) \cdot p\big(A \mid B, C\big)\,. \tag{A.10}$$

Thus, the joint posterior for B and C conditioned on A is

$$p\big(B, C \mid A\big) = \frac{p\big(B\big) \cdot p\big(C \mid B\big) \cdot p\big(A \mid B, C\big)}{p\big(A\big)}\,, \tag{A.11}$$

where the terms are defined as:

joint posterior	$p\big(B, C \mid A\big)\,,$
prior	$p\big(B\big)\,,$
likelihood	$p\big(C \mid B\big) \cdot p\big(A \mid B, C\big)\,,$
normalization term	$p\big(A\big)\,.$

■

A.2 Gaussian Distribution

The probability density function for a multivariate Gaussian distribution with random vector $\mathbf{x} \in \mathbb{R}^{n\times 1}$, mean $\boldsymbol{\mu} \in \mathbb{R}^{n\times 1}$, and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ is defined as

$$\mathcal{N}\big(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\big) = \frac{\exp\Big(-\frac{1}{2}\left[\mathbf{x}-\boldsymbol{\mu}\right]^\top \boldsymbol{\Sigma}^{-1}\left[\mathbf{x}-\boldsymbol{\mu}\right]\Big)}{\sqrt{|2\pi\boldsymbol{\Sigma}|}}\,. \tag{A.12}$$

Note, $(2\pi)^n\,|\boldsymbol{\Sigma}| = |2\pi\boldsymbol{\Sigma}|$.

Lemma A.2.1 (Joint distribution of Gaussian variables) *If the random variables $\mathbf{x}_1 \in \mathbb{R}^{n\times 1}$ and $\mathbf{x}_2 \in \mathbb{R}^{m\times 1}$ have the Gaussian distributions*

$$p\big(\mathbf{x}_1\big) = \mathcal{N}\big(\mathbf{x}_1 \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\big)\,, \tag{A.13a}$$

$$p\big(\mathbf{x}_2 \mid \mathbf{x}_1\big) = \mathcal{N}\big(\mathbf{x}_2 \mid \mathbf{C}\mathbf{x}_1, \boldsymbol{\Sigma}_2\big)\,, \tag{A.13b}$$

then the joint distribution of $\mathbf{x}_1$ and $\mathbf{x}_2$ is

$$\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} \sim \mathcal{N}\Big(\begin{bmatrix}\boldsymbol{\mu}_1\\ \mathbf{C}\boldsymbol{\mu}_1\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_1\mathbf{C}^\top\\ \mathbf{C}\boldsymbol{\Sigma}_1 & \mathbf{C}\boldsymbol{\Sigma}_1\mathbf{C}^\top + \boldsymbol{\Sigma}_2\end{bmatrix}\Big) \tag{A.14}$$

if $\mathrm{Cov}(\mathbf{x}_2) = \mathbf{C}\boldsymbol{\Sigma}_1\mathbf{C}^\top + \boldsymbol{\Sigma}_2$.

Lemma A.2.2 (Conditional distribution of Gaussian variables) *If the random variables $\mathbf{x}_1 \in \mathbb{R}^{n\times 1}$ and $\mathbf{x}_2 \in \mathbb{R}^{m\times 1}$ have the joint Gaussian distribution*

$$\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} \sim \mathcal{N}\Big(\begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{12}^\top & \boldsymbol{\Sigma}_{22}\end{bmatrix}\Big)\,, \tag{A.15}$$

then the marginal and conditional distributions of $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *are*

$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1 \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}), \tag{A.16a}$$

$$p(\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_2 \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}), \tag{A.16b}$$

$$p(\mathbf{x}_1 \mid \mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1 \mid \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{12}^{\top}), \tag{A.16c}$$

$$p(\mathbf{x}_2 \mid \mathbf{x}_1) = \mathcal{N}(\mathbf{x}_2 \mid \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{12}^{\top}\boldsymbol{\Sigma}_{11}^{-1}(\mathbf{x}_1 - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{12}^{\top}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}). \tag{A.16d}$$

Lemma A.2.3 (Product of Gaussian probability density functions) *The product of two Gaussian probability density functions is an un-normalized Gaussian probability density function:*

$$\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \cdot \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \propto \mathcal{N}(\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3) \tag{A.17}$$

where

$$\boldsymbol{\mu}_3 = \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right), \tag{A.18a}$$

$$\boldsymbol{\Sigma}_3 = \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1}. \tag{A.18b}$$

A.3 Matrix Derivatives

Some useful matrix identities include the chain rule, the derivative for a matrix inverse, and derivative of a logarithm:

$$\frac{\partial \mathbf{A}\mathbf{B}}{\partial x} = \frac{\partial \mathbf{A}}{\partial x}\mathbf{B} + \mathbf{A}\frac{\partial \mathbf{B}}{\partial x}, \tag{A.19a}$$

$$\frac{\partial \mathbf{A}^{-1}}{\partial x} = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{A}^{-1}, \tag{A.19b}$$

$$\frac{\partial \log|\mathbf{A}|}{\partial x} = \mathrm{Tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\right). \tag{A.19c}$$

Here, the matrices $\mathbf{A}$ and $\mathbf{B}$ are functions of the scalar x, $\mathrm{Tr}(\cdot)$ is the trace, and $|\cdot|$ is the determinant.

A.4 Continuous and Discrete Consensus Algorithms

The continuous-time and discrete-time, linear update dynamics are

$$\dot{\mathbf{x}}_t = \mathbf{A}_t \mathbf{x}_t \,, \tag{A.20a}$$

$$\mathbf{x}_{\ell+1} = \mathbf{W}_\ell \mathbf{x}_\ell \,. \tag{A.20b}$$

The i^{th} element of the continuous-time and discrete-time state vector are $\mathbf{x}_t = [x_t^i]$ and $\mathbf{x}_\ell = [x_\ell^i]$. The local computation at node i is

$$\dot{x}_t^i = \sum_{j=1}^{N} a_t^{ij} x_t^i \,, \tag{A.21a}$$

$$x_{\ell+1}^i = \sum_{j=1}^{N} w_\ell^{ij} x_\ell^i \,, \tag{A.21b}$$

where the ij^{th} element is denoted as $\mathbf{A}_t = \left[a_t^{ij}\right]$ and $\mathbf{W}_\ell = \left[w_\ell^{ij}\right]$. For the continuous-time case, $\mathbf{A}_t$ is a Metzler matrix — a matrix whose off-diagonal elements are non-negative (i.e., $a_t^{ij} \geq 0 \; \forall i \neq j$) — and the sum of each row is zero (i.e., $\mathbf{A}_t \mathbf{1} = \mathbf{0}$). For the discrete-time case, $\mathbf{W}_\ell$ is a left stochastic matrix — a matrix with non-zero elements (i.e., $w_\ell^{ij} \geq 0 \; \forall i, j$) and the sum of each row is one (i.e., $\mathbf{W}_\ell \mathbf{1} = \mathbf{1}$).

A static consensus strategy (i.e., $\mathbf{A}_t = \mathbf{A}$ and $\mathbf{W}_\ell = \mathbf{W}$) for average consensus uses the Laplacian matrix $\boldsymbol{\mathcal{L}}$ for the continuous-time case and the Perron matrix $\boldsymbol{\mathcal{P}}$ for the discrete-time case such that

$$\dot{\mathbf{x}}_t = -\boldsymbol{\mathcal{L}} \mathbf{x}_t \,, \tag{A.22a}$$

$$\mathbf{x}_{\ell+1} = \boldsymbol{\mathcal{P}} \mathbf{x}_\ell \,. \tag{A.22b}$$

Recall, the Laplacian and Perron matrix are related: $\boldsymbol{\mathcal{P}} = \mathbf{I} - \epsilon\boldsymbol{\mathcal{L}}$. The local computation at node i is

$$\dot{x}_t^i = \sum_{j \in \mathcal{N}(i)} \left(x_t^j - x_t^i\right) , \tag{A.23a}$$

$$x_{\ell+1}^i = x_\ell^i + \epsilon \sum_{j \in \mathcal{N}(i)} \left(x_\ell^j - x_\ell^i\right) . \tag{A.23b}$$

The j^{th} eigenvalue of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{P}}$ are related by

$$\mu_j = 1 - \epsilon\lambda_j . \tag{A.24}$$

A.5 Computational Cost

In [41], Assimakis et al. compared the Kalman Filter (KF) and Information Filter (IF) to determine the fastest filter for the time-varying and time-invariant forms of a linear SSM. Here, we expand this analysis to the Rauch-Tung-Striebel Smoother (RTSS) and Information Weighted Consensus Filter (IWCF). This enables a comparison of the algorithms as a function of the number of measurements T, the length of the state vectors n, and the length of the measurement vector m. Table A.1 summarizes the computational cost of matrix operations given the matrix dimensions and whether any matrices have known properties like the diagonals of the identity matrix $\mathbf{I}$ or a symmetric matrix $\mathbf{S}$. In this table, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are fully-populated matrices.

In terms of T, the difference between the batch and recursive implementations is that the batch algorithms have a cubic computational complexity $\mathcal{O}(T^3)$ while the recursive algorithms have a linear computation complexity $\mathcal{O}(T)$. Although the cubic complexity can be very limiting for batch computations with a large T, an analysis of the computational cost should still consider n and m for a given application. For some adaptive algorithms, the RTSS is implemented within the training phase. However, the parameter optimization in this dissertation only requires the RTSS for the

prediction phase. Consequently, the computational cost of the RTSS is not particularly important for the two algorithms presented as the main contribution of this dissertation.

Table A.1. Computation cost of matrix operations.

Matrix Operation	**Matrix Dimensions**	**Computational Cost**
$\mathbf{A} + \mathbf{B} = \mathbf{C}$	$(n \times m) + (n \times m)$	nm
$\mathbf{A} + \mathbf{B} = \mathbf{S}$	$(n \times n) + (n \times n)$	$\frac{1}{2}n^2 + \frac{1}{2}n$
$\mathbf{I} + \mathbf{A} = \mathbf{B}$	$(n \times n) + (n \times n)$	n
$\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$	$(n \times m) \cdot (m \times \ell)$	$2nm\ell - n\ell$
$\mathbf{A} \cdot \mathbf{B} = \mathbf{S}$	$(n \times m) \cdot (m \times n)$	$n^2m + nm - \frac{1}{2}n^2 - \frac{1}{2}n$
$\mathbf{A}^{-1} = \mathbf{B}$	$(n \times n)$	$\frac{1}{6}(16n^3 - 3n^2 - n)$

Computational Cost of Kalman Filter

Table A.2. Computational cost of the Kalman Filter.

Matrix Operation	**Matrix Dimensions**	**Computational Cost**
$\mathbf{H}_k\bar{\mathbf{P}}_k$	$(m \times n) \cdot (n \times n)$	$2n^2m - nm$
$\mathbf{H}_k\bar{\mathbf{P}}_k\mathbf{H}_k^\top$	$(m \times n) \cdot (n \times m)$	$nm^2 + nm - \frac{1}{2}m^2 - \frac{1}{2}m$
$\mathbf{S}_k = \mathbf{H}_k\bar{\mathbf{P}}_k\mathbf{H}_k^\top + \mathbf{R}_k$	$(m \times m) + (m \times m)$	$\frac{1}{2}m^2 + \frac{1}{2}m$
$\mathbf{S}_k^{-1}$	$(m \times m)$	$\frac{1}{6}(16m^3 - 3m^2 - m)$
$\mathbf{K}_k = \bar{\mathbf{P}}_k\mathbf{H}_k^\top\mathbf{S}_k^{-1}$	$(n \times m) \cdot (m \times m)$	$2nm^2 - nm$
$\mathbf{K}_k\mathbf{H}_k$	$(n \times m) \cdot (m \times n)$	$2n^2m - n^2$
$\mathbf{I} - \mathbf{K}_k\mathbf{H}_k$	$(n \times n) + (n \times n)$	n
$[\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\bar{\mathbf{x}}_k$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\mathbf{K}_k\mathbf{y}_k$	$(n \times m) \cdot (m \times 1)$	$2nm - n$
$\widetilde{\mathbf{x}}_k = \mathbf{K}_k\mathbf{y}_k + [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\bar{\mathbf{x}}_k$	$(n \times 1) + (n \times 1)$	n
$\widetilde{\mathbf{P}}_k = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\bar{\mathbf{P}}_k$	$(n \times n) \cdot (n \times n)$	$n^3 + \frac{1}{2}n^2 - \frac{1}{2}n$
$\bar{\mathbf{x}}_{k+1} = \mathbf{A}_k\widetilde{\mathbf{x}}_k$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\mathbf{A}_k\widetilde{\mathbf{P}}_k$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\mathbf{A}_k\widetilde{\mathbf{P}}_k\mathbf{A}_k^\top$	$(n \times n) \cdot (n \times n)$	$n^3 + \frac{1}{2}n^2 - \frac{1}{2}n$
$\bar{\mathbf{P}}_{k+1} = \mathbf{Q}_k + \mathbf{A}_k\widetilde{\mathbf{P}}_k\mathbf{A}_k^\top$	$(n \times n) + (n \times n)$	$\frac{1}{2}n^2 + \frac{1}{2}n$
Step-wise computational cost: $4n^3 + \frac{7}{2}n^2 - \frac{3}{2}n + 4n^2m + nm + 3nm^2 + \frac{1}{6}(16m^3 - 3m^2 - m)$		
Total computational complexity: $\mathcal{O}(4n^3T)$ if $n \gg m$		

Computational Cost of Information Filter

Table A.3. Computational cost of the Information Filter.

Matrix Operation	**Matrix Dimensions**	**Computational Cost**
$\mathbf{R}_k^{-1}$	$(m \times m)$	$\frac{1}{6}(16m^3 - 3m^2 - m)$
$\mathbf{H}_k^\top \mathbf{R}_k^{-1}$	$(n \times m) \cdot (m \times m)$	$2nm^2 - nm$
$\mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{y}_k$	$(n \times m) \cdot (m \times 1)$	$2nm - n$
$\widetilde{\mathbf{z}}_k = \bar{\mathbf{z}}_k + \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{y}_k$	$(n \times 1) + (n \times 1)$	n
$\mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{H}_k$	$(n \times n) \cdot (n \times n)$	$n^2 m + nm - \frac{1}{2}n^2 - \frac{1}{2}n$
$\widetilde{\mathbf{Z}}_k = \bar{\mathbf{Z}}_k + \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{H}_k$	$(n \times n) + (n \times n)$	$\frac{1}{2}n^2 + \frac{1}{2}n$
$\widetilde{\mathbf{P}}_k = \widetilde{\mathbf{Z}}_k^{-1}$	$(n \times n)$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\widetilde{\mathbf{x}}_k = \widetilde{\mathbf{Z}}_k^{-1} \widetilde{\mathbf{z}}_k$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\mathbf{K}_k = \widetilde{\mathbf{Z}}_k^{-1} \mathbf{H}_k^\top \mathbf{R}_k^{-1}$	$(n \times n) \cdot (n \times m)$	$2n^2 m - nm$
$\mathbf{A}_k \widetilde{\mathbf{Z}}_k^{-1}$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\mathbf{A}_k \widetilde{\mathbf{Z}}_k^{-1} \mathbf{A}_k^\top$	$(n \times n) \cdot (n \times n)$	$n^3 + \frac{1}{2}n^2 - \frac{1}{2}n$
$\bar{\mathbf{P}}_{k+1} = \mathbf{Q}_k + \mathbf{A}_k \widetilde{\mathbf{Z}}_k^{-1} \mathbf{A}_k^\top$	$(n \times n) + (n \times n)$	$\frac{1}{2}n^2 + \frac{1}{2}n$
$\bar{\mathbf{Z}}_{k+1} = \bar{\mathbf{P}}_{k+1}^{-1}$	$(n \times n)$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\mathbf{A}_k \widetilde{\mathbf{Z}}_k^{-1} \widetilde{\mathbf{z}}_k$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\bar{\mathbf{z}}_{k+1} = \bar{\mathbf{Z}}_{k+1} \mathbf{A}_k \widetilde{\mathbf{Z}}_k^{-1} \widetilde{\mathbf{z}}_k$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{P}}_{k+1} \bar{\mathbf{z}}_{k+1}$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
Step-wise computational cost: $\frac{1}{6}(50n^3 + \frac{89}{2}n^2 - \frac{47}{2}n) + 3n^2 m + nm + 2nm^2 + \frac{1}{6}(16m^3 - 3m^2 - m)$		
Total computational complexity: $\mathcal{O}(\frac{25}{3}n^3 T)$ if $n \gg m$		

Computational Cost of Rauch-Tung-Striebel Smoother

Table A.4. Computational cost of the Rauch-Tung-Striebel Smoother.

Matrix Operation	**Matrix Dimensions**	**Computational Cost**
$\bar{\mathbf{x}}_{k+1} = \mathbf{A}_k \widetilde{\mathbf{x}}_k$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\widetilde{\mathbf{P}}_k \mathbf{A}_k^\top$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\mathbf{A}_k \widetilde{\mathbf{P}}_k \mathbf{A}_k^\top$	$(n \times n) \cdot (n \times n)$	$n^3 + \frac{1}{2}n^2 - \frac{1}{2}n$
$\bar{\mathbf{P}}_{k+1} = \mathbf{A}_k \widetilde{\mathbf{P}}_k \mathbf{A}_k^\top + \mathbf{Q}_k$	$(n \times n) + (n \times n)$	$\frac{1}{2}n^2 + \frac{1}{2}n$
$\mathbf{G}_k = \widetilde{\mathbf{P}}_k \mathbf{A}_k^\top [\bar{\mathbf{P}}_{k+1}]^{-1}$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\widehat{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}_{k+1}$	$(n \times 1) + (n \times 1)$	n
$\mathbf{G}_k[\widehat{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}_{k+1}]$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\widehat{\mathbf{x}}_k = \widetilde{\mathbf{x}}_k + \mathbf{G}_k[\widehat{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}_{k+1}]$	$(n \times 1) + (n \times 1)$	n
$\widehat{\mathbf{P}}_{k+1} - \bar{\mathbf{P}}_{k+1}$	$(n \times n) + (n \times n)$	$\frac{1}{2}n^2 + \frac{1}{2}n$
$\mathbf{G}_k[\widehat{\mathbf{P}}_{k+1} - \bar{\mathbf{P}}_{k+1}]$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\mathbf{G}_k[\widehat{\mathbf{P}}_{k+1} - \bar{\mathbf{P}}_{k+1}]\mathbf{G}_k^\top$	$(n \times n) \cdot (n \times n)$	$n^3 + \frac{1}{2}n^2 - \frac{1}{2}n$
$\widehat{\mathbf{P}}_k = \widetilde{\mathbf{P}}_k + \mathbf{G}_k[\widehat{\mathbf{P}}_{k+1} - \bar{\mathbf{P}}_{k+1}]\mathbf{G}_k^\top$	$(n \times n) + (n \times n)$	$\frac{1}{2}n^2 + \frac{1}{2}n$
Step-wise computational cost: $8n^3 + \frac{7}{2}n^2$		
Total computational complexity: $\mathcal{O}(8n^3T)$ if $n \gg m$		

Computational Cost of Information Weighted Consensus Filter

Table A.5. Computational cost of the Information Weighted Consensus Filter.

Matrix Operation	**Matrix Dimensions**	**Computational Cost**
$\mathbf{R}_k^{-1}$	$(m \times m)$	$\frac{1}{6}(16m^3 - 3m^2 - m)$
$\mathbf{H}_k^\top \mathbf{R}_k^{-1}$	$(n \times m) \cdot (m \times m)$	$2nm^2 - nm$
$\mathbf{i}_k^i = \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{y}_k^i$	$(n \times m) \cdot (m \times 1)$	$2nm - n$
$\mathbf{w}_{k,0}^i = \bar{\mathbf{z}}_k^i + \mathbf{i}_k^i$	$(n \times 1) + (n \times 1)$	n
$\mathbf{w}_{k,L}^i$		$2L(N_i + 1)n$
$\mathbf{I}_k^i = \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{H}_k$	$(n \times n) \cdot (n \times n)$	$n^2m + nm - \frac{1}{2}n^2 - \frac{1}{2}n$
$\mathbf{W}_{k,0}^i = \bar{\mathbf{Z}}_k^i + \mathbf{I}_k^i$	$(n \times n) + (n \times n)$	$\frac{1}{2}n^2 + \frac{1}{2}n$
$\mathbf{W}_{k,L}^i$		$2L(N_i + 1)n^2$
$\left[\mathbf{W}_{k,L}^i\right]^{-1}$	$(n \times n)$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\widetilde{\mathbf{x}}_k^i = \left[\mathbf{W}_{k,L}^i\right]^{-1} \mathbf{w}_{k,L}^i$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\widetilde{\mathbf{Z}}_k^i = N\mathbf{W}_{k,L}^i$		n^2
$\widetilde{\mathbf{P}}_k^i = \left[\widetilde{\mathbf{Z}}_k^i\right]^{-1}$	$(n \times n)$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\mathbf{A}_k \widetilde{\mathbf{P}}_k^i$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\mathbf{A}_k \widetilde{\mathbf{P}}_k^i \mathbf{A}_k^\top$	$(n \times n) \cdot (n \times n)$	$n^3 + \frac{1}{2}n^2 - \frac{1}{2}n$
$\bar{\mathbf{P}}_{k+1}^i = \mathbf{Q}_k + \mathbf{A}_k \widetilde{\mathbf{P}}_k^i \mathbf{A}_k^\top$	$(n \times n) + (n \times n)$	$\frac{1}{2}n^2 + \frac{1}{2}n$
$\bar{\mathbf{Z}}_{k+1}^i = \left[\bar{\mathbf{P}}_{k+1}^i\right]^{-1}$	$(n \times n)$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\bar{\mathbf{x}}_{k+1} = \mathbf{A}_k \widetilde{\mathbf{x}}_k$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
Step-wise computational cost: 111		
Total computational cost: $\mathcal{O}(\frac{25}{3}n^3T)$ if $n \gg m$		

B. PSEUDOCODE

B.1 Pseudocode for Chapter 2

Algorithm 1 Kalman Filter.

1: Calculate *a posteriori*

$$\mathbf{v}_k = \mathbf{y}_k - \mathbf{H}_k \bar{\mathbf{x}}_k \tag{B.1}$$

$$\mathbf{S}_k = \mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^\top + \mathbf{R}_k \tag{B.2}$$

$$\mathbf{K}_k = \bar{\mathbf{P}}_k \mathbf{H}_k^\top \mathbf{S}_k^{-1} \tag{B.3}$$

$$\widetilde{\mathbf{x}}_k = \bar{\mathbf{x}}_k + \mathbf{K}_k \mathbf{v}_k \tag{B.4}$$

$$\widetilde{\mathbf{P}}_k = \bar{\mathbf{P}}_k - \mathbf{K}_k \mathbf{H}_k \bar{\mathbf{P}}_k \tag{B.5}$$

2: Calculate *a priori*

$$\bar{\mathbf{x}}_k = \mathbf{A}_{k-1} \widetilde{\mathbf{x}}_{k-1} \tag{B.6}$$

$$\bar{\mathbf{P}}_k = \mathbf{A}_{k-1} \widetilde{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^\top + \mathbf{Q}_{k-1} \tag{B.7}$$

Algorithm 2 Information Filter.

1: Calculate local information

$$\mathbf{i}_k = \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{y}_k \tag{B.8}$$

$$\mathbf{I}_k = \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{H}_k \tag{B.9}$$

2: Calculate *a posteriori*

$$\widetilde{\mathbf{z}}_k = \bar{\mathbf{z}}_k + \mathbf{i}_k \tag{B.10}$$

$$\widetilde{\mathbf{Z}}_k = \bar{\mathbf{Z}}_k + \mathbf{I}_k \tag{B.11}$$

3: Calculate *a priori*

$$\mathbf{M}_k = \mathbf{A}_{k-1}^{-\top} \widetilde{\mathbf{Z}}_{k-1} \mathbf{A}_{k-1}^{-1} \tag{B.12}$$

$$\mathbf{\Sigma}_k = \mathbf{M}_k + \mathbf{Q}_{k-1}^{-1} \tag{B.13}$$

$$\bar{\mathbf{Z}}_k = \mathbf{M}_k - \mathbf{M}_k \mathbf{\Sigma}_k^{-1} \mathbf{M}_k \tag{B.14}$$

$$\bar{\mathbf{z}}_k = \bar{\mathbf{Z}}_k \mathbf{A}_{k-1} \widetilde{\mathbf{Z}}_{k-1}^{-1} \widetilde{\mathbf{z}}_{k-1} \tag{B.15}$$

Algorithm 3 Rauch-Tung-Striebel Smoother.

1: Calculate *a priori*

$$\bar{\mathbf{x}}_{k+1} = \mathbf{A}_k \widetilde{\mathbf{x}}_k \tag{B.16}$$

$$\bar{\mathbf{P}}_{k+1} = \mathbf{A}_k \widetilde{\mathbf{P}}_k \mathbf{A}_k^\top + \mathbf{Q}_k \tag{B.17}$$

2: Calculate smoothed

$$\mathbf{G}_k = \widetilde{\mathbf{P}}_k \mathbf{A}_k^\top \bar{\mathbf{P}}_{k+1}^{-1} \tag{B.18}$$

$$\widehat{\mathbf{x}}_k = \widetilde{\mathbf{x}}_k + \mathbf{G}_k \left[\widehat{\mathbf{m}}_{k+1} - \bar{\mathbf{x}}_{k+1}\right] \tag{B.19}$$

$$\widehat{\mathbf{P}}_k = \widetilde{\mathbf{P}}_k + \mathbf{G}_k \left[\widehat{\mathbf{P}}_{k+1} - \bar{\mathbf{P}}_{k+1}\right] \mathbf{G}_k^\top \tag{B.20}$$

B.2 Pseudocode for Chapter 3

The variables are written without the parameters $\boldsymbol{\theta}$ for simplicity.

Algorithm 4 Average consensus filter.

1: Send message $\mathcal{M}_{\ell-1}^{i}$ to C_i^n
2: Receive messages $\mathcal{M}_{\ell-1}^{j}$ to $j \in C_i^n$
3: $\mathcal{M}_{\ell}^{i} = \mathcal{M}_{\ell-1}^{i} + \epsilon \sum_{j \in C_i^n} \left[\mathcal{M}_{\ell-1}^{j} - \mathcal{M}_{\ell-1}^{i} \right]$

Algorithm 5 Kalman Filter with energy function.

1: Calculate *a posteriori*

$$\mathbf{v}_k = \mathbf{y}_k - \mathbf{H}_k \bar{\mathbf{x}}_k \tag{B.21}$$

$$\mathbf{S}_k = \mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^{\top} + \mathbf{R}_k \tag{B.22}$$

$$\mathbf{K}_k = \bar{\mathbf{P}}_k \mathbf{H}_k^{\top} \mathbf{S}_k^{-1} \tag{B.23}$$

$$\widetilde{\mathbf{x}}_k = \bar{\mathbf{x}}_k + \mathbf{K}_k \mathbf{v}_k \tag{B.24}$$

$$\widetilde{\mathbf{P}}_k = \bar{\mathbf{P}}_k - \mathbf{K}_k \mathbf{H}_k \bar{\mathbf{P}}_k \tag{B.25}$$

2: Energy function

$$\varphi_k = \varphi_{k-1} + \frac{1}{2} \left[\log|2\pi \mathbf{S}_k| + \mathbf{v}_k^{\top} \mathbf{S}_k^{-1} \mathbf{v}_k \right] \tag{B.26}$$

3: Calculate *a priori*

$$\bar{\mathbf{x}}_k = \mathbf{A}_{k-1} \widetilde{\mathbf{x}}_{k-1} \tag{B.27}$$

$$\bar{\mathbf{P}}_k = \mathbf{A}_{k-1} \widetilde{\mathbf{P}}_{k-1} \mathbf{A}_{k-1}^{\top} + \mathbf{Q}_{k-1} \tag{B.28}$$

Algorithm 6 Derivatives of the Kalman Filter with sensitivity equations.

1: Calculate *a posteriori* derivatives

$$\frac{\partial \mathbf{v}_k}{\partial \theta_j} = -\frac{\partial \mathbf{H}_k}{\partial \theta_j}\bar{\mathbf{x}}_k - \mathbf{H}_k\frac{\partial \bar{\mathbf{x}}_k}{\partial \theta_j} \tag{B.29}$$

$$\frac{\partial \mathbf{S}_k}{\partial \theta_j} = \frac{\partial \mathbf{H}_k}{\partial \theta_j}\bar{\mathbf{P}}_k\mathbf{H}_k^\top + \mathbf{H}_k\frac{\partial \bar{\mathbf{P}}_k}{\partial \theta_j}\mathbf{H}_k^\top + \mathbf{H}_k\bar{\mathbf{P}}_k\frac{\partial \mathbf{H}_k^\top}{\partial \theta_j} + \frac{\partial \mathbf{R}_k}{\partial \theta_j} \tag{B.30}$$

$$\frac{\partial \mathbf{K}_k}{\partial \theta_j} = \frac{\partial \bar{\mathbf{P}}_k}{\partial \theta_j}\mathbf{H}_k^\top\mathbf{S}_k^{-1} + \mathbf{P}_k^-\frac{\partial \mathbf{H}_k^\top}{\partial \theta_j}\mathbf{S}_k^{-1} - \bar{\mathbf{P}}_k\mathbf{H}_k^\top\mathbf{S}_k^{-1}\frac{\partial \mathbf{S}_k}{\partial \theta_j}\mathbf{S}_k^{-1} \tag{B.31}$$

$$\frac{\partial \widetilde{\mathbf{x}}_k}{\partial \theta_j} = \frac{\partial \bar{\mathbf{x}}_k}{\partial \theta_j} + \frac{\partial \mathbf{K}_k}{\partial \theta_j}\mathbf{v}_k + \mathbf{K}_k\frac{\partial \mathbf{v}_k}{\partial \theta_j} \tag{B.32}$$

$$\frac{\partial \widetilde{\mathbf{P}}_k}{\partial \theta_j} = \frac{\partial \bar{\mathbf{P}}_k}{\partial \theta_j} - \frac{\partial \mathbf{K}_k}{\partial \theta_j}\mathbf{H}_k\bar{\mathbf{P}}_k - \mathbf{K}_k\frac{\partial \mathbf{H}_k}{\partial \theta_j}\bar{\mathbf{P}}_k - \mathbf{K}_k\mathbf{H}_k\frac{\partial \bar{\mathbf{P}}_k}{\partial \theta_j} \tag{B.33}$$

2: Sensitivity equations

$$\frac{\partial \varphi_k}{\partial \theta_j} = \frac{\partial \varphi_{k-1}}{\partial \theta_j} + \frac{1}{2}\mathrm{Tr}\left(\mathbf{S}_k^{-1}\frac{\partial \mathbf{S}_k}{\partial \theta_j}\right) + \mathbf{v}_k^\top\mathbf{S}_k^{-1}\frac{\partial \mathbf{v}_k}{\partial \theta_j} - \frac{1}{2}\mathbf{v}_k^\top\mathbf{S}_k^{-1}\frac{\partial \mathbf{S}_k^{-1}}{\partial \theta_j}\mathbf{S}_k^{-1}\mathbf{v}_k \tag{B.34}$$

3: Calculate *a priori* derivatives

$$\frac{\partial \bar{\mathbf{x}}_k}{\partial \theta_j} = \frac{\partial \mathbf{A}_{k-1}}{\partial \theta_j}\widetilde{\mathbf{x}}_{k-1} + \mathbf{A}_{k-1}\frac{\partial \widetilde{\mathbf{x}}_{k-1}}{\partial \theta_j} \tag{B.35}$$

$$\frac{\partial \bar{\mathbf{P}}_k}{\partial \theta_j} = \frac{\partial \mathbf{A}_{k-1}}{\partial \theta_j}\widetilde{\mathbf{P}}_{k-1}\mathbf{A}_{k-1}^\top + \mathbf{A}_{k-1}\frac{\partial \widetilde{\mathbf{P}}_{k-1}}{\partial \theta_j}\mathbf{A}_{k-1}^\top + \mathbf{A}_{k-1}\widetilde{\mathbf{P}}_{k-1}\frac{\partial \mathbf{A}_{k-1}^\top}{\partial \theta_j} + \frac{\partial \mathbf{Q}_{k-1}}{\partial \theta_j} \tag{B.36}$$

Algorithm 7 Information Weighted Consensus Filter with energy function.

1: Energy function

$$\mathbf{v}_k = \mathbf{y}_k - \mathbf{H}_k \bar{\mathbf{x}}_k \tag{B.37}$$

$$\mathbf{S}_k = \mathbf{H}_k \bar{\mathbf{Z}}_k^{-1} \mathbf{H}_k^\top + \mathbf{R}_k \tag{B.38}$$

$$\varphi_k = \varphi_{k-1} + \frac{1}{2}\left[\log|2\pi\mathbf{S}_k| + \mathbf{v}_k^\top \mathbf{S}_k^{-1} \mathbf{v}_k\right] \tag{B.39}$$

2: Calculate local information

$$\mathbf{i}_k = \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{y}_k \tag{B.40}$$

$$\mathbf{I}_k = \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{H}_k \tag{B.41}$$

3: Consensus initialization

$$\mathbf{w}_{k,0} = \frac{1}{N} \bar{\mathbf{z}}_k + \mathbf{i}_k \tag{B.42}$$

$$\mathbf{W}_{k,0} = \frac{1}{N} \bar{\mathbf{Z}}_k + \mathbf{I}_k \tag{B.43}$$

4: Consensus loop

$$\mathcal{C}\{\, \mathbf{w}_{k,\ell}\,;\, \mathbf{W}_{k,\ell} \,\} \text{ with } \epsilon \tag{B.44}$$

5: Calculate *a posteriori*

$$\widetilde{\mathbf{x}}_k = \mathbf{W}_{k,L}^{-1} \mathbf{w}_{k,L} \tag{B.45}$$

$$\widetilde{\mathbf{Z}}_k = N \mathbf{W}_{k,L} \tag{B.46}$$

6: Calculate *a priori*

$$\bar{\mathbf{x}}_k = \mathbf{A}_{k-1} \widetilde{\mathbf{x}}_{k-1} \tag{B.47}$$

$$\bar{\mathbf{Z}}_k = \left(\mathbf{A}_{k-1} \widetilde{\mathbf{Z}}_{k-1}^{-1} \mathbf{A}_{k-1}^\top + \mathbf{Q}_{k-1}\right)^{-1} \tag{B.48}$$

Algorithm 8 Derivatives of the Information Weighted Consensus Filter with sensitivity equations.

1: Energy function

$$\frac{\partial \mathbf{v}_k}{\partial \theta_j} = -\frac{\partial \mathbf{H}_k}{\partial \theta_j}\bar{\mathbf{x}}_k - \mathbf{H}_k \frac{\partial \bar{\mathbf{x}}_k}{\partial \theta_j} \tag{B.49}$$

$$\frac{\partial \mathbf{S}_k}{\partial \theta_j} = \frac{\partial \mathbf{H}_k}{\partial \theta_j}\bar{\mathbf{Z}}_k^{-1}\mathbf{H}_k^\top - \mathbf{H}_k \left[\bar{\mathbf{Z}}_k\right]^{-1} \frac{\partial \bar{\mathbf{Z}}_k}{\partial \theta_j} \left[\bar{\mathbf{Z}}_k\right]^{-1} \mathbf{H}_k^\top + \mathbf{H}_k \bar{\mathbf{Z}}_k^{-1} \frac{\partial \mathbf{H}_k^\top}{\partial \theta_j} + \frac{\partial \mathbf{R}_k}{\partial \theta_j} \tag{B.50}$$

$$\frac{\partial \varphi_k}{\partial \theta_j} = \frac{\partial \varphi_{k-1}}{\partial \theta_j} + \frac{1}{2}\mathrm{Tr}\left(\mathbf{S}_k^{-1}\frac{\partial \mathbf{S}_k}{\partial \theta_j}\right) + \mathbf{v}_k^\top \mathbf{S}_k^{-1}\frac{\partial \mathbf{v}_k}{\partial \theta_j} - \frac{1}{2}\mathbf{v}_k^\top \mathbf{S}_k^{-1}\frac{\partial \mathbf{S}_k^{-1}}{\partial \theta_j}\mathbf{S}_k^{-1}\mathbf{v}_k \tag{B.51}$$

2: Calculate local information

$$\frac{\partial \mathbf{i}_k}{\partial \theta_j} = -\mathbf{H}_k^\top \left[\mathbf{R}_k\right]^{-1} \frac{\partial \mathbf{R}_k}{\partial \theta_j} \left[\mathbf{R}_k\right]^{-1} \mathbf{y}_k \tag{B.52}$$

$$\frac{\partial \mathbf{I}_k}{\partial \theta_j} = -\mathbf{H}_k^\top \left[\mathbf{R}_k\right]^{-1} \frac{\partial \mathbf{R}_k}{\partial \theta_j} \left[\mathbf{R}_k\right]^{-1} \mathbf{H}_k \tag{B.53}$$

3: Consensus initialization

$$\frac{\partial \mathbf{w}_{k,0}}{\partial \theta_j} = \frac{1}{N}\frac{\partial \bar{\mathbf{z}}_k}{\partial \theta_j} + \frac{\partial \mathbf{i}_k}{\partial \theta_j} \tag{B.54}$$

$$\frac{\partial \mathbf{W}_{k,0}}{\partial \theta_j} = \frac{1}{N}\frac{\partial \bar{\mathbf{Z}}_k}{\partial \theta_j} + \frac{\partial \mathbf{I}_k}{\partial \theta_j} \tag{B.55}$$

4: Consensus loop

$$\mathcal{C}\left\{ \frac{\partial \mathbf{w}_{k,\ell}}{\partial \theta_j} ; \frac{\partial \mathbf{W}_{k,\ell}}{\partial \theta_j} \right\} \text{ with } \epsilon \tag{B.56}$$

5: Calculate *a posteriori* derivatives

$$\frac{\partial \widetilde{\mathbf{x}}_k}{\partial \theta_j} = -\left[\mathbf{W}_{k,L}\right]^{-1} \frac{\partial \mathbf{W}_{k,L}}{\partial \theta_j} \left[\mathbf{W}_{k,L}\right]^{-1} \mathbf{w}_{k,L} + \left[\mathbf{W}_{k,L}\right]^{-1} \frac{\partial \mathbf{w}_{k,L}}{\partial \theta_j} \tag{B.57}$$

$$\frac{\partial \widetilde{\mathbf{Z}}_k}{\partial \theta_j} = N\frac{\partial \mathbf{W}_{k,L}}{\partial \theta_j} \tag{B.58}$$

6: Calculate *a priori* derivatives

$$\frac{\partial \bar{\mathbf{x}}_k}{\partial \theta_j} = \frac{\partial \mathbf{A}_{k-1}}{\partial \theta_j}\widetilde{\mathbf{x}}_{k-1} + \mathbf{A}_{k-1}\frac{\partial \widetilde{\mathbf{x}}_{k-1}}{\partial \theta_j} \tag{B.59}$$

$$\begin{aligned}\frac{\partial \bar{\mathbf{Z}}_k}{\partial \theta_j} = &\left(\mathbf{A}_{k-1}\left[\widetilde{\mathbf{Z}}_{k-1}\right]^{-1}\mathbf{A}_{k-1}^\top + \mathbf{Q}_{k-1}\right)^{-1}\left(\frac{\partial \mathbf{A}_{k-1}}{\partial \theta_j}\left[\widetilde{\mathbf{Z}}_{k-1}\right]^{-1}\mathbf{A}_{k-1}^\top\right.\\ &+ \mathbf{A}_{k-1}\left[\widetilde{\mathbf{Z}}_{k-1}\right]^{-1}\frac{\partial \widetilde{\mathbf{Z}}_{k-1}}{\partial \theta_j}\left[\widetilde{\mathbf{Z}}_{k-1}\right]^{-1}\mathbf{A}_{k-1}^\top + \mathbf{A}_{k-1}\left[\widetilde{\mathbf{Z}}_{k-1}\right]^{-1}\frac{\partial \mathbf{A}_{k-1}^\top}{\partial \theta_j}\\ &\left.+\frac{\partial \mathbf{Q}_{k-1}}{\partial \theta_j}\right)\left(\mathbf{A}_{k-1}\left[\widetilde{\mathbf{Z}}_{k-1}\right]^{-1}\mathbf{A}_{k-1}^\top + \mathbf{Q}_{k-1}\right)^{-1}\end{aligned} \tag{B.60}$$

C. THE INFORMATION FORM

Consider the linear state-space model

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{q}_k \,, \qquad \mathbf{q}_k \sim \mathcal{N}\big(\mathbf{0}, \mathbf{Q}_k\big) \,, \tag{C.1a}$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{r}_k \,, \qquad \mathbf{r}_k \sim \mathcal{N}\big(\mathbf{0}, \mathbf{R}_k\big) \,. \tag{C.1b}$$

Table C.1 summarizes the relationship between the Kalman Filter and Information Filter.

Table C.1. Summary of Kalman Filter and Information Filter equations.

	Kalman Filter	**Transformation**	**Information Filter**
Prediction	$\bar{\mathbf{P}}_k = \mathbf{A}_{k-1}\widetilde{\mathbf{P}}_{k-1}\mathbf{A}_{k-1}^\top + \mathbf{Q}_{k-1}$ $\bar{\mathbf{x}}_k = \mathbf{A}_{k-1}\widetilde{\mathbf{x}}_{k-1}$	$\bar{\mathbf{z}}_k = \big(\bar{\mathbf{P}}_k\big)^{-1}$ $\Longleftrightarrow$ $\bar{\mathbf{z}}_k = \big(\bar{\mathbf{P}}_k\big)^{-1}\bar{\mathbf{x}}_k$	$\mathbf{M}_k = \big[\mathbf{A}_{k-1}^\top\big]^{-1}\widetilde{\mathbf{Z}}_{k-1}\mathbf{A}_{k-1}^{-1}$ $\mathbf{\Sigma}_k = \mathbf{M}_k + \mathbf{Q}_{k-1}^{-1}$ $\bar{\mathbf{Z}}_k = \mathbf{M}_k - \mathbf{M}_k\left[\mathbf{\Sigma}_k\right]^{-1}\mathbf{M}_k$ $\bar{\mathbf{z}}_k = \bar{\mathbf{Z}}_k\mathbf{A}_{k-1}\big[\widetilde{\mathbf{Z}}_{k-1}\big]^{-1}\widetilde{\mathbf{z}}_{k-1}$
Update	$\mathbf{S}_k = \mathbf{H}_k\bar{\mathbf{P}}_k\mathbf{H}_k^\top + \mathbf{R}_k$ $\mathbf{K}_k = \bar{\mathbf{P}}_k\mathbf{H}_k^\top\mathbf{S}_k^{-1}$ $\widetilde{\mathbf{P}}_k = \bar{\mathbf{P}}_k - \mathbf{K}_k\mathbf{H}_k\bar{\mathbf{P}}_k$ $\widetilde{\mathbf{x}}_k = \bar{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{H}_k\bar{\mathbf{x}}_k\right)$	$\widetilde{\mathbf{z}}_k = \big(\widetilde{\mathbf{P}}_k\big)^{-1}$ $\Longleftrightarrow$ $\widetilde{\mathbf{z}}_k = \big(\widetilde{\mathbf{P}}_k\big)^{-1}\widetilde{\mathbf{x}}_k$	$\mathbf{i}_k = \mathbf{H}_k^\top\mathbf{R}_k^{-1}\mathbf{y}$ $\mathbf{I}_k = \mathbf{H}_k^\top\mathbf{R}_k^{-1}\mathbf{H}_k$ $\widetilde{\mathbf{z}}_k = \bar{\mathbf{z}}_k + \mathbf{i}_k$ $\widetilde{\mathbf{Z}}_k = \bar{\mathbf{Z}}_k + \mathbf{I}_k$

C.1 A Posteriori Information Vector

Transforming the state vector into the information vector starts with matrix manipulation of the Kalman gain:

$$\mathbf{K}_k = \bar{\mathbf{P}}_k \mathbf{H}_k^\top \left(\mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1} \tag{C.2a}$$

$$= \widetilde{\mathbf{P}}_k \widetilde{\mathbf{P}}_k^{-1} \bar{\mathbf{P}}_k \mathbf{H}_k^\top \left(\mathbf{R}_k \left[\mathbf{R}_k^{-1} \mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^\top + \mathbf{I}\right]\right)^{-1} \tag{C.2b}$$

$$= \widetilde{\mathbf{P}}_k \left(\bar{\mathbf{P}}_k^{-1} + \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{H}_k\right) \bar{\mathbf{P}}_k \mathbf{H}_k^\top \left(\mathbf{R}_k^{-1} \mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^\top + \mathbf{I}\right)^{-1} \mathbf{R}_k^{-1} \tag{C.2c}$$

$$= \widetilde{\mathbf{P}}_k \mathbf{H}_k^\top \left(\mathbf{I} + \mathbf{R}_k^{-1} \mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^\top\right) \left(\mathbf{R}_k^{-1} \mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^\top + \mathbf{I}\right)^{-1} \mathbf{R}_k^{-1} \tag{C.2d}$$

$$= \widetilde{\mathbf{P}}_k \mathbf{H}_k^\top \mathbf{R}_k^{-1} \,. \tag{C.2e}$$

Plug the Kalman gain into the state update and manipulate the equation such that

$$\widetilde{\mathbf{x}}_k = \bar{\mathbf{x}}_k + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{H}_k \bar{\mathbf{x}}_k\right) \tag{C.3a}$$

$$= \bar{\mathbf{x}}_k + \widetilde{\mathbf{P}}_k \mathbf{H}_k^\top \mathbf{R}_k^{-1} \left(\mathbf{y}_k - \mathbf{H}_k \bar{\mathbf{x}}_k\right) \tag{C.3b}$$

$$= \bar{\mathbf{x}}_k + \widetilde{\mathbf{P}}_k \left(\mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{y}_k - \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{H}_k \bar{\mathbf{x}}_k\right) \tag{C.3c}$$

$$= \widetilde{\mathbf{P}}_k \left(\left[\widetilde{\mathbf{P}}_k\right]^{-1} - \mathbf{H}_k^\top \mathbf{R}^{-1} \mathbf{H}_k\right) \bar{\mathbf{x}}_k + \widetilde{\mathbf{P}}_k \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{y}_k \tag{C.3d}$$

$$= \widetilde{\mathbf{P}}_k \left[\bar{\mathbf{P}}_k\right]^{-1} \bar{\mathbf{x}}_k + \widetilde{\mathbf{P}}_k \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{y}_k \,. \tag{C.3e}$$

Multiplying each side by the inverse covariance matrix yields the information vector update

$$\widetilde{\mathbf{z}}_k = \bar{\mathbf{z}}_k + \mathbf{i}_k \,, \tag{C.4}$$

where

$$\widetilde{\mathbf{z}}_k = \left[\widetilde{\mathbf{P}}_k\right]^{-1} \widetilde{\mathbf{x}}_k \,, \tag{C.5a}$$

$$\bar{\mathbf{z}}_k = \left[\bar{\mathbf{P}}_k\right]^{-1} \bar{\mathbf{x}}_k \,, \tag{C.5b}$$

$$\mathbf{i}_k = \mathbf{H}_k^\top \mathbf{R}_k^{-1} \mathbf{y}_k \,. \tag{C.5c}$$

C.2 A Posteriori Information Matrix

Transforming the covariance matrix into the information matrix depends upon the Woodbury matrix identity such that

$$\widetilde{\mathbf{P}}_k = \bar{\mathbf{P}}_k - \mathbf{K}_k\mathbf{H}_k\bar{\mathbf{P}}_k \tag{C.6a}$$

$$= \bar{\mathbf{P}}_k - \bar{\mathbf{P}}_k\mathbf{H}_k^\top\left(\mathbf{H}_k\bar{\mathbf{P}}_k\mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1}\mathbf{H}_k\bar{\mathbf{P}}_k \tag{C.6b}$$

$$= \left(\bar{\mathbf{P}}_k^{-1} + \mathbf{H}_k^\top\mathbf{R}_k^{-1}\mathbf{H}_k\right)^{-1}. \tag{C.6c}$$

Taking the inverse of each side yields the information matrix update

$$\widetilde{\mathbf{Z}}_k = \bar{\mathbf{Z}}_k + \mathbf{I}_k\,, \tag{C.7}$$

where

$$\widetilde{\mathbf{Z}}_k = \widetilde{\mathbf{P}}_k^{-1}\,, \tag{C.8a}$$

$$\bar{\mathbf{Z}}_k = \bar{\mathbf{P}}_k^{-1}\,, \tag{C.8b}$$

$$\mathbf{I}_k = \mathbf{H}_k^\top\mathbf{R}_k^{-1}\mathbf{H}_k\,. \tag{C.8c}$$

C.3 A Priori Information Vector

Transforming the state vector into the information vector is simply matrix manipulation and substitution:

$$\bar{\mathbf{x}}_k = \mathbf{A}_{k-1}\widetilde{\mathbf{x}}_{k-1} \tag{C.9a}$$

$$\bar{\mathbf{P}}_k\left[\bar{\mathbf{P}}_k\right]^{-1}\bar{\mathbf{x}}_k = \mathbf{A}_{k-1}\widetilde{\mathbf{P}}_{k-1}\left[\widetilde{\mathbf{P}}_{k-1}\right]^{-1}\widetilde{\mathbf{x}}_{k-1} \tag{C.9b}$$

$$\bar{\mathbf{P}}_k\bar{\mathbf{z}}_k = \mathbf{A}_{k-1}\widetilde{\mathbf{P}}_{k-1}\widetilde{\mathbf{z}}_{k-1} \tag{C.9c}$$

$$\bar{\mathbf{z}}_k = \bar{\mathbf{Z}}_k\mathbf{A}_{k-1}\left[\widetilde{\mathbf{z}}_{k-1}\right]^{-1}\widetilde{\mathbf{z}}_{k-1}\,. \tag{C.9d}$$

C.4 A Priori Information Matrix

Transforming the state matrix into the information matrix depends upon Hua's identity such that

$$\bar{\mathbf{P}}_k = \mathbf{A}_{k-1}\widetilde{\mathbf{P}}_{k-1}\mathbf{A}_{k-1}^\top + \mathbf{Q}_{k-1} \tag{C.10a}$$

$$\left[\bar{\mathbf{P}}_k\right]^{-1} = \left(\mathbf{A}_{k-1}\widetilde{\mathbf{P}}_{k-1}\mathbf{A}_{k-1}^\top + \mathbf{Q}_{k-1}\right)^{-1} \tag{C.10b}$$

$$= \left(\mathbf{M}_k + \mathbf{Q}_{k-1}\right)^{-1} \tag{C.10c}$$

$$= \mathbf{M}_k - \left(\mathbf{M}_k^{-1} + \mathbf{M}_k^{-1}\mathbf{Q}_{k-1}^{-1}\mathbf{M}_k^{-1}\right)^{-1} \tag{C.10d}$$

$$= \mathbf{M}_k - \mathbf{M}_k\left(\mathbf{M}_k + \mathbf{Q}_{k-1}^{-1}\right)^{-1}\mathbf{M}_k \tag{C.10e}$$

$$= \mathbf{M}_k - \mathbf{M}_k\left[\mathbf{\Sigma}_k\right]^{-1}\mathbf{M}_k\,. \tag{C.10f}$$

Substitution yields the information prediction

$$\bar{\mathbf{Z}}_k = \mathbf{M}_k - \mathbf{M}_k\left[\mathbf{\Sigma}_k\right]^{-1}\mathbf{M}_k\,, \tag{C.11}$$

where

$$\mathbf{\Sigma}_k = \mathbf{M}_k + \mathbf{Q}_{k-1}^{-1}\,, \tag{C.12a}$$

$$\mathbf{M}_k = \left[\mathbf{A}_{k-1}^\top\right]^{-1}\widetilde{\mathbf{z}}_{k-1}\mathbf{A}_{k-1}^{-1}\,. \tag{C.12b}$$

This formulation of $\mathbf{M}_k$ results from simple matrix manipulation and substitution:

$$\mathbf{M}_k = \left(\mathbf{A}_{k-1}\widetilde{\mathbf{P}}_{k-1}\mathbf{A}_{k-1}^\top\right)^{-1} \tag{C.13a}$$

$$= \left[\mathbf{A}_{k-1}^\top\right]^{-1}\left[\widetilde{\mathbf{P}}_{k-1}\right]^{-1}\left[\mathbf{A}_{k-1}\right]^{-1} \tag{C.13b}$$

$$= \left[\mathbf{A}_{k-1}^\top\right]^{-1}\widetilde{\mathbf{z}}_{k-1}\mathbf{A}_{k-1}^{-1}\,. \tag{C.13c}$$

D. CONVERSION OF A COVARIANCE FUNCTION TO A STATE-SPACE MODEL

A covariance function can be converted to a hybrid state-space model (SSM), which can then be converted to a discrete SSM. Consider the system described with a Gaussian Process and a measurement model with additive noise denoted as

$$f(t) \sim \mathcal{GP}\big(0, k(t,t')\big)\,, \tag{D.1a}$$

$$y_k = f(t_k) + r_k\,, \qquad r_k \sim \mathcal{N}\big(0, R\big)\,, \tag{D.1b}$$

where $R = \sigma_{\mathrm{n}}^2$. The conversion process consists of the following six steps:

1. If the covariance function is stationary, then the spectral density $S(\omega)$ of $f(t)$ can be computed using a Fourier transform such that

$$S(\omega) = \int_{-\infty}^{\infty} k(\tau)\, \mathrm{e}^{-i\omega\tau} \mathrm{d}\tau\,, \tag{D.2}$$

   where $k(\tau)$ is the covariance function and $\tau = |t - t'|$.

2. Identify the transfer function $G(i\omega)$ and the spectral density q_c of the continuous-time white noise with

$$S(\omega) = G(i\omega)\, q_c G(-i\omega)\ . \tag{D.3}$$

   Not all covariance functions contain an analytic solution for the spectral density $S(\omega)$, but can be approximated using a Taylor series expansion or Padé approximation.

3. Convert the transfer function $G(i\omega)$ into a hybrid SSM

$$\dot{\mathbf{x}}(t) = \mathbf{F}\mathbf{x}(t) + \mathbf{L}w(t)\ , \tag{D.4a}$$

$$y_k = \mathbf{H}\mathbf{x}_k + r_k\,, \qquad r_k \sim \mathcal{N}\big(\mathbf{0}, R\big)\,, \tag{D.4b}$$

where the dynamic process is continuous, the observation process is discrete, and the state $\mathbf{x}(t)$ contains n stochastic processes

$$\mathbf{x}(t) = \begin{bmatrix} f(t) & \frac{\mathrm{d}f(t)}{\mathrm{d}t} & \cdots & \frac{\mathrm{d}^{n-1}f(t)}{\mathrm{d}t^{n-1}} \end{bmatrix}^\top\,. \tag{D.5}$$

4. Solve the continuous Lyapunov equation for the covariance matrix $\mathbf{P}_\infty$

$$\frac{\mathrm{d}\mathbf{P}_\infty}{\mathrm{d}t} = \mathbf{F}\mathbf{P}_\infty + \mathbf{P}_\infty\mathbf{F}^\top + \mathbf{L}q_c\mathbf{L}^\top = \mathbf{0}\,. \tag{D.6}$$

5. Solve for the process matrix $\mathbf{A}_k$

$$\mathbf{\Phi}(\tau) = \exp(\mathbf{F}\tau) \tag{D.7a}$$

$$\mathbf{A}_k = \mathbf{\Phi}(\Delta t_k) \tag{D.7b}$$

where $\Delta t_k = t_{k+1} - t_k$.

6. Solve the discrete Lyapunov equation for the covariance matrix $\mathbf{Q}_k$. If the covariance matrix is stable (stationary), then use

$$\mathbf{Q}_k = \mathbf{P}_\infty - \mathbf{A}_k\mathbf{P}_\infty\mathbf{A}_k^\top\,. \tag{D.8}$$

If the covariance matrix is unstable, then use

$$\mathbf{Q}_k = \int_0^{\Delta t_k} \mathbf{\Phi}(\Delta t_k - \tau)\,\mathbf{L}q_c\mathbf{L}^\top\mathbf{\Phi}(\Delta t_k - \tau)^\top\,\mathrm{d}\tau\,. \tag{D.9}$$

This provides all the coefficient matrices to define the discrete SSM

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}\,, \qquad \mathbf{q}_{k-1} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{Q}_{k-1}\big)\,, \tag{D.10a}$$

$$y_k = \mathbf{H}\mathbf{x}_k + r_k, \qquad r_k \sim \mathcal{N}\big(0, R\big)\,. \tag{D.10b}$$

D.1 An Example Conversion of the Matérn Covariance Function

This example provides the steps to demonstrate the relation between covariance functions and a SSM. Consider the Matérn covariance function

$$k_\nu(\tau) = \sigma_{\mathrm{M}}^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{\tau}{\ell}\right)^\nu K_\nu\left(\sqrt{2\nu}\frac{\tau}{\ell}\right)\,. \tag{D.11}$$

where $\Gamma(\nu)$ is the gamma function, K_ν is the modified Bessel function of the second kind, and $\nu, \sigma, \ell > 0$ are smoothness, magnitude, and length parameters respectively. A common simplification $\nu = 3/2$ provides the covariance function

$$k_{3/2}(\tau) = \sigma_{\mathrm{M32}}^2 \left(1 + \frac{\sqrt{3}}{\ell}\tau\right) \exp\left(-\frac{\sqrt{3}}{\ell}\tau\right)\,. \tag{D.12}$$

Step (1) is to calculate the spectral density by evaluating the Fourier transform of the covariance function:

$$S(\omega) = \int_{-\infty}^{\infty} k(\tau)\, \mathrm{e}^{-i\omega t}\mathrm{d}t \tag{D.13a}$$

$$= \int_{-\infty}^{\infty} \sigma_{\mathrm{M32}}^2 \left(1 + \frac{\sqrt{3}}{\ell}|t|\right) \mathrm{e}^{-\frac{\sqrt{3}}{\ell}|t|}\mathrm{e}^{-i\omega t}\mathrm{d}t \tag{D.13b}$$

$$= \frac{4\lambda^3\sigma_{\mathrm{M32}}^2}{\left(\lambda + i\omega\right)^2\left(\lambda - i\omega\right)^2}\,, \qquad \text{where } \lambda = \frac{\sqrt{3}}{\ell}\,. \tag{D.13c}$$

In step (2), the spectral density $S(\omega)$ is related to the transfer function $G(i\omega)$ and the spectral density q_c by the relation

$$S(\omega) = \frac{4\lambda^3\sigma_{\mathrm{M32}}^2}{\left(\lambda + i\omega\right)^2\left(\lambda - i\omega\right)^2} = G(i\omega)\, q_c G(-i\omega)\,. \tag{D.14}$$

If the spectral density is

$$q_c = 4\lambda^3\sigma^2\,, \tag{D.15}$$

then the transfer function is

$$G(i\omega) = \frac{1}{(\lambda + i\omega)^2}\,. \tag{D.16}$$

For step (3), the expanded transfer function

$$G(i\omega) = \frac{1}{(i\omega)^2 + 2\lambda\,(i\omega) + \lambda^2} \tag{D.17}$$

may be converted into a SSM in controllable canonical form:

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{D.18}$$

However, the observation model had additive noise, which accounts for

$$R = \sigma_{\mathrm{n}}^2 \tag{D.19}$$

in the hybrid SSM.

For step (4), solve the continuous-time Lyapunov equation for the covariance matrix $\mathbf{P}_\infty$:

$$\frac{\mathrm{d}\mathbf{P}_\infty}{\mathrm{d}t} = \mathbf{F}\mathbf{P}_\infty + \mathbf{P}_\infty\mathbf{F}^\top + \mathbf{L}q_c\mathbf{L}^\top = \mathbf{0} \tag{D.20}$$

$$\begin{bmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{bmatrix} + \begin{bmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{bmatrix} \begin{bmatrix} 0 & -\lambda^2 \\ 1 & -2\lambda \end{bmatrix} \tag{D.21}$$

$$+ \begin{bmatrix} 0 \\ 1 \end{bmatrix} 4\lambda^3\sigma_{\mathrm{M32}}^2 \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{D.22}$$

$$\begin{bmatrix} P_{12} & P_{22} \\ -\lambda^2 P_{11} - 2\lambda P_{12} & -\lambda^2 P_{12} - 2\lambda P_{22} \end{bmatrix} + \begin{bmatrix} P_{12} & -\lambda^2 P_{11} - 2\lambda P_{12} \\ P_{22} & -\lambda^2 P_{12} - 2\lambda P_{22} \end{bmatrix} \tag{D.23}$$

$$+ \begin{bmatrix} 0 & 0 \\ 0 & 4\lambda^3 \sigma_{\mathrm{M32}}^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{D.24}$$

$$\begin{bmatrix} 2P_{12} & P_{22} - \lambda^2 P_{11} - 2\lambda P_{12} \\ P_{22} - \lambda^2 P_{11} - 2\lambda P_{12} & -2\lambda^2 P_{12} - 4\lambda P_{22} + 4\lambda^3 \sigma_{\mathrm{M32}}^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{D.25}$$

Since $P_{12} = 0$, the equation simplifies to

$$\begin{bmatrix} 0 & P_{22} - \lambda^2 P_{11} \\ P_{22} - \lambda^2 P_{11} & -4\lambda P_{22} + 4\lambda^3 \sigma_{\mathrm{M32}}^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{D.26}$$

Since $P_{22} = \lambda^2 \sigma_{\mathrm{M32}}^2$, then $P_{11} = \sigma_{\mathrm{M32}}^2$. Consequently, the covariance matrix is

$$\mathbf{P}_\infty = \sigma_{\mathrm{M32}}^2 \begin{bmatrix} 1 & 0 \\ 0 & \lambda^2 \end{bmatrix}. \tag{D.27}$$

For step (5), solve for the process matrix

$$\mathbf{A}_k = \exp\left(\mathbf{F}\Delta t_k\right) \tag{D.28a}$$

$$= \exp\left(\begin{bmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{bmatrix} \Delta t_k\right) \tag{D.28b}$$

$$= \begin{bmatrix} 1 & \mathrm{e}^{\Delta t_k} \\ \mathrm{e}^{-\lambda^2 \Delta t_k} & \mathrm{e}^{-2\lambda \Delta t_k} \end{bmatrix}. \tag{D.28c}$$

For step (6), solve the discrete Lyapunov equation for the covariance matrix $\mathbf{Q}_k$. If the covariance matrix is stable (stationary), then use

$$\mathbf{Q}_k = \mathbf{P}_\infty - \mathbf{A}_k \mathbf{P}_\infty \mathbf{A}_k^\top \tag{D.29a}$$

$$= \sigma_{\mathrm{M32}}^2 \begin{bmatrix} 1 & 0 \\ 0 & \lambda^2 \end{bmatrix} - \sigma_{\mathrm{M32}}^2 \begin{bmatrix} 1 & \mathrm{e}^{\Delta t_k} \\ \mathrm{e}^{-\lambda^2 \Delta t_k} & \mathrm{e}^{-2\lambda \Delta t_k} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \lambda^2 \end{bmatrix} \begin{bmatrix} 1 & \mathrm{e}^{-\lambda^2 \Delta t_k} \\ \mathrm{e}^{\Delta t_k} & \mathrm{e}^{-2\lambda \Delta t_k} \end{bmatrix} \tag{D.29b}$$

$$= \sigma_{\mathrm{M32}}^2 \begin{bmatrix} 1 & 0 \\ 0 & \lambda^2 \end{bmatrix} - \sigma_{\mathrm{M32}}^2 \begin{bmatrix} 1 & \mathrm{e}^{\Delta t_k} \\ \mathrm{e}^{-\lambda^2 \Delta t_k} & \mathrm{e}^{-2\lambda \Delta t_k} \end{bmatrix} \begin{bmatrix} 1 & \mathrm{e}^{-\lambda^2 \Delta t_k} \\ \lambda^2 \mathrm{e}^{\Delta t_k} & \lambda^2 \mathrm{e}^{-2\lambda \Delta t_k} \end{bmatrix} \tag{D.29c}$$

$$= \sigma_{\mathrm{M32}}^2 \left(\begin{bmatrix} 1 & 0 \\ 0 & \lambda^2 \end{bmatrix} - \begin{bmatrix} 1 + \lambda^2 \mathrm{e}^{\Delta t_k} \mathrm{e}^{\Delta t_k} & \mathrm{e}^{-\lambda^2 \Delta t_k} + \lambda^2 \mathrm{e}^{(1-2\lambda)\Delta t_k} \\ \mathrm{e}^{-\lambda^2 \Delta t_k} + \lambda^2 \mathrm{e}^{(1-2\lambda)\Delta t_k} & \mathrm{e}^{-2\lambda^2 \Delta t_k} + \lambda^2 \mathrm{e}^{-4\lambda \Delta t_k} \end{bmatrix} \right) \tag{D.29d}$$

$$= \sigma_{\mathrm{M32}}^2 \begin{bmatrix} -\lambda^2 \mathrm{e}^{2\Delta t_k} & -\mathrm{e}^{-\lambda^2 \Delta t_k} - \lambda^2 \mathrm{e}^{(1-2\lambda)\Delta t_k} \\ -\mathrm{e}^{-\lambda^2 \Delta t_k} - \lambda^2 \mathrm{e}^{(1-2\lambda)\Delta t_k} & \lambda^2 - \mathrm{e}^{-2\lambda^2 \Delta t_k} - \lambda^2 \mathrm{e}^{-4\lambda \Delta t_k} \end{bmatrix}. \tag{D.29e}$$

E. DEMONSTRATION OUTPUTS

This section includes the outputs for all the demos in the dissertation with an adaptive component. An 'iteration' is one step of the BFGS algorithm while a 'function count' is the number of calls to the objective function and the gradient. The objective function (i.e., the energy function) and the gradient (i.e., the sensitivity equations) may be called multiple times in an iteration as the BFGS algorithm determines the next step. The function count is labeled 'Func-count' and the gradient count is labeled 'Grad-count'. The value of the objective is labeled 'f(x)'.

E.1 Adaptive Gaussian Process Regression

The following results correspond to the demonstration in Figs. 2.4 and 2.5.

Adaptive Gaussian Process Regression

Iteration	Func-count	Grad-count	f(x)	Step-size
0	1	1	640.295	
1	2	2	436.641	0.0044304
2	6	6	314.045	0.0150067
3	8	8	254.833	0.1
4	11	11	214.002	0.0424256
5	14	14	180.284	0.0330938
6	16	16	172.834	0.443656
7	19	19	150.255	0.19
8	22	22	145.517	0.137051
9	24	24	143.038	0.154695
10	26	26	139.04	0.1
11	28	28	137.522	0.354756
12	29	29	137.193	1
13	30	30	136.899	1
14	31	31	136.834	1
15	32	32	136.778	1
16	33	33	136.758	1
17	34	34	136.754	1
18	35	35	136.753	1
19	36	36	136.753	1
20	37	37	136.753	1

```
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Line search cannot find an acceptable point along the
                current search direction
 Iterations : 21
 Function Count : 47
 Minimum found : 136.7531
 Intern Time : 0.014237 seconds
 Total Time : 5.6622 seconds
```

gp parameters	Func-count=1	Func-count=47
sexp.magnSigma2’	5000	1.3828e+05
sexp.lengthScale’	100	116.29
periodic.magnSigma2’	1	6.3212
periodic.lengthScale’	1	1.3561
periodic.lengthScale_sexp’	100	103.13
matern52.magnSigma2’	10	0.29099
matern52.lengthScale’	10	0.6578
gaussian.sigma2’	1	0.046015

E.2 Adaptive Kalman Filter

The following results correspond to the demonstration in Fig. 2.8.

Adaptive Kalman Filter

Iteration	Func-count	Grad-count	f(x)	Step-size
0	1	1	639.94	
1	2	2	436.331	0.0044296
2	7	7	317.035	0.011875
3	10	10	222.677	0.217614
4	14	14	186.646	0.00625
5	18	18	151.077	0.0704201
6	20	20	146.906	0.1
7	23	23	143.282	0.235047
8	24	24	138.177	1
9	26	26	137.839	0.288216
10	28	28	137.028	0.250907
11	29	29	136.693	1
12	30	30	136.646	1
13	31	31	136.636	1
14	32	32	136.634	1
15	33	33	136.634	1
16	34	34	136.634	1
17	35	35	136.634	1
18	36	36	136.634	1

```
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in x was smaller than the specified tolerance TolX.
 Iterations : 19
 Function Count : 37
 Minimum found : 136.634
 Intern Time : 0.01862 seconds
 Total Time : 118.2705 seconds
```

gp parameters	Func-count=1	Func-count=37
sexp.magnSigma2	5000	1.4182e+05
sexp.lengthScale	100	124.35
periodic.magnSigma2	1	6.3375
periodic.lengthScale	1	1.3569
periodic.lengthScale_sexp	100	105.22
matern52.magnSigma2	10	0.28992
matern52.lengthScale	10	0.65691
gaussian.sigma2	1	0.046004

E.3 A Comparison of Models for Aerial Target Tracking in 1-D

The following results correspond to the demonstration in Fig. 2.9.

Target Tracking with the Near Constant Acceleration Model

Iteration	Func-count	Grad-count	f(x)	Step-size
0	1	1	301.231	
1	2	2	177.692	0.00503306
2	3	3	137.016	1
3	4	4	134.083	1
4	5	5	130.11	1
5	6	6	129.679	1
6	7	7	129.511	1
7	8	8	129.344	1
8	9	9	129.146	1
9	10	10	129.079	1
10	11	11	129.068	1
11	12	12	129.057	1
12	13	13	129.048	1
13	14	14	129.041	1
14	15	15	129.035	1
15	16	16	129.031	1
16	17	17	129.028	1
17	18	18	129.026	1
18	19	19	129.025	1
19	20	20	129.024	1
20	21	21	129.024	1
21	22	22	129.024	1
...	...	...	...	...
28	29	29	129.024	1
29	30	30	129.024	1
30	31	31	129.024	1
31	32	32	129.024	1

```
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in the objective function value was less than TolFun.
 Iterations : 32
 Function Count : 33
 Minimum found : 129.0236
 Intern Time : 0.0092129 seconds
 Total Time : 0.54243 seconds
```

gp parameters	Func-count=1	Func-count=33
ncam.qc	0.1	0.0011271
ncam.p11	1	6.0114e-07
ncam.p22	1	0.25926
ncam.p33	1	2.8091e-21
gaussian.sigma2	1	12.53

The following results correspond to the demonstration in Fig. 2.10.

```
                    Target Tracking with the Singer Acceleration Model
 Iteration  Func-count   Grad-count         f(x)         Step-size
     0           1           1              313.05
     1           2           2             174.151      0.00435452
     2           3           3             148.671               1
     3           4           4             134.132               1
     4           5           5             131.169               1
     5           6           6             129.574               1
     6           7           7             129.346               1
     7           8           8             128.781               1
     8           9           9             128.674               1
     9          10          10             128.565               1
    10          11          11             128.527               1
    11          12          12             128.498               1
    12          13          13             128.493               1
    13          14          14              128.49               1
    14          15          15             128.486               1
    15          16          16             128.477               1
    16          17          17             128.467               1
    17          18          18             128.457               1
    18          19          19             128.452               1
    19          20          20              128.45               1
    20          21          21             128.448               1
    21          22          22             128.447               1
    22          23          23             128.446               1
    23          24          24             128.446               1
   ...         ...         ...                 ...             ...
    34          35          35             128.446               1
    35          36          36             128.446               1
    36          37          37             128.446               1
    37          38          38             128.446               1
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in the objective function value was less than TolFun.
 Iterations : 38
 Function Count : 39
 Minimum found : 128.4458
 Intern Time : 0.009633 seconds
 Total Time : 0.77271 seconds

gp parameters               Func-count=1       Func-count=39
 singer.alpha                     0.01             0.15269
 singer.magnSigma2                 0.1           0.0097857
 singer.p11                          1          1.7312e-07
 singer.p22                          1             0.23625
 singer.p33                          1          7.6397e-12
 gaussian.sigma2                     1              12.656
```

The following results correspond to the demonstration in Fig. 2.11.

```
Target Tracking with the Linear and Constant Covariance Function
 Iteration  Func-count   Grad-count         f(x)         Step-size
     0           1           1             526.862
     1           2           2             247.261      0.00219861
     2           3           3             188.056               1
     3           4           4             153.028               1
     4           5           5             141.977               1
     5           6           6             138.891               1
     6           7           7             138.427               1
     7           8           8               138.4               1
     8           9           9             138.399               1
     9          10          10             138.399               1
    10          11          11             138.399               1
    11          12          12             138.399               1
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in x was smaller than the specified tolerance TolX.
 Iterations : 12
 Function Count : 13
 Minimum found : 138.3993
 Intern Time : 0.0029998 seconds
 Total Time : 0.17121 seconds

gp parameters               Func-count=1       Func-count=13
 linear.coeffSigma2                1            0.36254
 constant.constSigma2            100             59.149
 gaussian.sigma2                   1              22.17
```

The following results correspond to the demonstration in Fig. 2.12.

Target Tracking with the Matérn ($\nu = 3/2$) Covariance Function

Iteration	Func-count	Grad-count	f(x)	Step-size
0	1	1	154.687	
1	3	3	146.019	0.0261052
2	5	5	145.632	0.142265
3	6	6	144.846	1
4	10	10	140.314	0.0366811
5	13	13	140.164	0.0118601
6	14	14	139.263	1
7	15	15	137.547	1
8	16	16	136.723	1
9	17	17	135.64	1
10	18	18	132.852	1
11	20	20	131.798	0.217284
12	21	21	130.843	1
13	22	22	130.814	1
14	24	24	130.672	4
15	26	26	130.118	0.5
16	27	27	130.08	1
17	28	28	130.047	1
18	29	29	130.046	1
19	30	30	130.046	1

Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in x was smaller than the specified tolerance TolX.
 Iterations : 20
 Function Count : 31
 Minimum found : 130.0461
 Intern Time : 0.0064547 seconds
 Total Time : 0.32607 seconds

gp parameters	Func-count=1	Func-count=31
matern32.magnSigma2	100	146.63
matern32.lengthScale	1	30.694
gaussian.sigma2	1	12.918

The following results correspond to the demonstration in Fig. 2.13.

Target Tracking with the Matérn ($\nu = 5/2$) Covariance Function

Iteration	Func-count	Grad-count	f(x)	Step-size
0	1	1	153.671	
1	3	3	149.366	0.0114901
2	5	5	148.802	0.223596
3	7	7	148.44	0.169734
4	10	10	146.355	2.29832
5	13	13	146.301	0.0204562
6	15	15	143.697	4
7	16	16	136.372	1
8	17	17	132.935	1
9	19	19	130.819	0.221199
10	20	20	129.737	1
11	21	21	129.616	1
12	22	22	129.569	1
13	23	23	129.56	1
14	24	24	129.502	1
15	25	25	129.498	1
16	26	26	129.497	1
17	27	27	129.497	1

Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in the objective function value was less than TolFun.
 Iterations : 18
 Function Count : 28
 Minimum found : 129.497
 Intern Time : 0.0052582 seconds
 Total Time : 0.28189 seconds

gp parameters	Func-count=1	Func-count=28
matern52.magnSigma2	100	142.82
matern52.lengthScale	1	24.033
gaussian.sigma2	1	12.811

The following results correspond to the demonstration in Fig. 2.14.

Target Tracking with the Squared Exponential Covariance Function

Iteration	Func-count	Grad-count	f(x)	Step-size
0	1	1	345.71	
1	2	2	178.198	0.00410757
2	5	5	142.877	0.0796587
3	6	6	141.754	1
4	7	7	141.691	1
5	8	8	141.497	1
6	13	13	129.042	34.6397
7	15	15	129.034	0.1
8	17	17	128.954	0.151668
9	18	18	128.94	1
10	19	19	128.935	1
11	20	20	128.935	1
12	21	21	128.935	1
13	22	22	128.935	1

```
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in x was smaller than the specified tolerance TolX.
 Iterations : 14
 Function Count : 23
 Minimum found : 128.9347
 Intern Time : 0.0043366 seconds
 Total Time : 0.36834 seconds
```

gp parameters	Func-count=1	Func-count=23
sexp.magnSigma2	1e+05	129.23
sexp.lengthScale	100	17.297
gaussian.sigma2	1	12.661

The following results correspond to the demonstration in Fig. 2.15.

```
Target Tracking with the Exponential Covariance Function
 Iteration  Func-count   Grad-count         f(x)         Step-size
     0           1           1             1004.49
     1           2           2             362.214      0.00167671
     2           4           4             243.852       0.305434
     3           8           8              141.65       0.209991
     4          11          11             139.758        0.12292
     5          12          12             139.733              1
     6          14          14             139.659              4
     7          21          21             137.827        2.22853
     8          24          24             137.798       0.0863443
     9          25          25             137.092              1
    10          26          26             137.026              1
    11          27          27             137.023              1
    12          28          28             137.023              1
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in x was smaller than the specified tolerance TolX.
 Iterations : 13
 Function Count : 30
 Minimum found : 137.0233
 Intern Time : 0.0049239 seconds
 Total Time : 0.15249 seconds

gp parameters               Func-count=1       Func-count=30
 exp.magnSigma2                    1             114.82
 exp.lengthScale                   1             8.1542
 gaussian.sigma2                   1          0.0023424
```

E.4 Target Tracking of Hypersonic Boost-Glide Vehicle

The following results correspond to the demonstration in Fig. 2.17 with the squared exponential model.

Target Tracking of Hypersonic Boost-Glide Vehicle

Iteration	Func-count	Grad-count	f(x)	Step-size
0	1	1	711.715	
1	2	2	421.339	0.00207925
2	6	6	405.299	0.00891881
3	7	7	351.252	1
4	8	8	318.779	1
5	9	9	313.506	1
6	10	10	311.208	1
7	11	11	311.059	1
8	12	12	311.038	1
9	13	13	311.036	1
10	14	14	311.036	1
11	15	15	311.036	1

```
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in x was smaller than the specified tolerance TolX.
 Iterations : 12
 Function Count : 16
 Minimum found : 311.0361
 Intern Time : 0.028485 seconds
 Total Time : 0.84437 seconds
```

gp parameters	Func-count=1	Func-count=16
sexp.magnSigma2	1e+05	2804.2
sexp.lengthScale	100	104.71
gaussian.sigma2	1	15.149

The following results correspond to the demonstration in Fig. 2.17 with the Singer acceleration model.

Target Tracking of Hypersonic Boost-Glide Vehicle

Iteration	Func-count	Grad-count	f(x)	Step-size
0	1	1	1196.44	
1	2	2	702.66	0.00194181
2	4	4	462.178	0.276752
3	6	6	426.926	0.269294
4	7	7	392.219	1
5	10	10	385.656	0.0924621
6	11	11	384.663	1
7	15	15	376.314	3.11838
8	19	19	376.231	0.00625
9	20	20	374.163	1
10	22	22	369.473	0.640249
11	23	23	367.555	1
12	24	24	366.613	1
13	25	25	364.211	1
14	26	26	364.063	1
15	27	27	364.008	1
16	28	28	363.977	1
17	29	29	363.972	1
18	30	30	363.972	1
19	31	31	363.971	1
20	32	32	363.971	1
21	33	33	363.971	1
...	...	...	...	...
66	96	96	310.937	1
67	97	97	310.937	1
68	98	98	310.937	1

Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Change in the objective function value was less than TolFun.
 Iterations : 69
 Function Count : 99
 Minimum found : 310.9371
 Intern Time : 0.036338 seconds
 Total Time : 9.448 seconds

gp parameters	Func-count=1	Func-count=99
singer.alpha	0.01	0.02487
singer.magnSigma2	0.1	8.2665e-05
singer.p11	1	2550.6
singer.p22	1	0.018407
singer.p33	1	8.9816e-11
gaussian.sigma2	1	14.108

E.5 Fault Detection of Aircraft Traffic Data

The following results correspond to the demonstration in Fig. 2.19. Note, the results for the earliest iterations do not show up in the MATLAB workspace due to the large amount of warnings: "`Warning: Matrix is singular, close to singular or badly scaled. Results may be inaccurate. RCOND = NaN.`"

Fault Detection of Latitude Data in Aircraft Traffic

Iteration	Func-count	Grad-count	f(x)	Step-size
...	...	...	...	...
23	52	52	-1205.7	0.0124836
24	53	53	-1211.26	1
25	54	54	-1218.09	1
26	56	56	-1224.24	0.5
27	58	58	-1227.48	0.548427
28	60	60	-1230.24	0.261946
29	61	61	-1234.88	1

```
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Line search cannot find an acceptable point along the
                current search direction
 Iterations : 30
 Function Count : 83
 Minimum found : -1234.8769
 Intern Time : 0.020529 seconds
 Total Time : 93.6651 seconds
```

gp parameters	Func-count=1	Func-count=83
ncam.qc	0.0001	0.000111
ncam.p11	1	2056.7
ncam.p22	1	0.069571
ncam.p33	1	2.3315
matern52.magnSigma2	100	0.0056514
matern52.lengthScale	1	0.046618
gaussian.sigma2	1	0.0001586

The following results correspond to the demonstration in Fig. 2.20. Note, the results for the earliest iterations do not show up in the MATLAB workspace due to the large amount of warnings: "`Warning: Matrix is singular, close to singular or badly scaled. Results may be inaccurate. RCOND = NaN.`"

Fault Detection of Longitude Data in Aircraft Traffic

Iteration	Func-count	Grad-count	f(x)	Step-size
...	...	...	...	...
4	14	14	518.188	0.25
5	15	15	493.969	1
6	16	16	493.091	1
7	17	17	493.032	1
8	37	37	492.994	1.74955
9	42	42	492.993	0.0297453

```
Optimizer Results
 Algorithm Used: Broyden-Fletcher-Goldfarb-Shanno (BFGS)
 Exit message : Line search cannot find an acceptable point along the
                current search direction
 Iterations : 10
 Function Count : 65
 Minimum found : 492.9935
 Intern Time : 0.06276 seconds
 Total Time : 80.3424 seconds
```

gp parameters	Func-count=1	Func-count=65
ncam.qc	0.0001	9.9973e-05
ncam.p11	1	0.26921
ncam.p22	1	0.054843
ncam.p33	1	0.0065388
matern52.magnSigma2	100	5257.1
matern52.lengthScale	1	0.47564
gaussian.sigma2	1	0.21561

VITA

VITA

Michael A. Jacobs was born in Los Angeles, California. He completed his B.S. in Mechanical Engineering in the Department of Aerospace & Mechanical Engineering (AME) at the University of Southern California (USC) in Los Angeles, California in 2011. He then went on to obtain an M.S in in Mechanical Engineering with an emphasis in Dynamics and Control in the AME Department at USC in 2013. He completed his Ph.D. in 2019 from Purdue University in the School of Aeronautics and Astronautics with a Dynamics and Control major and Aerospace Systems minor.