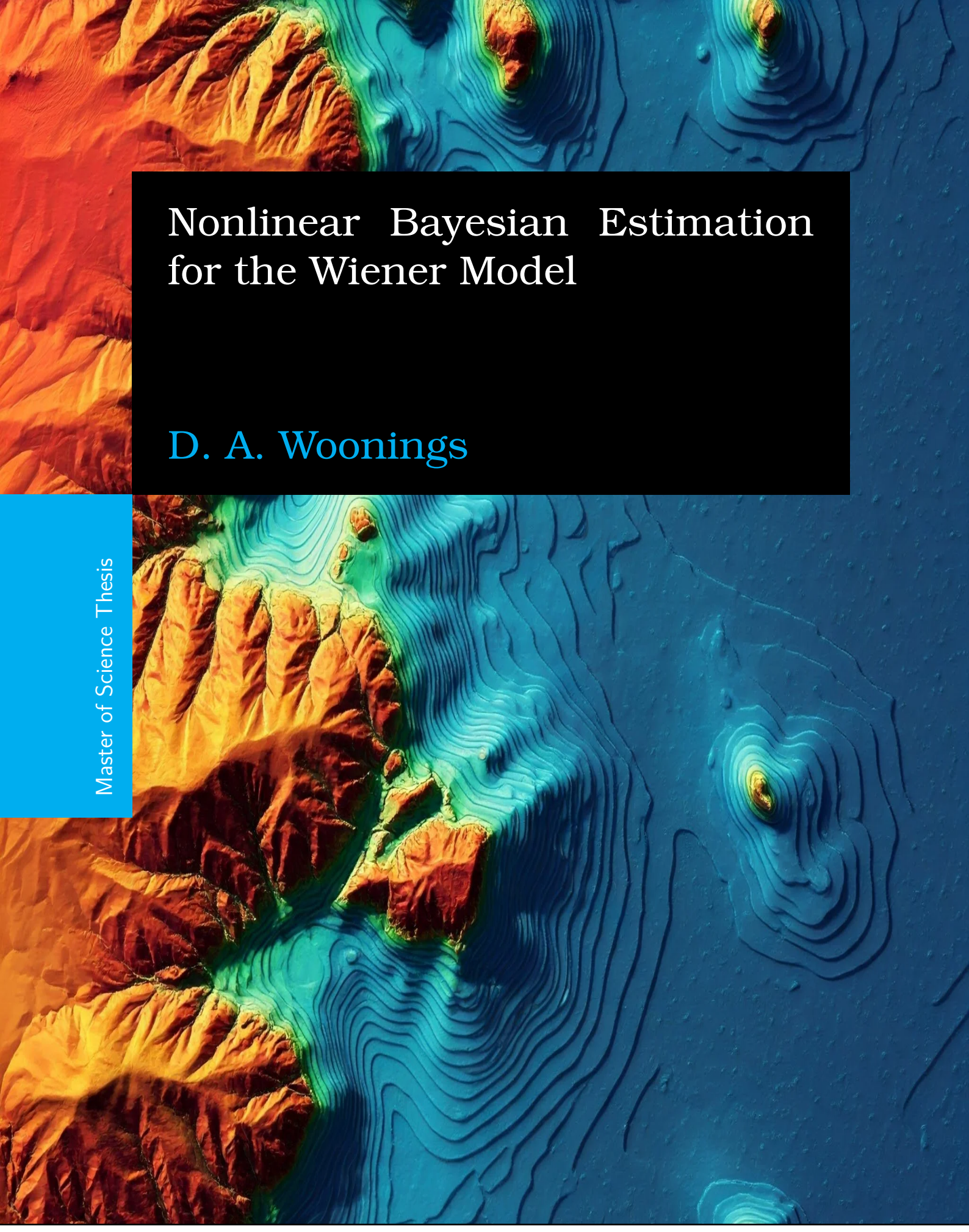

Nonlinear Bayesian Estimation for the Wiener Model

D. A. Woonings

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Nonlinear Bayesian Estimation for the Wiener Model

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

D. A. Woonings

January 26, 2026

Faculty of Mechanical Engineering (ME) · Delft University of Technology

Abstract

This thesis addresses the challenge of autonomous bathymetric mapping, the process of creating topographical maps of the ocean floor, using an Autonomous Underwater Vehicle (AUV). Conventional surveys rely on expensive crewed vessels and often fail to exploit the underlying statistical patterns of the collected bathymetric data. This work proposes a model-driven estimation framework that treats bathymetric mapping as a nonlinear system identification problem.

Using properties of the underlying state-space model and by representing the seabed through a linear combination of known basis functions, a novel estimation algorithm is derived. The proposed method formulates a dual Bayesian state-parameter estimator in which affine Minimum Mean Square Error (MMSE) estimators are constructed for both the unknown model parameters and the latent system states. By alternating between these closed-form estimators in a fixed-point iteration, the algorithm progressively reduces state-induced uncertainty and improves parameter estimation accuracy.

Finally, a filtering extension of the dual state-parameter estimator is introduced. This extension enables scalable processing of large data sets by operating online, while incurring a limited loss in accuracy compared to the batch formulation. The proposed estimators are evaluated in Monte Carlo experiments and benchmarked against a state-of-the-art methods. The results show that the proposed methods consistently outperform existing estimation algorithms in terms of accuracy and computational complexity.

Table of Contents

Acknowledgements

This work would not have been possible without the guidance, insight, and support of my supervisors, Peyman and Sasan. This project provided me with a valuable introduction to academic research, and I am sincerely grateful for their mentorship throughout this work. Additionally, I am deeply thankful to my parents, Claar and Frank, for their continued support throughout the years.

Delft, University of Technology
January 26, 2026

Daniël Woonings

Chapter 1

Introduction

Bathymetric charts represent the topography of the ocean floor by mapping the seabed depth as a function of geographical coordinates. Accurate bathymetric maps are crucial for maritime operations, including navigation and route planning [39]. They are also essential for engineering offshore structures and for deploying undersea infrastructure such as telecommunication and power cables [43].

Conventional bathymetric surveys are typically conducted using crew-operated surface vessels equipped with Global Positioning System (GPS) for accurate georeferencing. These surveys use sonar sensors to systematically measure the distance from the vessel to the seabed [21]. Survey lines are usually run in a "lawnmower" pattern, ensuring full coverage and minimizing gaps [22]. While these methods achieve high accuracy, they require large crews, expensive vessels, and cover limited areas.

After data collection, missing values are filled using algorithms that apply advanced interpolation techniques to generate a digital terrain model [8]. These methods make very limited assumptions about the structure of the seabed, focusing primarily on reducing noise from sonar sensor measurements [33]. By not leveraging the statistical patterns in the bathymetric data, conventional approaches further increase both operational complexity and overall costs.

The high operational cost of conventional surveys has motivated the development of scalable, autonomous mapping approaches. Autonomous Underwater Vehicle (AUV)s offer a key solution [58], equipped with onboard sensors, navigation systems, and sonar to perform underwater surveys independently. Multiple AUVs can operate simultaneously, increasing coverage and reducing mission duration while minimizing human and vessel requirements.

In parallel, advances in mapping algorithms have enabled model driven representations of the seabed [35]. Rather than treating missing data through local interpolation, these approaches exploit statistical patterns within the bathymetric data to produce more accurate and consistent terrain models. These methods include techniques from machine learning [54] and Bayesian inference [2]. By combining AUV-based data collection with sophisticated algorithms, it is possible to achieve autonomous bathymetric mapping in a cost-effective manner.

While AUVs enable autonomous exploration, they also introduce the challenge of underwater navigation, which is particularly difficult due to the absence of a ground truth position measurement, like GPS. Consequently, AUVs must rely onboard estimation algorithms for position estimation. As the ultimate objective is to learn the bathymetric map as a function of the vehicle's position, accurate localization is critical.

1-1 Goal Statement

The overall goal of this thesis is to develop and evaluate a principled estimation framework for learning bathymetric charts from data collected by an AUV. This goal is pursued through three specific subgoals.

1) Model Driven Representation of Seabed and Estimation Problem Formulation

The first objective of this thesis is to provide a rigorous formulation of the bathymetric learning problem. Rather than relying on data-driven interpolation techniques, this work adopts a model-driven representation of the seabed. A model-based description enables incorporation of prior knowledge and facilitates extrapolation beyond sampled regions. Moreover, it allows the accuracy of the resulting map to be controlled through the choice of model complexity, enabling coarser representations to be learned from reduced sampling when fine-scale detail is not required.

Combined with the AUV data collection process, the model-driven representation defines an input–output relationship: the vehicle trajectory serves as the input to a nonlinear static map that produces bathymetric measurements. Together with the vehicle dynamics, this naturally leads to a state-space formulation of the data collection process. As Bayesian inference provides a principled and well-established framework for estimation of state-space models, the first goal of this thesis is to formulate the bathymetric learning problem explicitly as a Bayesian inference problem.

2) Development of the Dual State-Parameter Estimator

The second objective is to develop an estimation algorithm that jointly addresses vehicle localization and bathymetric map learning within a unified probabilistic framework. Because the seabed is represented by a parameterized model, the goal is to estimate an unknown vector of parameters from noisy measurements.

Accurate bathymetric estimation requires precise knowledge of the AUV trajectory, while reliable localization benefits from an accurate seabed representation. This interdependence creates a dual estimation problem, in which both the system state and unknown map parameters must be inferred simultaneously.

To address this challenge, this thesis proposes the dual state-parameter estimator. By leveraging the structure of the estimation problem, the approach decomposes the overall task into two tractable subproblems, one for vehicle localization and one for bathymetric map estimation. The proposed formulation offers a systematic alternative to the two dominant paradigms in the state-space model literature.

3) Validation and Benchmarking in Simulation

The final objective of this thesis is to evaluate the performance of the dual state-parameter estimation in simulation. The dual state-parameter estimator will be assessed through Monte Carlo experiments and benchmarked against two state-of-the-art approaches. To further assess real-world applicability, a Unity-based simulator will be employed. This simulator enables generation of realistic bathymetric data that can closely mimic operational environments.

1-2 Organisation

Chapter 2 addresses the first subgoal of the thesis by introducing the modelling abstractions for the AUV dynamics and the model of the seabed.

The core contribution of this thesis is presented in Chapter 3, which develops the dual state–parameter estimator for the joint estimation of the system states and the unknown seabed parameters. The proposed approach is positioned within existing literature in Chapter 4.

The estimator in Chapter 3 is formulated in a batch setting and its computational complexity limits scalability. Chapter 6 therefore introduces a recursive implementation of the dual state–parameter estimator, enabling processing of large data sets.

The performance of the batch dual state-parameter estimator is evaluated in Chapter 5. This chapter presents Monte Carlo simulations under idealized conditions and benchmarks the proposed estimator against two state-of-the-art approaches. To assess practical applicability, a Unity-based simulator is employed to generate a realistic AUV trajectory for testing. The impact of recursive computation on estimation performance is examined through an additional Monte Carlo experiment in Chapter 6.

Finally, Chapter 7 summarizes the main findings of the thesis, evaluates the extent to which the stated goals are achieved, and outlines directions for future work.

Chapter 2

Problem Description

The preceding chapter motivated the need for principled, model-driven approaches to autonomous bathymetric mapping. Building on this motivation, the present chapter translates the high-level concepts introduced earlier into a mathematical problem definition. Specifically, it introduces a state-space model of the Autonomous Underwater Vehicle (AUV) in which the measurements correspond to bathymetric data.

The chapter also presents a parametric formulation of the seabed model, expressed as a linear combination of known basis functions. The learning task is to estimate the unknown coefficients of the seabed model. In this way, the seabed mapping task is cast as a system identification problem. The identification problem is formalised within a Bayesian inference framework, leading to the Maximum a Posteriori (MAP) and Minimum Mean Square Error (MMSE) estimators.

This chapter is organised as follows. Section 2-1 describes the seabed model and the dynamical model of the AUV. Section 2-2 introduces the Wiener model as a formal description of the system and discusses the probabilistic formulation of the state-space model. Section 2-3 introduces the Bayesian estimation problem. Finally, Section 2-4 explains why the resulting estimation problem is challenging.

2-1 System Description

Figure 2-1 shows an abstract 2D view of the AUV model. The AUV position x can be interpreted as the state of a dynamical system, with the control input u corresponding to the planar velocity of the AUV. For simplicity, the AUV position is described using planar coordinates. In practice, the position vector would represent a geosynchronized position.

The AUV dynamics are approximated by a linear-time-varying difference equation (Equation (2-3)), in which the system matrices are assumed to be known. This assumption is reasonable as the AUV dynamics can be identified in a separate experiment using existing system identification techniques.

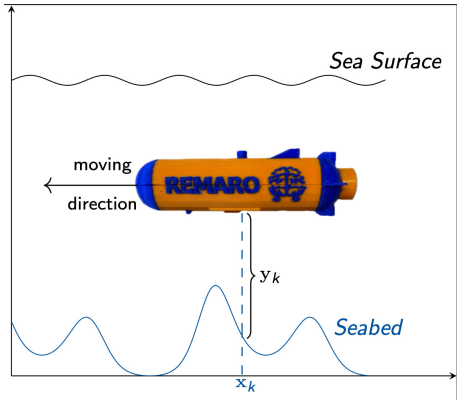

Figure 2-1: Configuration of AUV

The AUV is equipped with a sensor that measures the distance to the seabed. The measurement model of the sensor depends nonlinearly (and only) on the AUV position, denoted by $h_\theta(\cdot)$. From this perspective, the measurement model provides a description of the seabed.

In this work, the seabed is modelled as a linear combination of $N+1$ known basis functions,

$$h_\theta : \mathbb{R}^2 \to \mathbb{R}, \quad h_\theta(\mathrm{x}) = \sum_{n=0}^{\mathrm{N}} \theta_n \phi_n(\mathrm{x}), \tag{2-1}$$

where $\theta \in \mathbb{R}^{\mathrm{N}+1}$, is the vector of unknown parameters, and $\phi_n \colon \mathbb{R}^2 \to \mathbb{R}$, $\phi(\mathrm{x}) = [\phi_0(\mathrm{x}), \ldots, \phi_{\mathrm{N}}(\mathrm{x})]^{\mathsf{T}}$ is the vector of basis functions evaluated at x. Since θ is the only unknown in the seabed model, constructing a map from bathymetric data reduces to finding the parameter vector θ that reproduces the observed depths.

In this work the Fourier basis is selected, the Fourier basis functions are given by:

$$\begin{cases} \phi_0(\mathrm{x}) = 1 & n = 0 \\ \phi_n(\mathrm{x}) = \sum\limits_{\ell \in \{-1,1\}} \exp(j\langle \ell f_n, \mathrm{x} \rangle) & n \geq 1. \end{cases} \tag{2-2}$$

The sum over $\ell \in \{-1, 1\}$ ensures that each basis function is real-valued, because the positive and negative frequency components are included symmetrically. Computationally, the Fourier basis is convenient. When the latent state is Gaussian, the first two statistical moments of the basis functions can be computed in closed form (see Section 3-2). This computation is important because the distribution of $h_\theta(\mathrm{x})$ is unknown, as it is obtained through a nonlinear transformation of a (Gaussian) random variable. Nevertheless, reasoning about the underlying probabilistic model remains possible via its mean and covariance.

The seabed model has two design parameters, namely N and f_n. The model size N determines the amount of basis functions and hence number of parameters to be estimated. The frequency vectors f_n control the spatial resolution of the seabed model. Low frequencies capture large-scale variations, while higher frequencies capture finer details. Geometrically, the seabed can be interpreted as a sum of sinusoids with different frequencies. Adding more frequencies improves the model's ability to represent complex seabeds, but also increases the

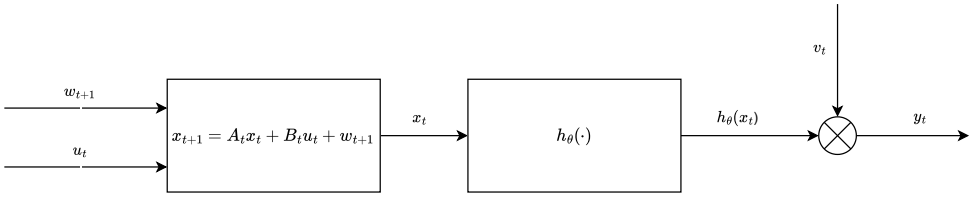

Figure 2-2: Block representation of Wiener model

dimensionality of the inference problem, making estimation more computationally expensive. If the parameter estimation method allows for a large number of parameters, automated design methods for f_n are available [45].

In summary, the AUV is described by a state-space model: the process model is linear, while the observation model is a nonlinear function of the state. A linear process with a nonlinear output is referred to as a Wiener model, which will be formalized in the next section.

2-2 Stochastic Wiener Model

Wiener models are a class of nonlinear state-space models that combine a linear dynamic component with a static nonlinear component [56]. The input first passes through the dynamic linear block, whose output is then transformed by the nonlinear block, as shown in Figure 2-2. Formally, consider a discrete-time linear time-varying system, where the state at time t evolves according to

$$\mathrm{x}_{t+1} = \mathrm{A}_t \mathrm{x}_t + \mathrm{B}_t \mathrm{u}_t + \mathrm{w}_{t+1}, \qquad \mathrm{y}_t = h_\theta(\mathrm{x}_t) + \mathrm{v}_t, \tag{2-3}$$

where $\mathrm{x}_t \in \mathbb{R}^2$ is the latent position/state vector, $\mathrm{A}_t \in \mathbb{R}^{2\times 2}$ and $\mathrm{B}_t \in \mathbb{R}^{2\times 2}$ are the known system matrices, and $\mathrm{u}_t \in \mathbb{R}^2$ represents the sequence of known controlled inputs/velocities. The process noise $\mathrm{w}_{t+1} \sim \mathcal{N}(0, \Sigma_{\mathrm{w}_{t+1}})$ is a zero-mean Gaussian vector with covariance $\Sigma_{\mathrm{w}_{t+1}}$. Observations $\mathrm{y}_t \in \mathbb{R}$ are generated according to Equation (2-1). Measurements are corrupted by additive Gaussian noise $\mathrm{v}_t \sim \mathcal{N}(0, \sigma_{\mathrm{v}_t})$.

The process and measurement noise induce probability distributions over the dynamics and observations. These probability distributions are written as

$$\mathrm{x}_{t+1} \sim p(\mathrm{x}_{t+1}|\mathrm{x}_t), \qquad \mathrm{y}_t \sim p(\mathrm{y}_t|\mathrm{x}_t, \theta). \tag{2-4}$$

Note that the state transition model ($p(\mathrm{x}_{t+1}|\mathrm{x}_t)$) does not depend on the unknown parameters, whereas the observation distribution ($p(\mathrm{y}_t|\mathrm{x}_t, \theta)$) does depend on the unknown parameters. Together, the dynamical system, the unknown parameters, and the statistics of the noise processes define a generative probabilistic model. The likelihood of generating a sequence of states and observations is characterized by the joint distribution:

$$p(\bar{\mathrm{x}}, \bar{\mathrm{y}} \mid \theta) = p(\mathrm{x}_0) \prod_{t=0}^{T} p(\mathrm{y}_t \mid \mathrm{x}_t, \theta) \prod_{t=0}^{T-1} p(\mathrm{x}_{t+1} \mid \mathrm{x}_t), \tag{2-5}$$

where $\bar{\mathrm{x}} = [\mathrm{x}_0, \ldots, \mathrm{x}_T]^\top$ and $\bar{\mathrm{y}} = [\mathrm{y}_0, \ldots, \mathrm{y}_T]^\top$. Having defined the probabilistic model, the estimation task is to infer the unknown parameter vector θ from the observed data $\bar{\mathrm{y}}$. In the next section, this is formalized using a Bayesian decision-theoretic framework.

2-3 Bayes Decision Principle

In Bayesian inference, the prior, observation model, and observed data are combined to estimate the unknown parameters by solving an optimization problem. Formulating this optimization problem requires two additional components: the prior distribution and the loss function. The prior $p(\theta)$ represents the knowledge about θ before observing any data. Throughout this thesis a uniform prior is assumed, i.e.:

$$p(\theta) = \mathcal{U}([a,b]^{(N+1)}). \tag{2-6}$$

The support of $p(\theta)$ is denoted by Θ. The loss function $\ell(\theta,\cdot)$, expresses the discrepancy between the true parameters and the estimated parameters.

The Bayes estimate is defined by minimizing the expected loss with respect to the joint distribution of $\bar{\mathrm{y}}$ and θ:

$$\hat{\theta}(\bar{\mathrm{y}}) = \arg\min_{\nu(\cdot)\in\mathcal{F}_\theta} \mathbb{E}^{p(\theta,\bar{\mathrm{y}})}[\ell(\theta,\nu(\bar{\mathrm{y}}))]\,, \tag{2-7}$$

where $\mathcal{F}_\theta = \left\{\nu\colon \mathbb{R}^{T+1}\to\Theta\right\}$ denotes the space of all possible estimators.

The expectation in (2-7) is taken with respect to the joint distribution $p(\theta,\bar{\mathrm{y}}) = p(\bar{\mathrm{y}} \mid \theta)p(\theta)$, which combines prior information about the parameters with information from the data through the likelihood. Conditioning on the observed measurements yields an equivalent formulation in terms of the posterior distribution $p(\theta \mid \bar{\mathrm{y}})$. Specifically, the expectation (2-7) can be written as:

$$\mathbb{E}^{p(\theta,\bar{\mathrm{y}})}\Big[\ell\big(\theta,\hat{\theta}(\bar{\mathrm{y}})\big)\Big] = \iint \ell\big(\theta,\hat{\theta}(\bar{\mathrm{y}})\big)\,p(\theta,\bar{\mathrm{y}})\,d\theta\,d\bar{\mathrm{y}} = \int\left[\int \ell\big(\theta,\hat{\theta}(\bar{\mathrm{y}})\big)\,p(\theta|\bar{\mathrm{y}})\,d\theta\right]p(\bar{\mathrm{y}})\,d\bar{\mathrm{y}}. \tag{2-8}$$

Here, $p(\theta \mid \bar{\mathrm{y}})$ called posterior distribution, represents the conditional distribution of the parameters θ given the measurements $\bar{\mathrm{y}}$. Since the outer integral depends only on $\bar{\mathrm{y}}$, the Bayes risk can be minimized pointwise. This leads to the equivalent formulation of the Bayes estimator:

$$\hat{\theta}(\bar{\mathrm{y}}) = \arg\min_{\nu(\cdot)\in\mathcal{F}_\theta} \mathbb{E}^{p(\theta|\bar{\mathrm{y}})}\left[\ell\big(\theta,\nu(\bar{\mathrm{y}})\big)\right]. \tag{2-9}$$

Rewriting Equation (2-7) as Equation (2-9) reduces the optimization problem to a pointwise minimization with respect to the observed data. Equation (2-9) defines an abstract Bayesian decision problem and constitutes a key step in the development of this work. At this point, the bathymetric mapping task has been reformulated as a Bayesian inference problem in a state–space framework. To move toward concrete estimation algorithms, the loss function is now specified. In the following, two particular choices of loss are considered, leading to estimators that correspond to two canonical summaries of the posterior distribution: its mean and its mode.

MMSE Estimate

A common Bayesian estimator is the MMSE estimator which is obtained by minimizing the squared-error loss function. The squared-error loss function is defined as:

$$\ell(\theta,\nu(\bar{\mathrm{y}})) = \|\theta-\nu(\bar{\mathrm{y}})\|^2. \tag{2-10}$$

The corresponding Bayes estimator minimizes the expected loss with respect to the posterior distribution of the parameters:

$$\hat{\theta}(\bar{\mathrm{y}}) = \arg\min_{\nu(\cdot)\in\mathcal{F}_\theta} \mathbb{E}^{p(\theta|\bar{\mathrm{y}})}\left[\|\theta - \nu(\bar{\mathrm{y}})\|^2\right]. \tag{2-11}$$

With some minor manipulations (computing gradient and setting to zero), it can be shown that the MMSE estimate corresponds to the posterior mean, i.e:

$$\hat{\theta}(\bar{\mathrm{y}}) = \mathbb{E}[\theta \mid \bar{\mathrm{y}}] = \int \theta\, p(\theta \mid \bar{\mathrm{y}})\, d\theta. \tag{2-12}$$

MAP Estimate

Another widely used Bayesian estimator is the MAP estimate. The MAP estimate can be derived by considering a binary loss function that assigns zero loss only when the estimate lies inside an ϵ-ball centered at the true parameter value, expressed as:

$$\ell^\epsilon(\theta, \nu(\bar{\mathrm{y}})) = \begin{cases} 0, & \|\theta - \nu(\bar{\mathrm{y}})\|^2 < \epsilon, \\ 1, & \text{otherwise.} \end{cases} \tag{2-13}$$

As the radius approaches zero, the corresponding Bayes estimator converges to the mode of the posterior distribution [3]:

$$\arg\min_{\nu(\cdot)\in\mathcal{F}_\theta} \lim_{\epsilon\to 0} \mathbb{E}^{p(\theta|\bar{\mathrm{y}})}\left[\ell^\epsilon\big(\theta, \nu(\bar{\mathrm{y}})\big)\right] = \arg\max_\theta p(\theta \mid \bar{\mathrm{y}}). \tag{2-14}$$

The MAP estimator is therefore defined as

$$\hat{\theta}_{\mathrm{MAP}}(\bar{\mathrm{y}}) \in \arg\max_\theta p(\theta \mid \bar{\mathrm{y}}). \tag{2-15}$$

or equivalently,

$$\hat{\theta}_{\mathrm{MAP}}(\bar{\mathrm{y}}) \in \arg\max_\theta \log p(\bar{\mathrm{y}} \mid \theta) + \log p(\theta). \tag{2-16}$$

2-4 Challenge

The previous section introduced two Bayesian estimators: the MAP estimator (Equation (2-15)) and the MMSE estimator (Equation (2-11)). Both estimators are associated with measures of central tendency of the posterior distribution, namely its mode and mean. From this perspective, the inference problem would be straightforward if a closed form expression of the posterior was known. Unfortunately, this is not the case, which is mainly a result from the fact the sequence of latent states is unknown.

The objective of the next two chapters is to obtain accurate approximations of the posterior distribution, or its mode in the MAP case. This section explains why the Bayesian inference problem is inherently challenging for the Wiener model.

Using Bayes' theorem, the posterior can be written as:

$$p(\theta \mid \bar{\mathrm{y}}) = \frac{p(\bar{\mathrm{y}} \mid \theta)\, p(\theta)}{p(\bar{\mathrm{y}})}. \tag{2-17}$$

Here, $p(\bar{\mathrm{y}} \mid \theta)$ is the observed-data likelihood, and $p(\bar{\mathrm{y}})$ is the model evidence. The fundamental challenge in evaluating Equation (2-17) is that the observed-data likelihood depends on the latent state. As a result, the likelihood cannot be evaluated in closed form. To see this, the observed-data likelihood can be factored as:

$$p(\bar{\mathrm{y}} \mid \theta) = \prod_{t=0}^{T} p(\mathrm{y}_t \mid \mathrm{y}_{1:t-1}, \theta), \tag{2-18}$$

To obtain a meaningful expression for the terms on the right-hand side of Equation (2-18), it is necessary to marginalize over the latent states.

$$p(\mathrm{y}_t \mid \mathrm{y}_{1:t-1}, \theta) = \int p(\mathrm{y}_t \mid \mathrm{x}_t, \theta)\, p(\mathrm{x}_t \mid \mathrm{y}_{1:t-1})\, d\mathrm{x}_t. \tag{2-19}$$

The second term $p(\mathrm{x}_t \mid \mathrm{y}_{1:t-1})$ does not have a closed-form solution, as it requires solving a nonlinear filtering problem. As a result, the corresponding integral must be approximated numerically, and neither the posterior nor the marginal likelihood can be evaluated in closed form. This challenge is illustrated in Figure 2-3. The inference task is to estimate the model parameters θ. While the transition and observation models are specified in terms of x, y and θ as shown in Figure 2-3. Therefore, the observations alone do not provide a complete specification of the underlying probabilistic model. Consequently, any parameter estimation algorithm must also account for the estimation of the latent state sequence.

The Bayesian formulation of the seabed mapping problem highlights the central challenge: the posterior distribution over the unknown parameters depends on latent states that cannot be observed directly, and the nonlinear measurement model prevents a closed-form solution. Consequently, computing either the MMSE or MAP estimates requires approximation methods. Chapter 4 reviews existing techniques that address this challenge, with a focus on Sequential Monte Carlo (SMC)-based approximations. In contrast, Chapter 3 adopts a different approach, obtaining an approximation by fixing a specific estimator structure which has a known closed form solution.

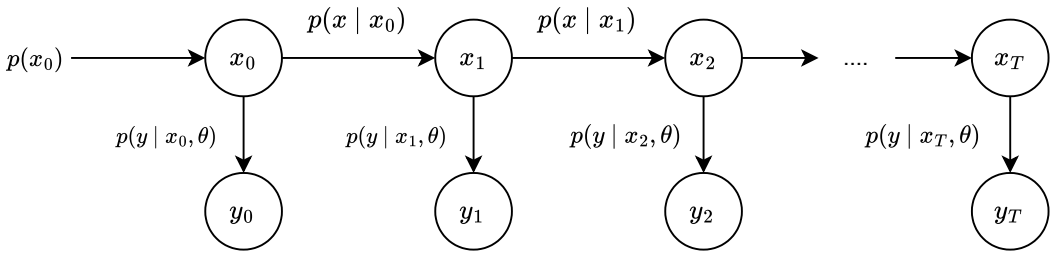

Figure 2-3: Latent state sequence drives the observation process

Chapter 3

Proposed Method

Dual Bayesian Affine MMSE Estimation

Chapter 2 introduced the Minimum Mean Square Error (MMSE) estimation problem (Equation (2-11)) for the stochastic Wiener model (Equation (2-3)). The goal of this chapter is to address the MMSE estimation problem by presenting the dual state-parameter estimator. The proposed method replaces the intractable MMSE estimator with the affine MMSE estimators. By restricting the estimator class to affine functions, a closed form solution for the restricted MMSE estimation problem is derived for both states and parameters. By alternating between these estimators in a fixed-point iteration, the method progressively reduces state-induced uncertainty.

Section 3-1 presents a high-level overview of the proposed estimator. In Section 3-3, a restricted MMSE estimator for the model parameters is presented. Section 3-4 develops a complementary affine MMSE estimator for the latent states. Finally, Section 3-5 integrates the parameter and state estimators into a unified nonlinear algorithm that iteratively refines both estimates.

3-1 High-Level Overview

As outlined in Section 2-4, the MMSE estimator requires marginalization over the unknown state trajectory. Marginalization introduces intractable state estimation integrals, which have to be approximated numerically. Typically, these integrals are approximated using Sequential Monte Carlo (SMC) techniques. While SMC methods are theoretically sound, they suffer from particle degeneracy and poor computational scaling. These problems make SMC methods impractical for estimation problems with large data records and many unknown parameters, like the bathymetric learning problem.

An alternative approach to circumvent the intractability of the MMSE estimator is to restrict the estimator class $\mathcal{F}_\theta$ to a set of estimators for which the optimal solution is known. A

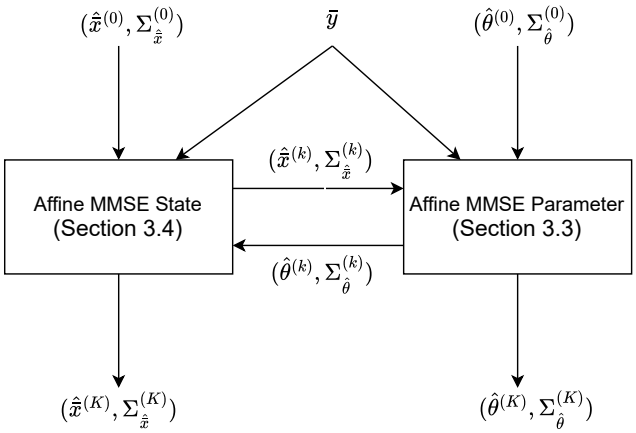

Figure 3-1: High Level overview of Dual estimation framework

common choice is the class of affine estimators, which yields the affine MMSE estimator. Under the restriction of an affine estimator, the optimization problem (Equation (2-11)) admits a closed-form solution that depends on the first- and second-order moments of the joint distribution $p(\bar{\mathrm{y}}, \theta)$ [27, Chapter 3]. For many nonlinear systems, computing the moments associated with $\bar{\mathrm{y}}$ analytically is not possible. However, the structure of the Wiener model allows these moments to be computed in closed form. Specifically, the linear dynamics of the process, together with the linear dependence of the output map on θ, allow these moments to be computed in closed form.

The quantities $\mu_{\bar{\mathrm{y}}}$, $\Sigma_{\bar{\mathrm{y}}}$ and $\Sigma_{\bar{\mathrm{y}}\theta}$ depend on how the stochastic state trajectory propagates through the basis functions of the measurement model. To compute these moments, the first and second-order moments of the basis functions, referred to as Dynamic Basis Statistics (DBS), must be known. By selecting a family of basis functions for which the DBS can be computed, like the Fourier basis, a closed-form solution for the affine MMSE estimator is available for the Wiener model [51].

The performance of the affine MMSE parameter estimator is tied to accurate estimation of the DBS. Hence, more precise knowledge of the states decreases the parameter estimation error. This work builds on this observation, by computing an estimate of the DBS using $\bar{\mathrm{y}}$. To this end, this chapter develops an affine MMSE estimator for the state variable, which yields an improved estimate of the DBS.

The key contribution of this chapter is the combination of the separate estimators into a dual state-parameter estimator. The dependency between the estimators is illustrated in Figure 3-1. The computation of the dual state-parameter estimator is realized using a fixed-point algorithm, in which estimation of the state and parameters is alternated until convergence.

The remainder of the chapter develops the machinery not shown in Figure 3-1. Starting with the definition of the DBS collection.

3-2 Dynamic Basis Statistics

The Wiener model (Equation (2-3)) is subject to additive Gaussian process noise, implying that the state of the system is a Gaussian random variable. Therefore, the observation model (Equation (2-1)) is a nonlinear transformation of a Gaussian random variable, and the resulting probability distribution $p(h_\theta(\mathrm{x}))$ is unknown. However, it is possible to compute the mean and covariance of this distribution, which will be referred to as the DBS collection.

The DBS captures that the system state is not observable and only its statistics are available. The statistics of the state are propagated through the nonlinear observation model via the basis functions. The DBS collection influences the mean and covariance of the observation model. For the Bayesian affine parameter estimator (Section 3-3), the DBS uniquely characterize the estimator. That is, given the statistical description of the Wiener model, the only quantities that must be computed in order to solve for the affine estimator are the DBS. The definition of the DBS collection from [51] is repeated here.

Definition 1 (**Dynamic basis statistics collection [51, Definition 1]**). *Let the state sequence* $\bar{\mathrm{x}} = [\mathrm{x}_0^\mathsf{T}, \dots, \mathrm{x}_T^\mathsf{T}]^\mathsf{T}$ *be a trajectory of* (2-3) *for* $t \in \{0, \dots, T\}$, *and let* $\Phi(\bar{\mathrm{x}}) = [\phi(\mathrm{x}_0), \dots, \phi(\mathrm{x}_T)]$. *The dynamic basis statistics collection with respect to the joint distribution of the state trajectory is defined as*

$$\mu_\Phi = [\mu_\phi^0, \dots, \mu_\phi^T], \qquad \Sigma_\Phi = \begin{bmatrix} \Sigma_\phi^{00} & \dots & \Sigma_\phi^{0T} \\ \vdots & \ddots & \vdots \\ \Sigma_\phi^{T0} & \dots & \Sigma_\phi^{TT} \end{bmatrix}, \tag{3-1}$$

where μ_ϕ^t *is the mean of* $\phi(\mathrm{x}_t)$ *and* $\Sigma_\phi^{tt'}$ *is the covariance between* $\phi(\mathrm{x}_t)$ *and* $\phi(\mathrm{x}_{t'})$, *given by*

$$\mu_\phi^t \coloneqq \mathbb{E}[\phi(\mathrm{x}_t)], \qquad \Sigma_\phi^{tt'} \coloneqq \mathbb{E}[\phi(\mathrm{x}_t)\phi^\mathsf{T}(\mathrm{x}_{t'})] - \mathbb{E}[\phi(\mathrm{x}_t)]\mathbb{E}[\phi(\mathrm{x}_{t'})]^\mathsf{T}. \tag{3-2}$$

For the Wiener model with Gaussian measurement noise and Fourier basis functions, a closed form solution for the DBS is available. To derive a compact formula, the lifted representation of the system is introduced. In particular, the entire trajectory of the model can be written as:

$$\bar{\mathrm{x}} = \bar{\mathrm{A}}(\bar{\mathrm{B}}\bar{\mathrm{u}} + \bar{\mathrm{w}}), \tag{3-3}$$

where $\bar{\mathrm{u}} = [\mu_{\mathrm{x}_0}^\mathsf{T}, \mathrm{u}_0^\mathsf{T}, \dots, \mathrm{u}_{T-1}^\mathsf{T}]^\mathsf{T}$ is the input vector with the initial state mean as its first element, $\bar{\mathrm{w}} = [\mathrm{w}_0^\mathsf{T}, \mathrm{w}_1^\mathsf{T}, \dots, \mathrm{w}_T^\mathsf{T}]^\mathsf{T}$ and, $\Sigma_{\bar{\mathrm{w}}} = \mathrm{diag}(\Sigma_{\mathrm{x}_0}, \Sigma_{\mathrm{w}_1}, \dots, \Sigma_{\mathrm{w}_T})$. The lifted matrices $\bar{\mathrm{A}}$ and $\bar{\mathrm{B}}$ encode the dynamics over the entire trajectory, they have the following structure:

$$\bar{\mathrm{A}} = \begin{bmatrix} \mathbb{I} & 0 & 0 & \dots & 0 \\ \mathrm{A}_0 & \mathbb{I} & 0 & \dots & \vdots \\ \mathrm{A}_1\mathrm{A}_0 & \mathrm{A}_1 & \mathbb{I} & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ \prod_{i=0}^{T-1} \mathrm{A}_i & \prod_{i=1}^{T-1} \mathrm{A}_i & \dots & \mathrm{A}_{T-1} & \mathbb{I} \end{bmatrix}, \qquad \bar{\mathrm{B}} = \mathrm{diag}(\mathbb{I}, \mathrm{B}_0, \dots, \mathrm{B}_{T-1}). \tag{3-4}$$

The lifted representation also provides a compact expression for the state of the system at time t, namely $\mathrm{x}_t = \bar{\mathrm{A}}_t(\bar{\mathrm{B}}\bar{\mathrm{u}}+\bar{\mathrm{w}})$. Where $\bar{\mathrm{A}}_t$ corresponds to the $(t+1)^{\text{th}}$ block-row of $\bar{\mathrm{A}}$ in (3-4), i.e.,

$$\bar{\mathrm{A}}_t = \begin{bmatrix} \prod_{i=0}^{t-1} \mathrm{A}_i & \prod_{i=1}^{t-1} \mathrm{A}_i & \dots & \mathbb{I} & 0 & \dots & 0 \end{bmatrix}. \tag{3-5}$$

Additionally, the lifted representation allows for a compact notation of the distribution of the system state. At time t the state is distributed according to $\mathrm{x}_t \sim \mathcal{N}(\bar{\mathrm{A}}_t\bar{\mathrm{B}}\bar{\mathrm{u}}, \bar{\mathrm{A}}_t\Sigma_{\bar{\mathrm{w}}}\bar{\mathrm{A}}_t^{\mathsf{T}})$. Using the lifted representation, [51] derives the DBS for the Fourier basis with Gaussian process noise. For completeness, the result is repeated here.

Lemma 3-2.1 (Fourier DBS explicit expression [51, Lemma 4.1]). *For Fourier basis functions* (2-2) *and the dynamics of* (2-3)*, let the respective DBS introduced in Definition 1, denoted by* $(\mu_\phi^t, \Sigma_\Phi^{tt'})$*. Additionally, let* $\bar{\mathrm{A}}_t$ *and* $\bar{\mathrm{A}}_{t'}$ *represent the matrices defined analogously to* (3-5)*. Then, the following holds:*

1. **DBS mean:** *The* n^{th} *element is*

$$\mu_{\phi_n}^t = \sum_{\ell\in\{-1,1\}} \exp(j\langle \bar{\mathrm{A}}_t^{\mathsf{T}}\ell f_n, \bar{\mathrm{B}}\bar{\mathrm{u}}\rangle)\exp\big(-\frac{1}{2}\ell f_n^{\mathsf{T}}\bar{\mathrm{A}}_t\Sigma_{\bar{\mathrm{w}}}\bar{\mathrm{A}}_t^{\mathsf{T}}\ell f_n\big), \quad n \geq 1. \tag{3-6a}$$

2. **DBS covariance:** *The* $(m,n)^{th}$ *element of the covariance matrix is*

$$\begin{aligned}\Sigma_{\phi_{mn}}^{tt'} = \sum_{\ell,\ell'\in\{-1,1\}} &\exp(j\langle \bar{\mathrm{A}}_t^{\mathsf{T}}\ell f_m + \bar{\mathrm{A}}_{t'}^{\mathsf{T}}\ell' f_n, \bar{\mathrm{B}}\bar{\mathrm{u}}\rangle) \\ &\times \big(\exp\big(-\frac{1}{2}(\ell f_m^{\mathsf{T}}\bar{\mathrm{A}}_t + \ell' f_n^{\mathsf{T}}\bar{\mathrm{A}}_{t'})\Sigma_{\bar{\mathrm{w}}}(\bar{\mathrm{A}}_t^{\mathsf{T}}\ell f_m + \bar{\mathrm{A}}_{t'}^{\mathsf{T}}\ell' f_n)\big) - \\ &\exp\big(-\frac{1}{2}\ell f_m^{\mathsf{T}}\bar{\mathrm{A}}_t\Sigma_{\bar{\mathrm{w}}}\bar{\mathrm{A}}_t^{\mathsf{T}}\ell f_m\big)\exp\big(-\frac{1}{2}\ell f_n^{\mathsf{T}}\bar{\mathrm{A}}_t\Sigma_{\bar{\mathrm{w}}}\bar{\mathrm{A}}_t^{\mathsf{T}}\ell f_n\big)\big).\end{aligned} \tag{3-6b}$$

Figure 3-2 shows a schematic representation of the DBS operator, a nonlinear mapping from mean and covariance of the state, into the mean and covariance of the basis functions, following Equations (3-6a) and (3-6b). In the next section, the DBS is used to derive a closed form solution for the affine MMSE parameter estimator.

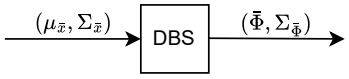

Figure 3-2: Block representation of the DBS operation

3-3 Affine MMSE Parameter Estimator

Building on the DBS, an MMSE estimator for the unknown parameters is constructed. By restricting the class of the estimator to be affine, a closed form solution for the estimator and estimator covariance is available by applying a well-known result from linear estimation theory. The resulting estimator can be viewed as function of the DBS collection. This

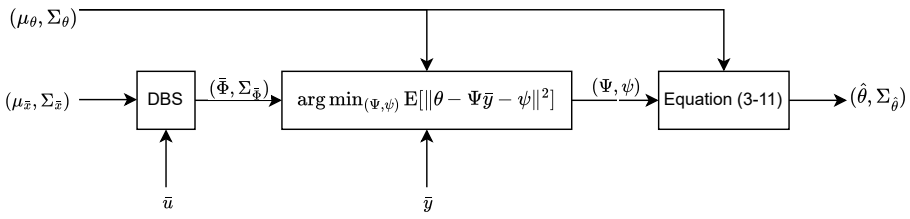

Figure 3-3: Block Representation of Proposed Parameter Estimate

section presents the mathematical formulation of the affine MMSE parameter estimator and its covariance.

Recall that the MMSE estimator is defined as:

$$\hat{\theta}(\bar{\mathrm{y}}) = \arg\min_{\nu(\cdot)\in\mathcal{F}_\theta} \mathbb{E}^{p(\theta|\bar{\mathrm{y}})}\big[\|\theta - \nu(\bar{\mathrm{y}})\|^2\big]. \tag{3-7}$$

where $\mathcal{F}_\theta = \left\{\hat{\theta}\colon \mathbb{R}^{T+1} \to \Theta\right\}$ denotes the set of all estimators. Directly solving (3-7) is intractable when the posterior is non-Gaussian, like for the Wiener model. To obtain a tractable approximation of the posterior mean, we restrict the estimator to the class of affine functions:

$$\mathcal{F}_\theta = \left\{ \hat{\theta}(\bar{\mathrm{y}}) = \Psi_\theta \bar{\mathrm{y}} + \psi_\theta \mid \Psi \in \mathbb{R}^{(\mathrm{N}+1)\times(T+1)},\, \psi \in \mathbb{R}^{\mathrm{N}+1} \right\}. \tag{3-8}$$

Note that this means that the optimization is now performed over decision variables $(\Psi_\theta, \psi_\theta)$, whereas the original optimization was performed over an abstract function space. The restriction of the optimization to (3-8), together with the closed-form expressions for the DBS, allows Equation (3-7) to be solved in closed form, as shown in [51, Thm. 3.2]. In particular, the optimal matrices are nonlinear functions of the DBS collection such that:

$$\Psi_\theta^\star(\mu_\Phi, \Sigma_\Phi) = \Sigma_\theta \mu_\Phi \big(\mu_\Phi^{\mathsf{T}} \Sigma_\theta \mu_\Phi + \mathcal{M}(\Sigma_\Phi) + \Sigma_{\bar{\mathrm{v}}}\big)^{-1}, \quad \psi_\theta^\star(\mu_\Phi, \Sigma_\Phi) = \mu_\theta - \Psi_\theta^\star(\mu_\Phi, \Sigma_\Phi)\mu_\Phi^{\mathsf{T}}\mu_\theta. \tag{3-9}$$

where $\mathcal{M}\colon \mathbb{R}^{((\mathrm{N}+1)(T+1))^2} \to \mathbb{R}^{(T+1)^2}$ is a linear operator whose $(t,t')^{\text{th}}$ component is

$$\mathcal{M}^{tt'}(\Sigma_\Phi) = \operatorname{tr}\big(\Sigma_\phi^{tt'}(\Sigma_\theta + \mu_\theta \mu_\theta^{\mathsf{T}})\big), \tag{3-10}$$

and $\Sigma_{\bar{\mathrm{v}}} = \operatorname{diag}(\sigma^2_{\mathrm{v}_0}, \ldots, \sigma^2_{\mathrm{v}_T})$ is the covariance matrix of the measurement noise sequence $\bar{\mathrm{v}} = [\mathrm{v}_0, \ldots, \mathrm{v}_T]^{\mathsf{T}}$.

Using the optimal matrices from Equation (3-9), the mean and covariance of the affine MMSE estimator are:

$$\begin{cases} \hat{\theta}(\bar{\mathrm{y}} \,|\, \mu_\Phi, \Sigma_\Phi) = \Psi_\theta^\star(\mu_\Phi, \Sigma_\Phi)\bar{\mathrm{y}} + \psi_\theta^\star(\mu_\Phi, \Sigma_\Phi), \\ \Sigma_{\hat{\theta}}(\mu_\Phi, \Sigma_\Phi) = \Sigma_\theta - \Sigma_\theta \mu_\Phi \big(\mu_\Phi^{\mathsf{T}} \Sigma_\theta \mu_\Phi + \mathcal{M}(\Sigma_\Phi) + \Sigma_{\bar{\mathrm{v}}}\big)^{-1} \mu_\Phi^{\mathsf{T}} \Sigma_\theta. \end{cases} \tag{3-11}$$

Figure 3-3 provides a schematic representation of the affine parameter estimator. The figure illustrates the estimator's inputs and outputs: given the mean and covariance statistics of the state trajectory, these statistics are propagated through the nonlinear basis functions via the DBS. The estimator then computes the affine MMSE parameter estimate and covariance. Similarly, an affine MMSE estimator can be computed for the latent state trajectory, which is discussed next.

3-4 Affine MMSE State Estimator

The previous section introduced the affine MMSE parameter estimator, which leverages the DBS to obtain a tractable estimate of the unknown parameters. In this section, we consider the dual problem of estimating the system state. This contrasts with the previous section, where the focus was on estimating the parameters.

As in the parameter estimation case, the state estimation problem is formulated using the MMSE criterion. Again, the estimator is restricted to the class of affine estimators. Under this restriction, the MMSE state estimator admits a closed-form solution. Importantly, in the dual setting, the state estimator depends on the mean and covariance of the parameters. In contrast to the previous section, where the parameter estimator was a function of the DBS collection, which are a nonlinear function of the state.

The affine MMSE state estimation problem is defined as:

$$\min_{\hat{\bar{\mathrm{x}}} \in X} \mathbb{E}\left[\left\|\bar{\mathrm{x}} - \hat{\bar{\mathrm{x}}}(\bar{\mathrm{y}})\right\|^2\right], \tag{3-12}$$

where,

$$X = \left\{ \hat{\bar{\mathrm{x}}}(\bar{\mathrm{y}}) = Z\bar{\mathrm{y}} + \zeta \;\middle|\; Z \in \mathbb{R}^{n_{\mathrm{x}}(T+1)\times(T+1)},\ \zeta \in \mathbb{R}^{n_{\mathrm{x}}(T+1)} \right\}, \tag{3-13}$$

denotes the class of affine state estimators for the latent state, based on the full measurement sequence.

Similar to [51, Thm. 3.2], the affine MMSE state estimation problem also has a closed form solution [52, Thm. 4.1]. The result is repeated here for ease of reference. The optimal matrices (Z, ζ) and are given by:

$$Z^\star(\mu_\theta, \Sigma_\theta) = \Sigma_{\bar{\mathrm{x}}\bar{\mathrm{y}}}\big(\mu_\Theta^{\mathsf{T}}\Sigma_\Phi\mu_\Theta + \mathcal{S}(\Sigma_\theta) + \Sigma_{\bar{\mathrm{v}}}\big)^{-1}, \quad \zeta^\star(\mu_\theta, \Sigma_\theta) = \bar{\mathrm{A}}\bar{\mathrm{B}}\bar{\mathrm{u}} - Z^\star(\mu_\theta, \Sigma_\theta)\mu_\Phi^{\mathsf{T}}\mu_\theta, \tag{3-14}$$

For the dual problem, the prior for the system state is $\mathcal{N}(\bar{\mathrm{A}}\bar{\mathrm{B}}\bar{\mathrm{u}},\ \bar{\mathrm{A}}\Sigma_{\bar{\mathrm{w}}}\bar{\mathrm{A}}^{\mathsf{T}})$, obtained using the lifted representation of the model. In Equation (3-14), (μ_Φ, Σ_Φ) denotes the DBS transformation (as defined in Definition 1) of the prior, i.e $(\mu_\Phi, \Sigma_\Phi) = \mathrm{DBS}(\bar{\mathrm{A}}\bar{\mathrm{B}}\bar{\mathrm{u}},\ \bar{\mathrm{A}}\Sigma_{\bar{\mathrm{w}}}\bar{\mathrm{A}}^{\mathsf{T}})$.

The function $\mathcal{S}\colon \mathbb{R}^{(\mathrm{N}+1)^2} \to \mathbb{R}^{(T+1)^2}$ is defined elementwise as

$$\mathcal{S}^{tt'}(\Sigma_\theta) = \mathrm{tr}\big(\Sigma_\theta(\Sigma_\phi^{tt'} + \mu_\phi^t \mu_\phi^{t'^{\mathsf{T}}})\big). \tag{3-15}$$

The matrix $\Sigma_{\bar{\mathrm{x}}\bar{\mathrm{y}}}$ is the cross-covariance between the latent state trajectory and the observations, and is given by

$$\Sigma_{\bar{\mathrm{x}}\bar{\mathrm{y}}} = \bar{\mathrm{A}}\Sigma_{\bar{\mathrm{w}}}\bar{\mathrm{A}}^{\mathsf{T}}\bar{\mathrm{C}}^{\mathsf{T}}\mu_\Theta, \quad \bar{\mathrm{C}} = \mathrm{diag}(\mathrm{C}_0, \ldots, \mathrm{C}_T), \quad \mathrm{C}_t = \mathbb{E}[\mathrm{J}_\phi(\mathrm{x}_t)], \tag{3-16}$$

where $\mu_\Theta = \mathrm{diag}(\mu_\theta^0, \ldots, \mu_\theta^T)$ with $\mu_\theta^t = \mu_\theta$ and

$$\mathrm{J}_\phi(\mathrm{x}_{t'}) = \left[\nabla_{\mathrm{x}}\phi_0(\mathrm{x}_{t'}), \ldots, \nabla_{\mathrm{x}}\phi_n(\mathrm{x}_{t'})\right]^{\mathsf{T}} \tag{3-17}$$

denotes the Jacobian of the basis functions evaluated at $\mathrm{x}_{t'}$, which has a closed form solution for the Fourier basis functions.

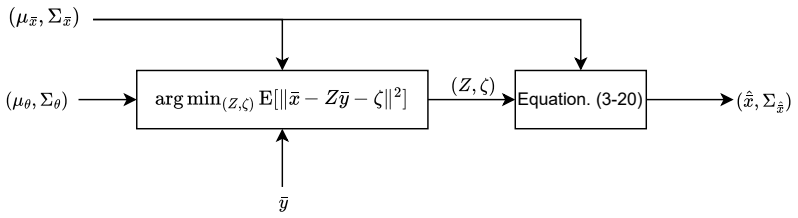

Figure 3-4: Block Representation of Proposed State Estimate

Using the optimal matrices from Equation (3-14), the mean and covariance of the affine MMSE estimator are:

$$\begin{cases} \hat{\mathrm{x}}_t(\bar{\mathrm{y}} \,|\, \mu_\theta, \Sigma_\theta) = \bar{\mathrm{A}}_t\big(\bar{\mathrm{B}}\bar{\mathrm{u}} + \Sigma_{\bar{\mathrm{w}}}\bar{\mathrm{A}}^{\mathsf{T}}\bar{\mathrm{C}}^{\mathsf{T}}\mu_\Theta(\mu_\Theta^{\mathsf{T}}\Sigma_\Phi\mu_\Theta + \mathcal{S}(\Sigma_\theta) + \Sigma_{\bar{\mathrm{v}}})^{-1}(\bar{\mathrm{y}} - \mu_\Phi^{\mathsf{T}}\mu_\theta)\big), \\ \Sigma_{\hat{\mathrm{x}}_t}(\mu_\theta, \Sigma_\theta) = \bar{\mathrm{A}}_t\big(\Sigma_{\bar{\mathrm{w}}} - \Sigma_{\bar{\mathrm{w}}}\bar{\mathrm{A}}^{\mathsf{T}}\bar{\mathrm{C}}^{\mathsf{T}}\mu_\Theta(\mu_\Theta^{\mathsf{T}}\Sigma_\Phi\mu_\Theta + \mathcal{S}(\Sigma_\theta) + \Sigma_{\bar{\mathrm{v}}})^{-1}\mu_\Theta^{\mathsf{T}}\bar{\mathrm{C}}\bar{\mathrm{A}}\Sigma_{\bar{\mathrm{w}}}\big)\bar{\mathrm{A}}_t^{\mathsf{T}}, \end{cases} \tag{3-18}$$

Figure 3-4 provides a schematic representation of the affine MMSE state estimator. The diagram highlights the dual dependence between state and parameter estimation, motivating their iterative coupling in the proposed dual estimation framework. In particular, the output of the state estimator is used as input to the parameter estimator, and vice versa.

3-5 Dual State-Parameter Estimator

Figures 3-4 and 3-3 illustrate that the output of any of the affine MMSE estimators can serve as input to the other the affine MMSE estimator. The dual state-parameter estimator essentially connects these two estimators. The interaction between the estimators for the dual state-parameter estimator is depicted schematically in Figure 3-5. The dual estimator can also be motivated from a performance perspective.

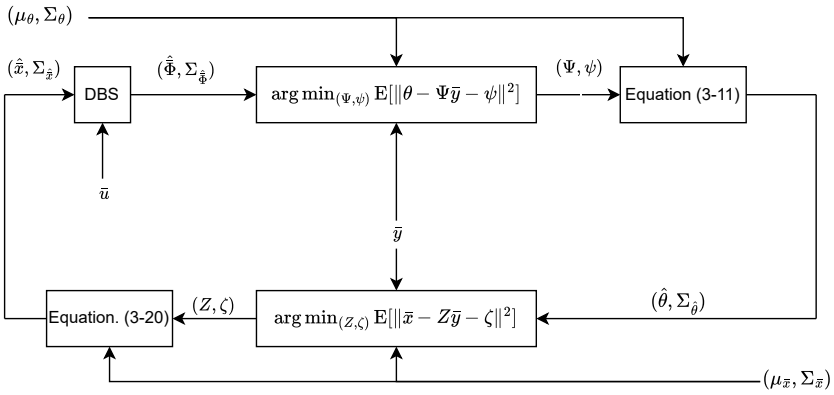

Figure 3-5: Block Representation of Proposed Dual Estimator

In the extreme case where the state trajectory is perfectly known, the only remaining uncertainty comes from measurement noise. In this case, the parameter estimation problem reduces to Bayesian linear regression. The affine MMSE parameter estimator can be shown

to be lower bounded by the estimator with perfect state knowledge, as formalized in the following Lemma:

Lemma 3-5.1 (**Lower bound on the MSE of the affine estimator [51, Lemma 2.1]**). *Let $\hat{\theta}(\bar{\mathrm{y}} \,|\, \mu_\Phi, \Sigma_\Phi)$ denote the affine MMSE estimator that uses only prior information about the system states, and let $\hat{\theta}(\bar{\mathrm{y}} \,|\, \Phi(\bar{\mathrm{x}}))$ be the affine MMSE estimator with complete knowledge of the states. The MSE of the marginalized estimator is then bounded below by the MSE achievable with full knowledge of the system states, i.e.,*

$$\mathbb{E}\big[\|\theta - \hat{\theta}(\bar{\mathrm{y}} \,|\, \Phi(\bar{\mathrm{x}}))\|^2\big] \leq \mathbb{E}\big[\|\theta - \hat{\theta}(\bar{\mathrm{y}} \,|\, \mu_\Phi, \Sigma_\Phi)\|^2\big].$$

Lemma 3-5.1 shows that improving knowledge of the state can improve the accuracy of the parameter estimate. Motivated by Lemma 3-5.1, we construct an estimate of the DBS, denoted $(\hat{\Phi}, \Sigma_{\hat{\Phi}})$ by taking the DBS transform of affine state estimate $(\hat{\bar{\mathrm{x}}}, \Sigma_{\hat{\bar{\mathrm{x}}}})$ computed using Equation (3-14). The estimated DBS is used to condition the affine MMSE parameter update, resulting in a nonlinear parameter estimator with improved performance:

$$\begin{cases} \hat{\theta}_{\mathrm{nl}}(\bar{\mathrm{y}} \,|\, \hat{\Phi}, \Sigma_{\hat{\Phi}}) = \Psi_\theta^\star(\hat{\Phi}, \Sigma_{\hat{\Phi}})\bar{\mathrm{y}} + \psi_\theta^\star(\hat{\Phi}, \Sigma_{\hat{\Phi}}), \\ \Sigma_{\hat{\theta}_{\mathrm{nl}}}(\hat{\Phi}, \Sigma_{\hat{\Phi}}) = \Sigma_\theta - \Sigma_\theta \hat{\Phi}\big(\hat{\Phi}^{\mathsf{T}} \Sigma_\theta \hat{\Phi} + \mathcal{M}(\Sigma_{\hat{\Phi}}) + \Sigma_{\bar{\mathrm{v}}}\big)^{-1} \hat{\Phi}^{\mathsf{T}} \Sigma_\theta. \end{cases} \tag{3-19}$$

Likewise, to further improve state estimation, we can replace the prior statistics $(\mu_\theta, \Sigma_\theta)$ with parameter estimates $(\hat{\theta}, \Sigma_{\hat{\theta}})$ in (3-14) to obtain the following nonlinear state estimator:

$$\begin{cases} \hat{\bar{\mathrm{x}}}_{\mathrm{nl}}(\bar{\mathrm{y}} \,|\, \hat{\theta}, \Sigma_{\hat{\theta}}) = Z^\star(\hat{\theta}, \Sigma_{\hat{\theta}})\bar{\mathrm{y}} + \zeta^\star(\hat{\theta}, \Sigma_{\hat{\theta}}), \\ \Sigma_{\hat{\bar{\mathrm{x}}}_{\mathrm{nl}}}(\hat{\theta}, \Sigma_{\hat{\theta}}) = \bar{\mathrm{A}} \Sigma_{\bar{\mathrm{w}}} \bar{\mathrm{A}}^{\mathsf{T}} - \bar{\mathrm{A}} \Sigma_{\bar{\mathrm{w}}} \bar{\mathrm{A}}^{\mathsf{T}} \bar{\mathrm{C}}^{\mathsf{T}} \hat{\theta} \big(\hat{\theta}^{\mathsf{T}} \Sigma_\Phi \hat{\theta} + \mathcal{S}(\Sigma_{\hat{\theta}}) + \Sigma_{\bar{\mathrm{v}}}\big)^{-1} \hat{\theta} \bar{\mathrm{C}} \bar{\mathrm{A}} \Sigma_{\bar{\mathrm{w}}} \bar{\mathrm{A}}^{\mathsf{T}}, \end{cases} \tag{3-20}$$

Using this nonlinear state estimator, we can construct the respective DBS collection $(\hat{\Phi}(\hat{\bar{\mathrm{x}}}_{\mathrm{nl}}, \Sigma_{\hat{\bar{\mathrm{x}}}_{\mathrm{nl}}}), \Sigma_{\hat{\Phi}}(\hat{\bar{\mathrm{x}}}_{\mathrm{nl}}, \Sigma_{\hat{\bar{\mathrm{x}}}_{\mathrm{nl}}}))$ according to Definition (3-1).

Algorithm 1 shows the computation of the dual state-parameter estimator, an algorithmic implementation of Figure 3-5. The algorithm initializes the parameter estimates from prior information and the DBS collection from Definition 1. It then alternates between updating the parameter estimates, their covariance, the state estimates, and the corresponding DBS statistics until the fixed-point convergence criterion is met.

Algorithm 1 is as a fixed-point iteration on the collection of estimates:

$$(\hat{\theta}_{\mathrm{nl}}, \Sigma_{\hat{\theta}_{\mathrm{nl}}}, \hat{\bar{\mathrm{x}}}_{\mathrm{nl}}, \Sigma_{\hat{\bar{\mathrm{x}}}_{\mathrm{nl}}}), \tag{3-21}$$

where each update applies the affine MMSE update of one subproblem while holding the remaining quantities fixed. A fixed point of the algorithm is therefore a set of statistics that is left unchanged by application of the affine parameter estimator and the the affine state estimator. The existence and uniqueness of fixed points are not established. Importantly, the dual state-parameter estimator computed by Algorithm 1 should not be interpreted as the exact MMSE estimator as their relation is unknown at the time of writing. In practice, convergence of Algorithm 1 is monitored evaluating the change in the parameter estimates, i.e. whether $\|\hat{\theta}_{\mathrm{nl}}^{(k)} - \hat{\theta}_{\mathrm{nl}}^{(k-1)}\| < \epsilon$ for some small threshold $\epsilon > 0$.

Algorithm 1 Dual state-parameter estimator

$(\hat{\theta}_{\text{nl}}^{(0)}, \Sigma_{\hat{\theta}_{\text{nl}}}^{(0)}) \leftarrow (\mu_\theta, \Sigma_\theta)$ from prior information

$(\hat{\Phi}^{(0)}, \Sigma_{\hat{\Phi}}^{(0)}) \leftarrow (\mu_\Phi, \Sigma_\Phi)$ from DBS collection Definition 1 according to (3-2.1)

for $k = 1, 2, \ldots$ until convergence **do**

$\mathcal{M}^{tt'}(\Sigma_{\hat{\Phi}}^{(k-1)}) \leftarrow \text{tr}(\Sigma_{\hat{\Phi}}^{(k-1)^{tt'}}(\Sigma_{\hat{\theta}_{\text{nl}}}^{(0)} + \hat{\theta}_{\text{nl}}^{(0)}\hat{\theta}_{\text{nl}}^{(0)^\mathsf{T}}))$

$\hat{\theta}_{\text{nl}}^{(k)} \leftarrow \hat{\theta}_{\text{nl}}^{(0)} + \Sigma_{\hat{\theta}_{\text{nl}}}^{(0)}\hat{\Phi}^{(k-1)}(\hat{\Phi}^{(k-1)^\mathsf{T}}\Sigma_{\hat{\theta}_{\text{nl}}}^{(0)}\hat{\Phi}^{(k-1)} + \mathcal{M}(\Sigma_{\hat{\Phi}}^{(k-1)}) + \Sigma_{\bar{\text{v}}})^{-1}(\bar{\text{y}} - \hat{\Phi}^{(k-1)^\mathsf{T}}\hat{\theta}_{\text{nl}}^{(0)})$

$\Sigma_{\hat{\theta}_{\text{nl}}}^{(k)} \leftarrow \Sigma_{\hat{\theta}_{\text{nl}}}^{(0)} - \Sigma_{\hat{\theta}_{\text{nl}}}^{(0)}\hat{\Phi}^{(k-1)}(\hat{\Phi}^{(k-1)^\mathsf{T}}\Sigma_{\hat{\theta}_{\text{nl}}}^{(0)}\hat{\Phi}^{(k-1)} + \mathcal{M}(\Sigma_{\hat{\Phi}}^{(k-1)}) + \Sigma_{\bar{\text{v}}})^{-1}\hat{\Phi}^{(k-1)^\mathsf{T}}\Sigma_{\hat{\theta}_{\text{nl}}}^{(0)}$

$\mathcal{S}^{tt'}(\Sigma_{\hat{\theta}_{\text{nl}}}^{(k-1)}) \leftarrow \text{tr}(\Sigma_{\hat{\theta}_{\text{nl}}}^{(k-1)}(\Sigma_{\hat{\Phi}}^{(0)^{tt'}} + \hat{\Phi}^{(0)^t}\hat{\Phi}^{(0)^{t'^\mathsf{T}}}))$

$\hat{\mu}_{\Theta_{\text{nl}}}^{(k-1)} \leftarrow \text{diag}(\hat{\theta}_{\text{nl}}^{(k-1)^0}, \ldots, \hat{\theta}_{\text{nl}}^{(k-1)^T})$ with $\hat{\theta}_{\text{nl}}^{(k-1)^t} = \hat{\theta}_{\text{nl}}^{(k-1)}$, for each $t = \{0, \ldots, T\}$

$\hat{\bar{\text{x}}}_{\text{nl}}^{(k)} \leftarrow \bar{\text{A}}\bar{\text{B}}\bar{\text{u}} + \bar{\text{A}}\Sigma_{\bar{\text{w}}}\bar{\text{A}}^\mathsf{T}\bar{\text{C}}^\mathsf{T}\hat{\mu}_{\Theta_{\text{nl}}}^{(k-1)}(\hat{\mu}_{\Theta_{\text{nl}}}^{(k-1)^\mathsf{T}}\Sigma_{\hat{\Phi}}^{(0)}\hat{\mu}_{\Theta_{\text{nl}}}^{(k-1)} + \mathcal{S}(\Sigma_{\hat{\theta}_{\text{nl}}}^{(k-1)}) + \Sigma_{\bar{\text{v}}})^{-1}(\bar{\text{y}} - \hat{\Phi}^{(0)^\mathsf{T}}\hat{\theta}_{\text{nl}}^{(k-1)})$

$\Sigma_{\hat{\bar{\text{x}}}_{\text{nl}}}^{(k)} \leftarrow \bar{\text{A}}\Sigma_{\bar{\text{w}}}\bar{\text{A}}^\mathsf{T} - \bar{\text{A}}\Sigma_{\bar{\text{w}}}\bar{\text{A}}^\mathsf{T}\bar{\text{C}}^\mathsf{T}\hat{\mu}_{\Theta_{\text{nl}}}^{(k-1)}(\hat{\mu}_{\Theta_{\text{nl}}}^{(k-1)^\mathsf{T}}\Sigma_{\hat{\Phi}}^{(0)}\hat{\mu}_{\Theta_{\text{nl}}}^{(k-1)} + \mathcal{S}(\Sigma_{\hat{\theta}_{\text{nl}}}^{(k-1)}) + \Sigma_{\bar{\text{v}}})^{-1}\hat{\mu}_{\Theta_{\text{nl}}}^{(k-1)^\mathsf{T}}\bar{\text{C}}\bar{\text{A}}\Sigma_{\bar{\text{w}}}\bar{\text{A}}^\mathsf{T}$

$\hat{\Phi}^{(k)} \leftarrow \hat{\Phi}(\hat{\bar{\text{x}}}_{\text{nl}}^{(k)}, \Sigma_{\hat{\bar{\text{x}}}_{\text{nl}}}^{(k)})$, $\Sigma_{\hat{\Phi}}^{(k)} \leftarrow \Sigma_{\hat{\Phi}}(\hat{\bar{\text{x}}}_{\text{nl}}^{(k)}, \Sigma_{\hat{\bar{\text{x}}}_{\text{nl}}}^{(k)})$ from DBS collection according to (3-2.1)

end for

Chapter Summary

In summary, this chapter has presented the dual state-parameter estimator, which alternates between affine estimators for the latent states and model parameters to capture their mutual dependence. To evaluate the proposed method and position it within the existing literature, the next chapter discusses the existing approaches for computing the MAP and MMSE estimates, highlighting the differences in methodology and computational strategies.

Chapter 4

Relation to Existing Work

The previous chapter introduced the dual state-parameter estimator for jointly estimating the parameters and latent states of a stochastic Wiener model. The proposed method combines affine Minimum Mean Square Error (MMSE) estimators for the states and parameters into nonlinear estimators able to capture more complex patterns.

This chapter positions the proposed estimator within the broader literature on Bayesian inference. In particular, it clarifies how the dual affine estimator relates to existing data augmentation–based approaches and motivates the selection of benchmark algorithms used for comparison in Chapter 5.

A central theme in this comparison is the treatment of latent states as auxiliary variables, a strategy that underlies many estimation techniques. This viewpoint reveals a common alternating structure shared by methods such as Expectation Maximization (EM) and Gibbs sampling. Despite this structural similarity, these approaches differ fundamentally in their inference objectives and in how they address the intractable state estimation problem induced by the nonlinear measurement model.

The remainder of the chapter is organized as follows. Section 4-1 provides a high-level comparison between the proposed method and related data augmentation–based approaches. Section 4-2 describes Particle Gibbs (PG) sampling, which serves as a sampling-based benchmark for MMSE estimation. Finally, Section 4-3 presents the Particle Smoothing EM (PSEM) algorithm, which is used as an optimization-based benchmark for Maximum a Posteriori (MAP) estimation.

4-1 Data Augmentation Algorithms

One of the defining characteristics of Algorithm 1 is the treatment of unknown states as auxiliary variables, which are estimated jointly with the unknown parameters. This approach, commonly referred to as data augmentation [50], is well-known in many estimation methods. By introducing auxiliary variables, data augmentation allows complex joint estimation

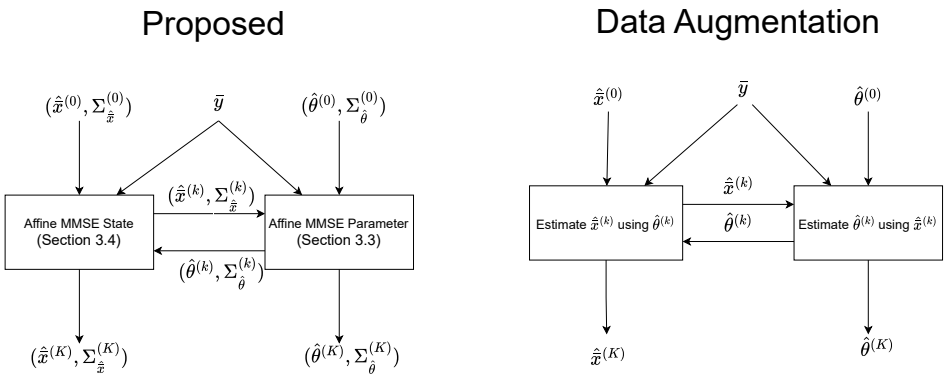

Figure 4-1: Proposed method vs General structure of decoupling estimation algorithms

problems to be broken into simpler subproblems. In state-space models, data augmentation introduces a strong coupling between the decision variables in the augmented optimization problem. A standard strategy to address this is to optimize one variable while keeping the other fixed, alternating between them [53]. Typically, this results in a two-step structure: first, estimating the states conditioned on a parameter estimate, and second, estimating the parameters given these states. This is displayed in Figure 4-1, note the similarity to Figure 3-1.

This decoupling is evident in several classical algorithms. In the EM algorithm [38, 24], the E-step estimates the latent states, while the M-step optimizes the parameters based on these states (see Section 4-3 for details). Similarly, Gibbs samplers [23, 12] alternate between sampling states and parameters (see Section 4-2). The dual smoothing framework of [53] also follows the data augmentation strategy, and provides good overview of different cost joint cost functions. All three algorithmic frameworks rely on the data augmentation principle and follow a similar alternating structure highlighted by Figure 4-1. In summary, these data augmentation methods share an alternating structure for estimating states and parameters. The key difference lies in how they handle state estimation. Understanding this distinction clarifies where the proposed dual affine estimator fits among existing approaches.

A key characteristic of data augmentation methods is that some subproblems remain inherently intractable. Specifically, even when conditioning on a parameter estimate, the associated smoothing or filtering distributions do not have closed-form solutions. This arises from the nonlinear measurement model, which makes the state estimation step a challenging nonlinear filtering or smoothing problem. Each framework must address the nonlinear state estimation problem in some form. In the proposed method, the smoothing distribution is approximated using an affine MMSE estimate, which tries to approximate the true mean of the smoothing distribution.

In both the EM and dual filtering frameworks, analogous strategies are employed to address the intractability of the state estimation step. Typically, the smoothing distribution is approximated using techniques such as the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) [24, 41]. The UKF and EKF rely on local linearization combined with a variant of the Kalman filter, which makes them efficient but also inaccurate [31]. By explicitly mapping the statistics of the latent state through the nonlinear basis functions, rather

than relying on linearization, the proposed method can capture the system statistics more accurately.

Alternatively, in the EM framework, variational approximations of the smoothing distribution can be used [4]. All three methods, UKF, EKF and variational approximation share a common principle with the proposed method: the true smoothing distribution is replaced with an approximation that admits a computationally efficient closed-form solution. In contrast, Gibbs samplers cannot rely on approximate smoothing distributions, as doing so would violate their asymptotic guarantees [46, Chapter 10].

Instead, they require a sampling procedure that asymptotically solves the original integration problem. Sequential Monte Carlo (SMC) methods are used, as their solutions converge to the smoothing/filtering distribution when the number of particles approaches infinity. When integrated into a Gibbs sampler, this approach is referred to as PG [15]. SMC methods have also been employed within the EM framework [47], where they are used to approximate the intractable smoothing integrals encountered during the E-step. Unlike purely approximate solutions, SMC methods offer a tunable trade-off between computational cost and accuracy.

Despite their similarity due to data augmentation, existing methods differ fundamentally in their underlying inference paradigms. EM and dual smoothing approaches are fundamentally optimization-based and typically target MAP estimates. Although EM, dual smoothing, and Gibbs sampling all rely on data augmentation, they differ in their inference goals. EM and dual smoothing are optimization-based methods that typically aim to find MAP estimates. In contrast, Gibbs sampling is a stochastic, sampling-based method designed to approximate the MMSE estimate. The proposed estimator targets the MMSE estimate but belongs to the class of optimization-based methods. It differs from classical EM approaches in how intractable state estimation integrals are handled. Rather than relying on particle smoothers or local linearization, the method replaces the exact smoothing distribution with a closed-form affine MMSE approximation.

To obtain a balanced evaluation, we therefore consider both an optimization-based and a sampling-based benchmark. PSEM is used as a reference for MAP estimation, as PSEM is known to provide highly accurate approximations of the E-step in nonlinear state-space models. For MMSE estimation, a variant of PG is employed. Next, both benchmark algorithms are described in detail.

4-2 Particle Gibbs Sampling

PG sampling [1] is a Monte Carlo method for drawing samples from the joint posterior $p(\bar{\mathrm{x}}, \theta \mid \bar{\mathrm{y}})$ of nonlinear state-space models. As the number of samples increases, the sample mean of the generated samples converges to the posterior mean. Formally, the output of the PG sampling is a sequence of K samples, i.e., $\{\theta^i\}_{i=1}^K$ The first L samples of the sequence are discarded, which is referred to as the burn-in. The posterior mean is approximated using the remaining samples:

$$\hat{\theta}_K(\bar{\mathrm{y}}) = \frac{1}{K-(L+1)} \sum_{i=L+1}^{K} \theta^{(i)}. \tag{4-1}$$

The PG sampler extends the Gibbs sampler to model classes for which direct sampling from the smoothing distribution $p(\bar{\mathrm{x}} \mid \theta, \bar{\mathrm{y}})$ is infeasible, as in the Wiener model. In such cases,

the Gibbs update of the latent state trajectory cannot be carried out using conventional sampling algorithms. PG addresses this issue by replacing the intractable sampling problem, by an update with a sample drawn from particle smoother. Importantly, even though the PG sampler uses an approximate particle smoother, the Markov kernel it defines leaves the exact joint posterior invariant, ensuring asymptotic correctness. First, the standard Gibbs sampler is reviewed which is extended to the PG sampler.

Following the data augmentation principle introduced in Section 4-1, the Gibbs sampler treats the latent states as auxiliary variables, allowing the joint inference problem to be decomposed into two subproblems. Specifically, the sampler alternates between two steps. First, the latent states are sampled conditional on the current parameter estimate, $p(\bar{\mathrm{x}} \mid \bar{\mathrm{y}}, \theta)$. Second, the parameters are updated conditional on the newly sampled states, $p(\theta \mid \bar{\mathrm{y}}, \bar{\mathrm{x}})$. This procedure is summarized in Algorithm 2.

Algorithm 2 Gibbs Sampler

Ensure: $\hat{\theta}_K$

Require: K and L

1: $\theta^{(0)} \sim p(\theta)$

2: $\bar{\mathrm{x}}^{(0)} \sim p(\bar{\mathrm{x}} \mid \theta^{(0)}, \bar{\mathrm{y}})$

3: **for** $k = 1 \ldots K$ **do**

4: $\quad \theta^{(k)} \sim p(\theta \mid \bar{\mathrm{x}}^{(k-1)}, \bar{\mathrm{y}})$

5: $\quad \bar{\mathrm{x}}^{(k)} \sim p(\bar{\mathrm{x}} \mid \theta^{(k)}, \bar{\mathrm{y}})$

6: **end for**

For the description provided in Section 2-1, step 2 requires sampling from a truncated multivariate normal distribution. Although this is a nontrivial task, it can be performed efficiently using algorithms such as those in [32]. In the case of the Wiener model, step 3 cannot rely on standard exact sampling methods, because the conditional distribution of the latent states is not available in closed form.

To overcome this limitation, [1] proposes using samples obtained from a Conditional Particle Filter (CPF) (see Appendix A-2 for details). The CPF replaces the intractable sampling of $p(\bar{\mathrm{x}}^{(k)} \mid \bar{\mathrm{y}}, \theta^k)$ with sampling from a particle approximation:

$$\hat{p}(\bar{\mathrm{x}}^{(k)} \mid \bar{\mathrm{y}}, \theta^{(k-1)}, \bar{\mathrm{x}}^{(k-1)}) = \sum_{i=1}^{N_f} \bar{\nu}^i \, \delta_{\bar{\mathrm{x}}^i}(\bar{\mathrm{x}}), \tag{4-2}$$

where $\{\bar{\mathrm{x}}^i, \bar{\nu}^i\}_{i=1}^{N_f}$ are particles and associated normalized weights generated by Algorithm 8. Intuitively, this approximation represents the filtering distribution as a discrete set of weighted trajectories: particles with higher weights correspond to regions of higher posterior probability. The particle approximation introduces a tuning parameter N_f, the number of particles, which directly controls the accuracy of the approximation: larger N_f leads to a closer representation of the true filtering distribution, at the cost of increased computational burden.

Unlike standard particle filters where all particles compete equally for survival, CPF ensures that a reference trajectory $\bar{\mathrm{x}}^{(k-1)}$ from the previous Gibbs iteration is retained throughout the filtering process. The remaining $N_f - 1$ particles evolve according to the Bootstrap Particle Filter (BPF) (See Appendix A-1 Algorithm 7 for details). The computational complexity of the CPF is $\mathcal{O}(N_f T)$, where T is the length of the data record. This is because each of the N_f particles must be propagated and reweighted at every time step.

A fundamental challenge in any SMC method is particle degeneracy. After several filtering steps, the effective sample size can decrease dramatically: most particle weights become negligible, leaving only a few particles with significant influence. This effectively reduces the diversity of the particle approximation, as many particles represent redundant copies of the same trajectory. In the context of PG, degeneracy is particularly problematic because it limits the ability of the sampler to explore the posterior distribution. If the CPF produces nearly identical trajectories across iterations, the Markov chain exhibits poor mixing and slow convergence.

Algorithm 3 displays the Particle Gibbs procedure, where a conditional particle filter with N_f particles is embedded within a Gibbs sampler to enable approximate sampling of latent states for the Wiener model.

Algorithm 3 Particle Gibbs Sampler

Ensure: $\hat{\theta}_K$

Require: K, L, N_f

1: $\theta^{(0)} \sim p(\theta)$

2: $\bar{\mathrm{x}}^{(0)} \sim \mathrm{BPF}(\bar{\mathrm{x}} \mid \theta^{(0)}, \bar{\mathrm{y}})$ according to Algorithm 7

3: **for** $k = 1 \ldots K$ **do**

4: $\quad \theta^{(k)} \sim p(\theta \mid \bar{\mathrm{x}}^{(k-1)}, \bar{\mathrm{y}})$ according to [32]

5: $\quad \bar{\mathrm{x}}^{(k)} \sim \mathrm{CPF}(\bar{\mathrm{x}} \mid \theta^{(k)}, \bar{\mathrm{y}}, \bar{\mathrm{x}}^{(k-1)})$ according to Algorithm 8

6: **end for**

7: $\hat{\theta}_K = \frac{1}{K-(L+1)} \sum_{i=L+1}^{K} \theta^{(i)}$

In terms of convergence, Particle Gibbs is a particle MCMC method whose behavior is governed by the convergence properties of the underlying Markov chain. From a theoretical standpoint, MCMC methods are asymptotically exact and converge to the true posterior distribution under mild regularity conditions [15]. In practice, however, verifying convergence of a Markov chain is notoriously difficult [14]. Moreover, the convergence of Particle Gibbs can be severely degraded by path degeneracy in the conditional particle filter, which leads to poor mixing of the Markov chain [34, 1]. As a consequence, reliable and objective convergence diagnostics are generally unavailable, and successful application of the method requires careful tuning.

4-3 Particle Smoothing EM

Whereas PG is a sampling based augmentation approach used to compute the MMSE estimate, the EM algorithm is an optimization based approach to compute the MAP estimate. The EM algorithm is an iterative method that maximizes the log posterior (Equation (2-16)) by optimizing a surrogate objective $Q(\theta, \theta')$ whose maximization implicitly increases the observed-data likelihood $p(\theta \mid \bar{\mathrm{y}})$.

From a data augmentation perspective, EM treats the latent states as auxiliary variables and alternates between two steps. In the E-step, the auxiliary function is computed as the expected complete-data log-likelihood with respect to the latent states, given the observations and current parameter estimate. In the M-step, the parameters are updated by maximizing this auxiliary function while keeping the state distribution fixed, iterating until convergence.

The auxiliary function is defined as the conditional expectation of the complete-data log-likelihood, conditioned on the observations $\bar{\mathrm{y}}$ and the parameter realization θ':

$$Q(\theta, \theta') = \mathbb{E}\big[\log p(\bar{\mathrm{x}}, \bar{\mathrm{y}} \mid \theta) \,\big|\, \bar{\mathrm{y}}, \theta'\big] = \int \log p(\bar{\mathrm{x}}, \bar{\mathrm{y}} \mid \theta)\, p(\bar{\mathrm{x}} \mid \bar{\mathrm{y}}, \theta')\, d\bar{\mathrm{x}}. \tag{4-3}$$

Importantly, $Q(\theta, \theta')$ is a function of the unknown parameters θ, while being conditioned on a fixed value of the parameters θ'. The auxiliary function can be decomposed into three terms corresponding to different model components.

$$Q(\theta, \theta') = I_1(\theta') + I_2(\theta') + I_3(\theta, \theta'), \tag{4-4}$$

with

$$I_1(\theta') = \int \log p(\mathrm{x}_0)\; p(\mathrm{x}_0 \mid \bar{\mathrm{y}}, \theta')\, d\mathrm{x}_0, \tag{4-5}$$

$$I_2(\theta') = \sum_{t=0}^{T-1} \iint \log p(\mathrm{x}_{t+1} \mid \mathrm{x}_t)\; p(\mathrm{x}_t, \mathrm{x}_{t+1} \mid \bar{\mathrm{y}}, \theta')\, d\mathrm{x}_t\, d\mathrm{x}_{t+1}, \tag{4-6}$$

$$I_3(\theta, \theta') = \sum_{t=0}^{T} \int \log p(\mathrm{y}_t \mid \mathrm{x}_t, \theta')\; p(\mathrm{x}_t \mid \bar{\mathrm{y}}, \theta')\, d\mathrm{x}_t. \tag{4-7}$$

Specifically, the terms I_1 and I_2 arise from the state transition density, capturing the contribution of the system dynamics, whereas I_3 originates from the observation model, representing the likelihood contribution of the measurements.

Equation 2-16 shows that the auxiliary function $Q(\theta, \theta')$ approximates the first term of the log-posterior (reference with Equation 2-16). The remaining step is to add the log-prior. The auxiliary MAP objective is therefore:

$$R(\theta, \theta') = Q(\theta, \theta') + \log p(\theta) \tag{4-8}$$

In the M-step, the parameters are updated by maximizing Equation (4-8):

$$M(\theta') = \arg\max_{\theta}\; R(\theta, \theta') = \arg\max_{\theta} I_3(\theta, \theta') + \log p(\theta). \tag{4-9}$$

Since the process model is known, only I_3 depends on the unknown parameters and I_1 and I_2 can be dropped from the optimization. The optimization in Equation (4-9) can be simplified even further by using the fact that the prior comes from the uniform distribution.

The uniform prior restricts each parameter to a bounded interval (recall Equation (2-6)). Specifically, each parameter must satisfy two linear inequality constraints corresponding to the lower and upper bounds of the uniform distribution. Namely:

$$a \leq \theta_i \leq b, \quad i = 0, \ldots, N.$$

Since there are N parameters, this results in a total of $2N$ linear inequality constraints. These constraints are incorporated into the M-step. The M-step is rewritten as the following constrained convex optimization problem:

$$M(\theta') = \arg\max_{\theta} \; I_3(\theta, \theta') \quad \text{subject to} \quad a \leq \theta_i \leq b, \quad i = 1, \ldots, N. \tag{4-10}$$

The optimization problem in (4-10) can be efficiently solved using standard numerical optimization techniques, such as interior-point methods. Together, Equation (4-8) and (4-10) define the backbone of the EM algorithm. The EM procedure for computing the MAP estimate of the Wiener model is shown in Algorithm 4.

Algorithm 4 Expectation-Maximization for MAP estimation

1: $\hat{\theta}_0 \leftarrow \mu_\theta$
2: **for** $k = 1, \ldots$ until converged **do**
3: $\quad I_3(\theta, \hat{\theta}_k) = \sum_{t=0}^{T} \int \log p(\mathrm{y}_t \mid \mathrm{x}_t, \theta')\, p(\mathrm{x}_t \mid \bar{\mathrm{y}}, \theta')\, d\mathrm{x}_t$
4: $\quad \hat{\theta}_{k+1} = \arg\max_\theta I_3(\theta, \hat{\theta}_k) \quad$ subject to $a \leq \theta_i \leq b, \quad i = 0, \ldots, N.$
5: **end for**

Equation (4-10) shows that the simplified objective corresponds to maximizing the expected likelihood of the observations under the current parameter estimate. Step 3 in Algorithm 4 does not have a closed-form solution and must be approximated numerically. To overcome this intractability, the integral is approximated numerically using an SMC method. The resulting class of EM algorithms is known as PSEM, and was first proposed by [47].

Following [47], this distribution can be approximated using a particle smoother, which is detailed in Appendix A-3. Similar to particle filtering, particle smoothing represents the smoothing distribution as a set of N_s weighted trajectories:

$$\hat{p}(\mathrm{x}_t \mid \bar{\mathrm{y}}) = \sum_{i=1}^{N_s} \bar{\nu}_t^i \, \delta_{\mathrm{x}_t^i}(\mathrm{x}_t), \tag{4-11}$$

where x_t^i denotes the i-th particle at time t and $\bar{\nu}_t^i$ is its normalized weight. The number of particles N_s directly controls the accuracy of the approximation: larger N_s provides a closer representation of the true smoothing distribution but increases computational cost.

As discussed in the previous section, particle degeneracy is a drawback for any SMC method, including the Forward Filtering Backward Simulation (FFBS) smoother. Degeneracy reduces the diversity of the particle set and causes the accuracy of the E-step approximation to collapse. Consequently, the M-step may produce suboptimal parameter updates, slowing convergence or leading to inaccurate MAP estimates. Degeneracy is particularly problematic

for long time series or highly nonlinear models, making careful selection of the number of particles N_s and the use of efficient smoothing techniques critical.

To approximate $I_3(\theta, \theta')$, the particle representation is substituted into Equation (4-3), yielding:

$$I_3(\theta, \theta') \approx \hat{I}_3(\theta, \theta') = \sum_{t=0}^{T} \sum_{i=1}^{N_s} \bar{\nu}_t^i \log p(\mathrm{y}_t \mid \mathrm{x}_t^i, \theta). \tag{4-12}$$

The M-step then becomes a constrained optimization over this approximate objective:

$$\hat{M}(\theta') = \arg\max_{\theta} \hat{I}_3(\theta, \theta') \quad \text{subject to} \quad a \leq \theta_i \leq b, \quad i = 1, \ldots, N. \tag{4-13}$$

Different particle smoothing algorithms can be employed to obtain $\hat{p}(\mathrm{x}_t \mid \bar{\mathrm{y}})$, such as the FFBS smoother of [25]. Since naive implementations of some smoothers incur $\mathcal{O}(N_s N_f T)$ cost, we adopt the efficient variant proposed in [6, Figure 5], which scales linearly with N_s and can handle large numbers of particles while mitigating degeneracy. The resulting particle smoothing EM (PSEM) algorithm is summarized in Algorithm 5.

Algorithm 5 Particle Smoothing EM for MAP estimation

Require: N_f, N_s, K

Ensure: $\hat{\theta}_K$

1: $\hat{\theta}_0 \leftarrow \mu_\theta$

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

3: $\{(\mathrm{x}_t^i, \bar{\nu}_t^i)_{i=1}^{N_f}\}_{t=0}^{T} \leftarrow$ BPF (Algorithm 7)

4: $\{(\mathrm{x}_t^i, \bar{\nu}_t^i)_{i=1}^{N_s}\}_{t=0}^{T} \leftarrow$ FFBS (Algorithm 9) using $\{(\mathrm{x}_t^i, \bar{\nu}_t^i)_{i=1}^{N_f}\}_{t=0}^{T}$

5: $\hat{I}_3(\theta, \hat{\theta}_k) = \sum_{t=0}^{T} \sum_{i=1}^{N_s} \bar{\nu}_t^i \log p(\mathrm{y}_t \mid \mathrm{x}_t^i, \theta)$

6: $\hat{\theta}_{k+1} = \arg\max_\theta R(\theta, \hat{\theta}_k)$ subject to $a \leq \theta_i \leq b, \quad i = 0, \ldots, N.$

7: **end for**

In terms of convergence, the EM algorithm is known to exhibit slow convergence. Under exact computation of the E-step, EM guarantees a monotonic increase of the observed-data log-posterior and convergence to a stationary point, whose optimality (local versus global) depends on the initialization [57]. For nonlinear state-space models, the likelihood surface is often highly non-convex and irregular, making EM particularly susceptible to convergence to poor local optima [14, Chapter 14.2]. In the PSEM algorithm, the E-step is approximated using a finite number of particles, and the monotonicity property no longer holds. As a result, classical EM convergence guarantees do not apply. Nevertheless, for sufficiently large particle sets, PSEM is known to provide accurate approximations of the E-step and has been observed to converge reliably in practice, albeit slowly [47].

4-4 Chapter Conclusion

This chapter positioned the dual state-parameter estimator within the landscape of data augmentation algorithms. While such methods share a common alternating structure, they differ

fundamentally in their inference objectives and in how they handle the intractable state estimation problem. PSEM targets MAP estimates through optimization, whereas PG targets MMSE estimation via sampling. In contrast, the proposed dual state-parameter estimator bridges these approaches by approximating the MMSE estimate using an optimization-based framework, while also accounting for the uncertainty in the augmented state variables, something the other methods cannot exploit. Building on this analysis, the next chapter evaluates the performance of these methods numerically, comparing their accuracy in a Monte Carlo experiment.

Chapter 5

Evaluation

The objective of this chapter is twofold. First, it evaluates the performance of the dual state-parameter framework developed in Chapter 3 and compares it with two data augmentation methods discussed in Chapter 4. The comparison is carried out through a Monte Carlo study comprising 10,000 distinct simulations. The results demonstrate that, for these configurations, the proposed method outperforms existing approaches. The results of the Monte Carlo experiment are presented in Section 5-1.

Based on these results, a high-fidelity simulation study is conducted to assess the practical applicability of the proposed dual estimation algorithm. In this experiment, a realistic physical model of an Autonomous Underwater Vehicle (AUV) is simulated, including sensor measurements and a controlled seabed model. The aim is to evaluate whether the proposed method shows sufficient promise for real-world AUV applications. This experiment is presented in Section 5-2.

5-1 Monte Carlo Experiment

This experiment has a similar configuration as the experiment in [51]. The objective of this experiment is to compare the proposed dual state-parameter estimator (Algorithm 1) with the Particle Smoothing EM (PSEM) method (Algorithm 5) [47] and the Particle Gibbs with Ancestor Sampling (PGAS) method (Algorithm 3) [34] through a Monte Carlo study. The PGAS algorithm is an extension of the Particle Gibbs (PG) algorithm presented in Chapter 4-2 which also performs an ancestor sampling step for the reference trajectory [34].

The comparison investigates how accurately each method estimates the unknown parameters as the level of state uncertainty increases, under idealized algorithmic conditions. To isolate algorithmic performance, the experiment deliberately abstracts away practical implementation considerations and focuses solely on relative estimation accuracy with respect to existing state-of-the-art methods.

Experimental Configuration

First, the configuration of the seabed is specified. As explained in Section 2-1, the seabed model is specified in terms of the number of basis functions N, and the $N-1$ different frequency vectors f_n. In this experiment, the number of unknown parameters is set to $N = 11$, which means ten different frequency vectors have to be specified. Specifically, $f_0 = [0, 0]^\mathsf{T}$, $f_n = [n\frac{2\pi}{10}, 0]^\mathsf{T}$ for $n \in \{1, 2, 3\}$, and $f_n = [(n-7)\frac{2\pi}{10}, \frac{2\pi}{6}]^\mathsf{T}$ for $n \in \{4, \dots, 10\}$.

As specified by Equation (2-6), the unknown parameters follow a uniform distribution. The lower bound is set to $a = 2$ and the upper bound is $b = 8$. Therefore, the distribution for the unknown parameters is $p(\theta) = \mathcal{U}([2, 8]^{11})$, which has mean $\mu_{\theta_n} = 5$ and covariance $\sigma^2_{\theta_n} = 3$.

The state-transition and input matrices are given by $\mathrm{A}_t = \mathbb{I}$ and $\mathrm{B}_t = \Delta t, \mathbb{I}$, with $\Delta t = 0.1$. As defined in Section 2-1, the process and measurement noise are Gaussian. Two different variances for the process noise are tested, the values are $\Sigma_\mathrm{w} = \sigma^2_\mathrm{w}\mathbb{I}$ where, $\sigma^2_\mathrm{w} = \{0.001,\, 0.01\}$. The measurement noise variance was kept consistent at $\sigma^2_\mathrm{v} = 0.01$ across all experiments. Finally, the initial position of the AUV is specified using a Gaussian distribution with mean $\mu_{\mathrm{x}_0} = [3.2, 2.8]^\mathsf{T}$ and covariance Σ_w.

A sequence of inputs $\bar{\mathrm{u}}^*$ of length $T = 100$ was taken from [52, Section 6], where each individual input is constrained within $[-200, 200]$. To generate 100 different state trajectories, 100 different realizations of process noise $\{\bar{\mathrm{w}}^i\}_{i=1}^{100}$ were sampled for both values of σ^2_w. The state trajectories were created according to Equation (3-3), i.e.

$$\bar{\mathrm{x}}^i = \bar{\mathrm{A}}(\bar{\mathrm{B}}\bar{\mathrm{u}}^* + \bar{\mathrm{w}}^i)$$

To generate different sequences of observations for the same state trajectory, 100 different realizations of theta were sampled from the parameter distribution, i.e. $\{\theta^i\}_{i=1}^{100}$, where $\theta^i \sim p(\theta)$. This construction allows the effect of parameter and measurement uncertainty to be evaluated independently of the state dynamics.

Finally, 100 different realizations of measurement noise were generated, i.e $\{\bar{\mathrm{v}}^i\}_{i=1}^{100}$. These were combined into 10,000 different observation sequences according to

$$\bar{\mathrm{y}}^{i,j} = \Phi(\bar{\mathrm{x}}^i)^\top \theta^j + \bar{\mathrm{v}}^i, \quad i, j = 1, \dots, 100,$$

That is, for each state trajectory i, a single measurement noise realization $\bar{\mathrm{v}}^i$ was reused across all parameter realizations j. As shown in Figure 5-1, varying the state trajectory, or parameter realization leads to significantly different observation sequences. Having specified how the synthetic dataset was created, the configuration of the algorithms is now provided.

Benchmark Algorithm Configuration

- **dual state-parameter:** As discussed in Section 3-5, Algorithm 1 requires one tuning parameter, the value of ϵ which determines convergence. This was set to $\epsilon = 10^{-4}$.
- **PGAS:** As discussed in Section 4-2, Algorithm 3 requires three tuning parameters. These are, the number of particles N_f, the number of iterations K, and the burn-in period L. To ensure particle degeneracy was not a fundamental issue a large number of particles was used namely $N_f = 7000$. The algorithm was run for a total of $K = 1000$ iterations. Only the second of samples were used, meaning $L = 500$.

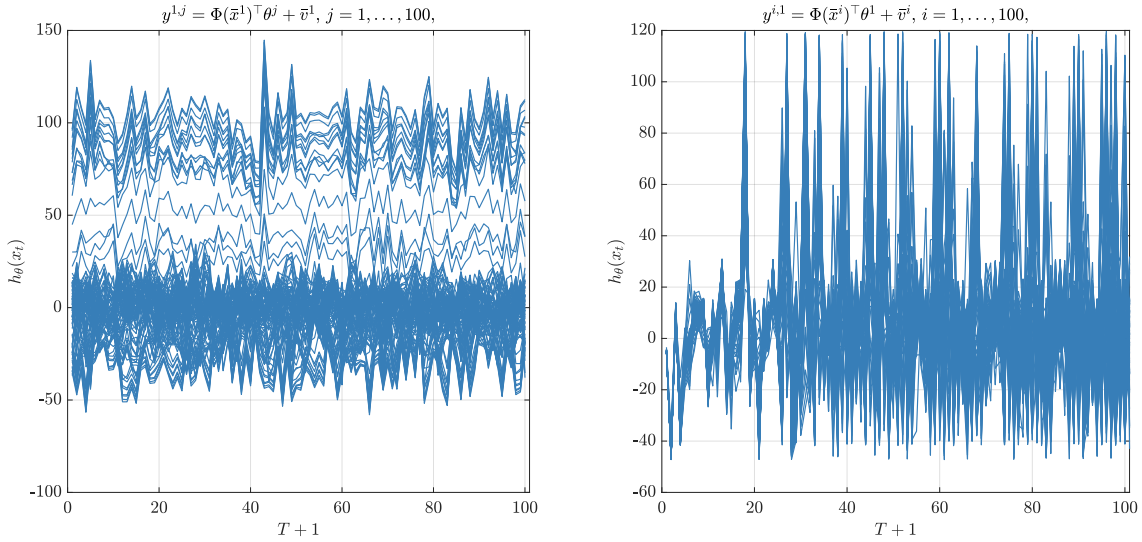

Figure 5-1: Observation sequences: left, varying parameters; right, varying noise.

- **PSEM:** As discussed in Section 4-3, Algorithm 5 requires 3 tuning parameters. These are, the number of particles forward particles N_f, the number of smoothing particles N_s and the number of iterations, K. To ensure particle degeneracy was not a limiting issue of the underlying Sequential Monte Carlo (SMC) method, a large number of particles was used namely $N_f = 7000$ and $N_s = 750$. To give the algorithm enough iterations, the algorithm was stopped after $K = 1000$.

Results

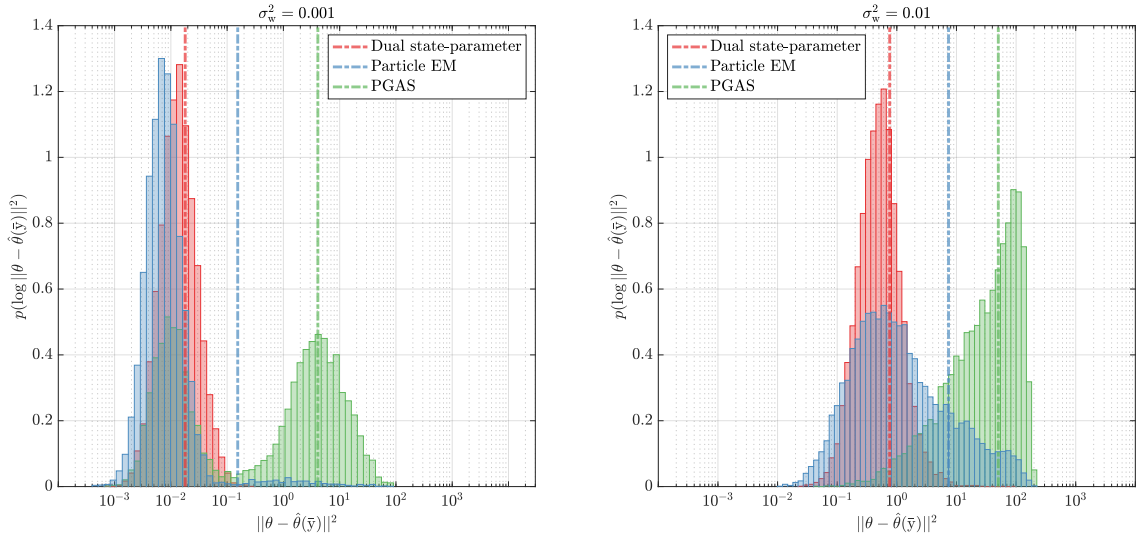

Figure 5-2: Parameter estimation error distributions of SMC-based methods and proposed method

Figure 5-2 compares the distributions of $\log ||\theta - \hat{\theta}(\bar{y})||^2$ across all 10,000 simulations for the

dual state-parameter estimator (red), PSEM (blue), and PGAS (green). Each histogram is normalized to integrate to one, representing the probability density of the squared error on a logarithmic scale. The dashed vertical lines indicate the empirical Mean Square Error (MSE) of each method. The dual state-parameter estimator exhibits a concentrated error distribution for both noises conditions, while PSEM and PGAS show increasingly heavy-tailed distributions as process noise increases.

In the low process-noise case $\sigma_{\mathrm{w}}^2 = 0.001$, PSEM demonstrates a lower MSE in a majority of runs, achieving a median error of 0.0076 compared to 0.0133 for the dual state-parameter estimator, and a first quartile of 0.0047 versus 0.0081 (Table 5-1). As discussed in Section 4-3, a uniform prior renders the MAP estimation equivalent to constrained maximum likelihood estimation. By contrast, the dual state–parameter estimator relies on a Gaussian approximation of the parameter posterior, introducing a distributional mismatch that degrades performance when state uncertainty is small.

However, PSEM's error distribution exhibits a heavy right tail, as evidenced by a mean and standard deviation (0.1543 and 1.9447) that are orders of magnitude larger than the median and interquartile range, indicating that a subset of runs converge to poor local optima. This variability results in a substantially higher overall MSE compared to the dual state-parameter estimator, which maintains consistent performance across realizations with a mean of 0.0180 and standard deviation of 0.0162. When process noise increases to $\sigma_{\mathrm{w}}^2 = 0.01$, the advantage of PSEM's diminishes. This is most likely due to the fact EM gets stuck in local minima more quickly compared to the dual state-parameter estimator when the process noise is increased.

PGAS exhibits substantially worse performance than both the dual state-parameter estimator and PSEM across both noise levels. In the low process-noise case $\sigma_{\mathrm{w}}^2 = 0.001$, the error distribution is bimodal (Figure 5-2, left panel), with one mode concentrated near 10^{-2} where PGAS achieves errors comparable to the dual estimator (Q1 = 0.0117), and a second mode centered around 10^{0} representing failed convergence cases. This bimodality arises from the Gibbs sampling structure: when initialized with parameters far from the true values, the conditional state distribution $p(\bar{\mathrm{x}} \mid \bar{\mathrm{y}}, \theta)$ may place negligible probability mass near the true state trajectory, causing the particle filter to lose track of the system. Once the state estimate diverges, the subsequent parameter update step $p(\theta \mid \bar{\mathrm{x}}, \bar{\mathrm{y}})$ operates on an incorrect state trajectory, preventing recovery and trapping the algorithm in a poor region of the parameter space.

In contrast, when process noise increases to $\sigma_{\mathrm{w}}^2 = 0.01$, the bimodal structure largely disappears and the distribution shifts toward uniformly poor performance (median 35.8833, IQR 71.2014). The increased state uncertainty renders particle degeneracy systematic rather than episodic, persisting even with a large particle budget ($N_f = 7000$). This reveals a fundamental limitation of PGAS, its inability to adequately explore the parameter space under high process noise. Such limitations could likely be alleviated by smoothing methods that incorporate an explicit backward pass.

Overall, the Monte Carlo study demonstrates that the dual state-parameter estimator achieves superior performance, while offering significant computational advantages. Across all 10,000 simulations, the dual estimator consistently achieves the lowest mean squared error in both noise regimes, and its performance degrades less as process noise increases, compared to the benchmark methods. This indicates better scalability to higher state uncertainty levels.

Beyond accuracy, the dual estimator demonstrates substantially faster convergence: across all experiments, the algorithm converged within a maximum of 412 iterations (with $\epsilon = 10^{-4}$), and never required the full budget of 1000 iterations. In contrast, PSEM exhausted all 1000 allocated iterations in every experiment without satisfying standard convergence criteria, despite using 7000 forward particles and 750 smoothing particles per iteration. The computational efficiency advantage, combined with superior accuracy and robustness to varying noise levels, establishes the dual state-parameter estimator as the most practical choice among the evaluated algorithms.

	Method	Mean	Std	Median	Q1	Q3	IQR
$\sigma_{\mathrm{w}}^2 = 0.001$	dual state-parameter	0.0180	0.0162	0.0133	0.0081	0.0220	0.0139
	PSEM	0.1543	1.9447	0.0076	0.0047	0.0122	0.0075
	PGAS	4.0977	7.7215	1.0107	0.0117	4.7839	4.7721
$\sigma_{\mathrm{w}}^2 = 0.01$	dual state-parameter	0.7596	1.478	0.5065	0.2959	0.8446	0.5487
	PSEM	7.3624	19.5159	0.8700	0.2911	3.7117	3.4205
	PGAS	50.3419	45.6790	35.8833	10.9081	82.1095	71.2014

Table 5-1: Summary statistics of estimation error for dual state–parameter, PSEM, and PGAS.

5-2 High-Fidelity Simulator

To assess the performance of the proposed dual state-parameter estimator under more realistic operating conditions. Specifically, the SMARCSIM simulator is extended [29]. SMARCSIM provides a detailed and physically accurate simulation environment for an existing AUV, called LoLo [19], including its nonlinear dynamics and onboard sensor. The simulator generates realistic inertial measurements, as well as depth measurements obtained from a simulated sonar sensor.

The simulated inertial and depth measurements are used as inputs to the estimator. When restricted to these measurements, the resulting estimation model corresponds to the same double-integrator dynamics used in the Monte Carlo experiment, allowing the estimator to be applied without modification. In this sense, the estimator operates under the same modelling assumptions as before, while the data are generated by a significantly more realistic system.

The seabed model employed in this experiment uses the same parameterization as in the Monte Carlo study, and the seabed parameters are treated as unknown and estimated jointly with the AUV position. The process and measurement noise statistics assumed by the estimator are also identical to those used previously, and the high process-noise scenario is considered. By keeping the experimental conditions consistent, this study enables a meaningful assessment of the performance degradation incurred when transitioning from a simplified simulation model to a high-fidelity, physics-based simulator.

The following section describes the structure of the SMARCSIM simulator and its role in generating the data used in this experiment.

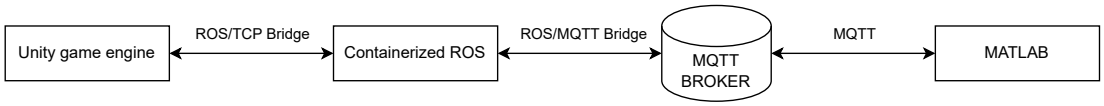

Figure 5-3: Structure of SMARCSIM simulator

Simulator Structure

The simulator architecture integrates three environments: Unity game engine, Robot Operating System (ROS), and MATLAB. These components communicate through a publish-subscribe messaging paradigm facilitated by a Message Queuing Telemetry Transport (MQTT) broker. The structure and communication pathways are shown in Figure 5-3.

- **Unity Game Engine:** Unity serves as the core physics simulation and rendering environment. It numerically integrates the equations of motion for the simulated vehicle, computing the dynamic state evolution including position, velocity, orientation, and angular rates based on applied control inputs. Unity also generates simulated sensor measurements for the IMU data and Sonar sensors with specified noise characteristics and sampling rates to ensure simulation fidelity. Unity communicates with the ROS network through a bidirectional ROS/TCP bridge, receiving control commands and transmitting sensor data and vehicle state information.

- **Containerized ROS Network:** The ROS environment operates within a containerized deployment, because the author's workstation cannot host ROS networks natively. It serves as the middleware layer that interfaces between the Unity simulator and the higher-level MATLAB node. ROS receives sensor data and state information from Unity while transmitting control commands back to the simulation. It implements the low-level control architecture that processes high-level commands from MATLAB and generates specific orientation and velocity control signals for the vehicle, most importantly attitude and position controllers. ROS connects to Unity via a TCP bridge and to the MQTT broker through a ROS/MQTT bridge, enabling communication with MATLAB.

- **MQTT Broker:** The MQTT broker functions as the central message routing infrastructure, implementing a lightweight publish-subscribe protocol. It decouples the ROS and MATLAB components without direct point-to-point connections. An MQTT broker is required because this follows the architectural design decisions established by the SMARC project. High-level commands are sent to the ROS network over MQTT, which are then used for implementing complex mission behaviors and coordination between system components.

- **MATLAB Node:** MATLAB functions as the host environment for the dual estimation algorithm. MATLAB was selected because an existing implementation of the estimation algorithm was available in this language, enabling code reuse and reducing development time. MATLAB generates high-level mission commands including waypoint targets and desired trajectories, which are transmitted to ROS via the MQTT broker.

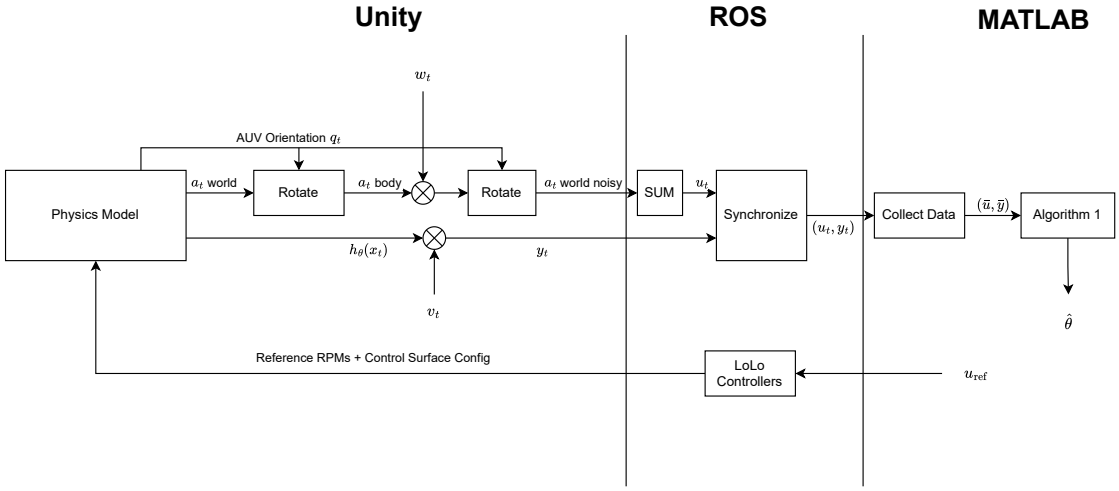

Figure 5-4: Implemented Simulator Data Logic

Flow of Data

Having defined the global structure of the simulator, this section describes the flow of data through the system during operation, as illustrated in Figure 5-4.

Before the simulation loop begins, the seabed is set according to the model specified by Equation (2-1), so that the seabed model matches the output map exactly. For testing purposes, we generate another seabed by sampling a $\theta_{\text{sim}} \sim p(\theta)$. The simulation loop then proceeds within the Unity physics model, which computes the true vehicle dynamics. At each time step, the physics model generates the true acceleration a_t in the body frame and the true depth measurement $h_\theta(\mathrm{x}_t)$ based on the current vehicle position and the seabed model. These measurements are corrupted by additive noise: the acceleration is affected by zero-mean process noise w_t, and the depth measurement is affected by zero-mean measurement noise v_t.

The Unity simulator computes the vehicle's motion in the world frame, while the IMU measures accelerations in the body frame. To simulate the noisy IMU measurements, the true acceleration in the world frame is first rotated into the body frame using the current vehicle orientation quaternion q_t. Additive noise is then applied in the body frame to model sensor imperfections. The resulting noisy acceleration is rotated back into the world frame and integrated to obtain u_t, the planar velocity of the vehicle. The depth measurement y_t is used directly, with only additive measurement noise applied.

Both sensor measurements, u_t and y_t, are transmitted from Unity to the ROS network via the ROS/TCP bridge. Within ROS, these asynchronous measurements are synchronized to ensure they are aligned in time, producing the paired measurements $(\mathrm{u}_t, \mathrm{y}_t)$ that are forwarded to MATLAB.

In MATLAB, the "Collect Data" block receives the synchronized measurements and concatenates them as inputs $(\bar{\mathrm{u}}, \bar{\mathrm{y}})$ which is used as input for the dual estimation algorithm. The dual estimation algorithm processes these measurements to produce state estimates $\hat{\theta}$, which

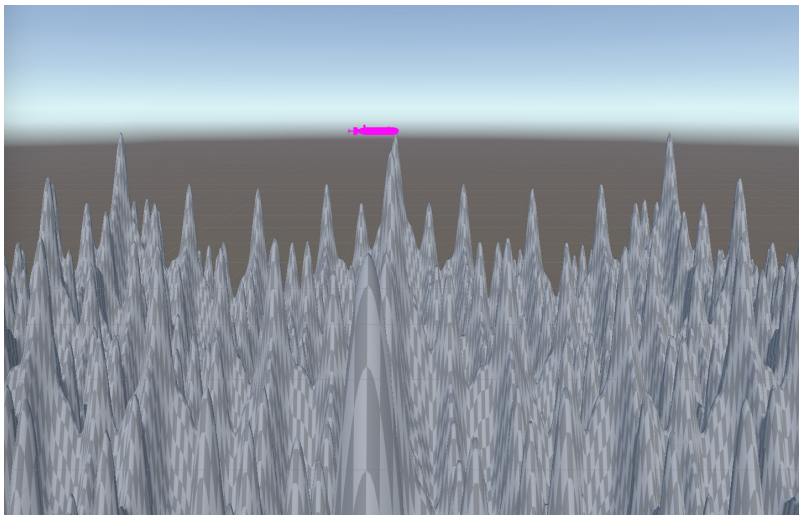

Figure 5-5: Example Simulator View from Unity

include both the vehicle position and the unknown seabed parameters.

Simultaneously, MATLAB generates high-level velocity commands u_t based on the desired mission trajectory. These commands are transmitted back to ROS via the MQTT broker, where the LoLo controllers convert the velocity setpoints into low-level control signals. Specifically, the controllers determine the required thruster RPMs and control surface configurations to achieve the commanded velocity. These control commands are sent back to the Unity physics model, closing the control loop and driving the vehicle dynamics for the next simulation time step.

Additionally, SMARCSIM allows manual control of the AUV via a keyboard interface, enabling the operator to adjust propeller RPMs and control surfaces to modify the vehicle's orientation. To generate a test input for the estimator, an example trajectory was followed for 30 seconds, producing a sequence of 300 measurements sampled at 0.1 seconds. This process yields a measured sequence of vehicle velocities and depth readings, which serves as the input for Algorithm 1.

Experimental Configuration

The high-fidelity SMARCSIM experiment was designed to evaluate the proposed dual estimation algorithm under realistic operating conditions. To maintain focus, the scope of this study is deliberately limited: only a single seabed realization a single vehicle trajectory is considered. A view of the simulator can be seen in Figure 5-5.

The seabed surface was generated according to the ground-truth model described in Section 5-1, with $N = 11$ unknown parameters. The frequency vectors were specified as $f_0 = [0, 0]^\top$, $f_n = [n\frac{2\pi}{10}, 0]^\top$ for $n \in \{1, 2, 3\}$, and $f_n = [(n-7)\frac{2\pi}{10}, \frac{2\pi}{6}]^\top$ for $n \in \{4, \ldots, 10\}$. The parameter vector θ was set to the first realization from the Monte Carlo study θ^1, providing a fixed reference for evaluation.

The process noise was set to $\sigma_w^2 = 0.01$, and the measurement noise variance was $\sigma_v^2 = 0.01$. The AUV was initialized at the origin $(0, 0)^\top$ in the Unity environment. A single different trajectories were generated manually using keyboard input, creating realistic, non-repetitive motion sequences. The IMU measurements were sampled at 0.1s intervals, while sonar depth measurements were sampled every 0.5s, consistent with the SMARCSIM sensor models. These measurements were then fed into MATLAB and processed using Algorithm 1, as illustrated by Figure 5-4.

Results

The right plot of Figure 5-6 shows the MSE as a function of the number of measurements for the two manually trajectory. The trajectory is itself shown in the left plot in Figure 5-6. The MSE decreases monotonically as more measurements are incorporated, demonstrating consistent convergence of the estimator. After 300 measurements, the MSE reaches 0.03.

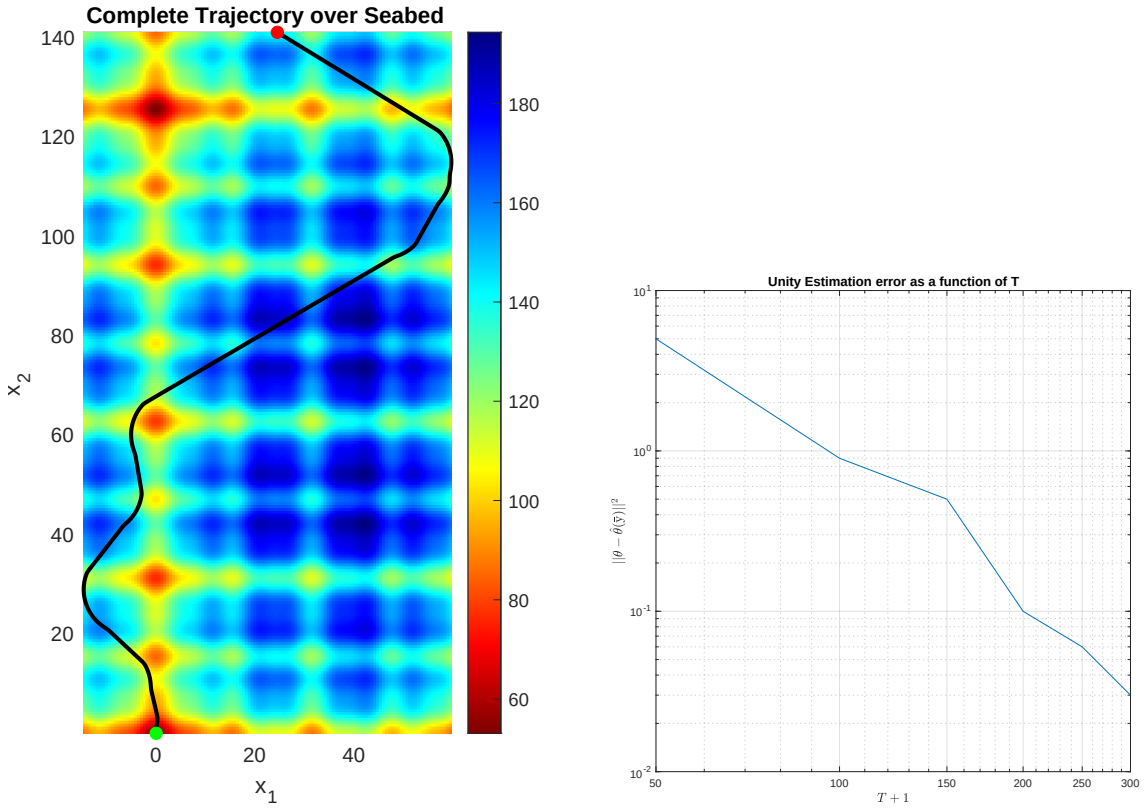

Figure 5-6: Results of Unity experiment. Left: input trajectory, right: estimation error.

5-3 Limitations

The proposed dual state–parameter estimator was evaluated through two complementary experiments. A large-scale Monte Carlo study with 10,000 simulations showed that the method consistently outperforms two state-of-the-art data augmentation approaches in the high-noise

experiment. While PSEM can achieve lower errors in some cases, its heavy-tailed error distribution indicates lower robustness and a higher overall expected mean squared error, and PGAS exhibits even more severe failure modes. In contrast, the proposed method maintains a tightly concentrated error distribution, better overall accuracy, and significantly faster convergence compared to PSEM.

The second experiment extended a high-fidelity simulator to generate realistic bathymetric data, capturing practical imperfections that arise in real experimental settings. The dual state–parameter approach demonstrated robustness to imperfections in the data collection process and continued to perform reliably, confirming its practical applicability beyond idealized simulation settings. Although the second experiment serves more as anecdotal evidence since only a single trajectory was considered.

Both experimental studies do not fully capture the scale of a realistic bathymetric survey. In practical settings, the recorded trajectory would be much longer, i.e., $T \gg 300$. Even with a conservative estimate of one depth measurement per second, the AUV can cover only a very limited area in 300 seconds. A more realistic simulation would consider several hours of AUV data, possibly with faster sampling rates. Consequently, the proposed method should be able to handle much longer trajectories, on the order of $T \approx 10^5$ measurements. Trajectories of this length are required to acquire sufficiently accurate bathymetric data for a meaningful survey area.

Both experiments employ a highly simplified seabed representation with $N = 11$ components. In practical scenarios, accurately modeling the seabed would require $N \gg 11$, which also increases computational demands. Strategies to address these issues, including scalable algorithmic approaches and systematic frequency selection, are discussed in Section 7-2.

Additionally, Algorithm 1 is not capable of handling data records of this length. Assuming $\mathrm{N} \ll T$, the dual state-parameter estimator has a per-iteration computational complexity of $\mathcal{O}(T^3)$, due to the computation of the Dynamic Basis Statistics (DBS) collection at each iteration. Consequently, computing the dual state-parameter estimator in Section 5-2 took over 5 minutes on a MacBook Pro with an M1 Pro processor. Given this, processing data records of $T \approx 10^5$ would be infeasible even when using improved computational resources.

The next chapter addresses this scalability limitation. Specifically, it introduces a recursive computation that reduces the time complexity of the dual state–parameter estimator.

Chapter 6

Filtering Dual Affine MMSE

As described in Chapter 1, the primary objective of this thesis is to assess the feasibility of model-driven estimation methods for bathymetric mapping problems. To this end, Chapter 3 introduced a dual state–parameter estimation framework, which was shown in Chapter 5 to outperform state-of-the-art approaches. The focus of the present chapter is the scalability of the dual state-parameter estimator. Specifically, this chapter derives a recursive dual state–parameter estimator with linear computational complexity, enabling online operation on long data records.

In batch form, the proposed method (Algorithm 1) is ill-suited for large-scale problems, as its computational complexity scales as $\mathcal{O}(T^3)$. This scaling is dominated by the repeated computation of the Dynamic Basis Statistics (DBS), which requires processing the entire data record. To overcome this limitation, this chapter introduces a recursive implementation of the dual state–parameter estimator, referred to as the parallel point filter, reflecting its pointwise update structure. The resulting algorithm achieves a computational complexity of $\mathcal{O}(T)$. To quantify the performance degradation introduced by the recursive formulation, a Monte Carlo study is conducted in which the parallel point filter is applied to a long data record and compared against a particle filter–based extension of the dual Kalman framework [53].

The remainder of this chapter is organized as follows. Section 6-1 provides a brief overview of recursive estimation theory and discusses recursive data augmentation strategies. Section 6-2 discusses an existing approach to the dual filtering problem. Section 6-3 presents the parallel point filter, which constitutes a recursive adaptation of Algorithm 1. Finally, Section 6-5 evaluates the proposed method through a Monte Carlo study, comparing its performance on very long data records against a state-of-the-art filtering approach.

6-1 Recursive Estimation

Recall that Section 2-3 defined the estimator $\hat{\theta}(\bar{\mathrm{y}})$ as a mapping from the full observation vector $\bar{\mathrm{y}}$ to an estimate $\hat{\theta}$, by minimizing Equation (2-9) for a particular choice of loss function. A key feature of this formulation is that all measurements are processed jointly, requiring

access to the complete dataset. In contrast, a recursive estimation algorithm processes the data sequentially, processing one measurement at a time. Each new observation y_t is used to update the current estimate $\hat{\theta}(t)$, after which the measurement itself is discarded. Information from past data is retained only through a statistic $S(t)$ acting as a memory buffer. Formally, a recursive estimation algorithm can be represented as:

$$\begin{cases} \hat{\theta}(t) = F(\hat{\theta}(t-1),\, S(t),\, \mathrm{y}_t), \\ S(t) = H(S(t-1),\, \hat{\theta}(t-1),\, \mathrm{y}_t). \end{cases} \tag{6-1}$$

The functions $F(\cdot)$ and $H(\cdot)$ define a fixed update step for $\hat{\theta}(t)$ and $S(t)$, for which the computational complexity does not depend on t. The information contained in $\mathrm{y}_{0:t}$ is summarized in the pair $(\hat{\theta}(t), S(t))$. So, instead of computing a single estimate $\hat{\theta}(\bar{\mathrm{y}})$, recursive alorithms produce a sequence of T estimates $\{(\hat{\theta}(t), S(t))\}_{t=0}^{T}$. By leveraging this structure, recursive algorithms are able to efficiently handle large amounts of data.

While there are many approaches to defining $F(\cdot)$, $H(\cdot)$ and $S(t)$ this thesis adopts the approach from [36, Chapter 2.3], which postulates recursive estimation as a non-linear Bayesian filtering problem. This perspective is adopted as it fits closely to the original problem definition presented in Chapter 2. The Bayesian filtering perspective is also widely adopted in the literature, as seen in, e.g., [7, 37, 53, 13].

In Bayesian filtering, the objective at time t is to characterize the posterior distribution $p(\xi \mid \mathrm{y}_{0:t})$, where ξ is a random variable of interest, commonly x_t or θ. By choosing $\hat{\xi}(t) = \mathbb{E}^{p(\xi|\mathrm{y}_{0:t})}[\xi]$ or $\hat{\xi}(t) = \arg\max_{\xi} p(\xi \mid \mathrm{y}_{0:t})$, the Minimum Mean Square Error (MMSE) and Maximum a Posteriori (MAP) estimators introduced in Section 2-3 are recovered in a recursive setting. At the final time T, the recursive formulation targets the same posterior distribution as the batch formulation, although this distribution may be represented or approximated differently depending on the filtering algorithm. In any case, the recursive estimator is obtained through efficient sequential updates as defined by Equation (6-1).

While the recursive formulation enables efficient, online computation of parameter estimates, it fundamentally restricts access to past measurements, which cannot be revisited once incorporated. Consequently, the performance of a recursive estimator depends critically on how effectively the statistic $S(t)$ summarizes the information contained in the measurement history. In nonlinear settings, constructing such summaries is challenging, and the resulting filtering problem admits no closed-form solution. As in the original batch formulation, implementations therefore rely on approximation, which in turn gives rise to recursive implementations of data augmentation algorithms.

For the data augmentation methods discussed in Chapter 3, only the dual smoothing approach [26, 53] is well suited for a recursive formulation. In contrast, Gibbs sampling does not naturally admit a recursive implementation as it would require restarting the chain whenever a new measurement arrives. While there have been attempts proposed to overcome this limitation [18], they have seen limited practical adoption [28]. Similarly, while the Expectation Maximization (EM) algorithm can be formulated recursively [10], doing so requires solving an online smoothing problem. Methods exist for this [44], but they are known to be brittle for systems with multiple unknown parameters [9]. Given these considerations, the dual Kalman framework provides a natural basis for a benchmark recursive method.

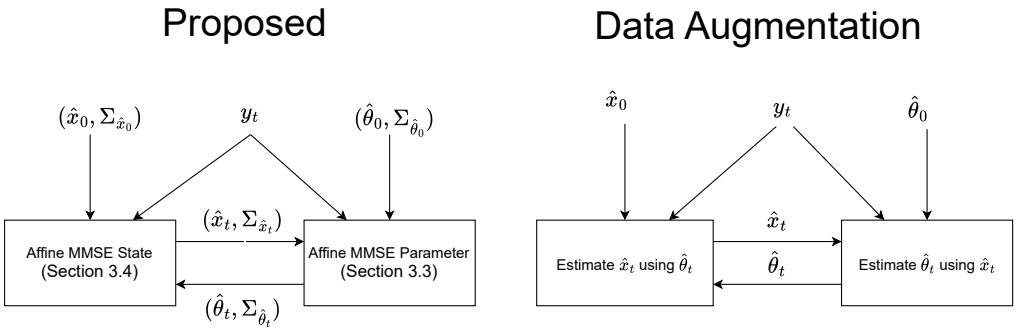

Figure 6-1: Structure of Recursive Augmentation Approaches

6-2 Online Joint State-Parameter Estimation

The previous section introduced the concept of recursive estimation and discussed how the data augmentation strategies from Chapter 3 can be adapted to a recursive setting. It was also argued that the dual filtering approach provides a natural benchmark for online joint state-parameter estimation. In this section, the technical details of the dual filtering approach from [26, Chapter 7] are presented .

As shown in Figure 6-1, the overall structure of the data augmentation approach decomposes the augmented estimation problem into two distinct filtering tasks: state estimation and parameter estimation. The dual filtering framework is modular, it does not prescribe a single specific algorithm. Rather, the dual estimation framework provides a general recipe for combining existing state and parameter filters to address the augmented estimation problem.

Due to its modular structure, the dual filtering framework admits multiple implementations. In the special case where both the system dynamics and observation model are linear, the framework can be realized using two Kalman filters [42]. For nonlinear systems, early approaches relied on the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) [26, Chapters 5, 7.]. More recently, the particle filter has become the preferred choice for the state estimation component due to its improved accuracy [48]. While particle methods perform well for state filtering, extending them to parameter estimation remains challenging, which is why the EKF and UKF are still commonly used for the parameter estimation component.

In the configuration used for benchmarking, a Bootstrap Particle Filter (BPF) (see Appendix A-1) is used for state estimation, while a standard Kalman filter is employed for parameter estimation. This choice is motivated by the problem structure: the measurement model is linear in θ, if a state estimate is provided. The BPF is preferred for state estimation due to its improved accuracy, compared to the EKF or UKF. The overall structure of the dual filtering setup is illustrated in Figure 6-2.

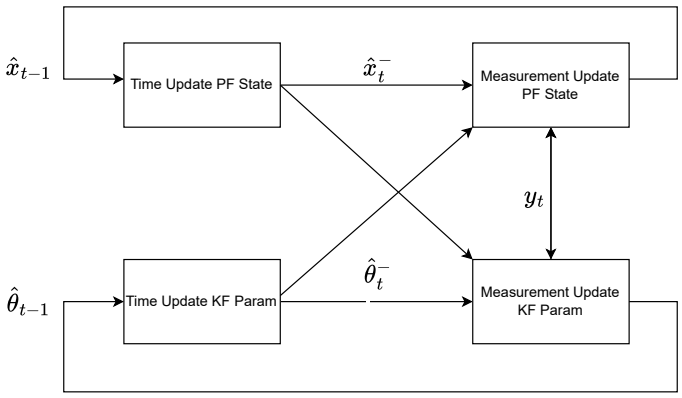

Figure 6-2: Schematic of BPF-KF Algorithm

6-3 Proposed Method

This section develops a recursive counterpart of the dual state–parameter estimator. As introduced in Chapter 3, the estimator combines two affine MMSE estimators: one targeting the unknown parameters, and one targeting the unknown states. Because both estimators are Bayesian, they naturally admit a recursive implementation. By combining the individual recursive estimators in the same manner as in Section 3-5, the dual state-parameter estimator can be computed recursively. The construction proceeds in three steps: (1) define the recursive affine parameter MMSE update, (2) define the recursive affine state MMSE update, and (3) combine them to form the recursive dual state-parameter estimator.

Recall that recursive algorithms produce an updated estimate at each time step t. For the dual state-parameter estimator, both affine estimators are Bayesian and yield Gaussian approximations of the posterior distributions $p(\theta \mid \mathrm{y}_{0:t}, \hat{\mathrm{x}}_{0:t})$ and $p(\mathrm{x}_{0:t} \mid \mathrm{y}_{0:t}, \hat{\theta})$, each fully characterized by their mean and covariance. This property enables the construction of sequential Bayesian updates, in which the estimators (and their covariances) obtained at time $t-1$ are used as priors at time t. As a result, equations derived in Chapter 3 can be reformulated as efficient rank-1 recursive updates to obtain a recursive implementation of the dual state-parameter estimator.

For the affine MMSE parameter estimator introduced in Section 3-3, a recursive update can be derived directly from Equation (6-2) by setting the full observation vector $\bar{\mathrm{y}}$ to the current measurement y_t. This yields the following sequential update for the parameter estimate and its covariance at time t:

$$\begin{cases} \hat{\theta}_t(\mathrm{y}_t \,|\, \hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \mu_{\phi_t}, \Sigma_{\phi_t}) = \Psi^\star_{\hat{\theta}_t}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \mu_{\phi_t}, \Sigma_{\phi_t})\mathrm{y}_t + \psi^\star_{\hat{\theta}_t}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \mu_{\phi_t}, \Sigma_{\phi_t}), \\ \Sigma_{\hat{\theta}_t}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \mu_{\phi_t}, \Sigma_{\phi_t}) = \Sigma_{\hat{\theta}_{t-1}} - \Sigma_{\hat{\theta}_{t-1}}\mu_{\phi_t}(\mu_{\phi_t}^{\mathsf{T}}\Sigma_\theta\mu_{\phi_t} + \mathcal{M}(\Sigma_{\phi_t}) + \sigma_{\mathrm{v}}^2)^{-1}\mu_{\phi_t}^{\mathsf{T}}\Sigma_{\hat{\theta}_{t-1}}. \end{cases} \tag{6-2}$$

The resulting pair $(\hat{\theta}_t, \Sigma_{\hat{\theta}_t})$ serves as the prior for the update corresponding to the next measurement y_{t+1}. The notation in Equation (6-2) emphasizes that the estimates at time t are conditioned on the estimates from time $t-1$. Accordingly, the solution of the underlying optimization problem (Equation (3-7)) evolves over time, reflecting the sequential incorporation

of new observations. At each time step t, the optimal matrices $\Psi^\star_{\theta_t}$ and $\psi^\star_{\theta_t}$ are computed based on the current prior and the new measurement.

$$\begin{aligned}\Psi^\star_{\theta_t}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \mu_{\phi_t}, \Sigma_{\phi_t}) &= \Sigma_{\hat{\theta}_{t-1}}\mu_{\phi_t}(\mu_{\phi_t}^\mathsf{T}\Sigma_{\hat{\theta}_{t-1}}\mu_{\phi_t} + \mathcal{M}(\Sigma_{\phi_t}) + \sigma_\mathrm{v}^2)^{-1},\\ \psi^\star_{\theta_t}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \mu_{\phi_t}, \Sigma_{\phi_t}) &= \hat{\theta}_{t-1} - \Psi^\star_{\theta_t}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \mu_{\phi_t}, \Sigma_{\phi_t})\mu_{\phi_t}^\mathsf{T}\hat{\theta}_{t-1}.\end{aligned} \tag{6-3}$$

Equations (6-2) and (6-3) provide the recursive counterpart to Equation (3-11), enabling the sequential update of the parameter estimates as new measurements become available.

Similarly, for the affine MMSE state estimator introduced in Section 3-4, the recursive update combines the measurement y_t with the state estimate computed at $t-1$, namely $(\hat{\mathrm{x}}_{t-1}, \Sigma_{\hat{\mathrm{x}}_{t-1}})$. To account for dynamic of the underlying process model, the estimates from time $t-1$ are pushed through the dynamic to obtain the prior variable at time t, often called the time update. The time update for state and DBS at time $t-1$ is:

$$\begin{cases}\hat{\mathrm{x}}_t^- = \mathrm{A}_t\hat{\mathrm{x}}_{t-1} + \mathrm{B}_t\mathrm{u}_{t-1}\\ \Sigma_{\hat{\mathrm{x}}_t^-} = \mathrm{A}_t\Sigma_{\hat{\mathrm{x}}_t}\mathrm{A}_t^\top + \Sigma_\mathrm{w}\\ (\hat{\phi}_t^-, \Sigma_{\hat{\phi}_t^-}) = \mathrm{DBS}(\hat{\mathrm{x}}_t^-, \Sigma_{\hat{\mathrm{x}}_t^-})\end{cases} \tag{6-4}$$

Using the time update formulas and Equation (3-18), the recursive update for the state estimator at time t is:

$$\begin{cases}\hat{\mathrm{x}}_t(\mathrm{y}_t \,|\, \mu_\theta, \Sigma_\theta, \hat{\mathrm{x}}_t^-, \Sigma_{\hat{\mathrm{x}}_t^-}) = \hat{\mathrm{x}}_t^- + \Sigma_{\hat{\mathrm{x}}_t^-}C_t^\mathsf{T}\mu_\theta(\mu_\theta^\mathsf{T}\hat{\phi}_t^-\mu_\theta + \mathcal{S}(\Sigma_\theta) + \sigma_\mathrm{v}^2)^{-1}(\mathrm{y}_t - \hat{\phi}_t^{-\mathsf{T}}\mu_\theta),\\ \Sigma_{\hat{\mathrm{x}}_t}(\mu_\theta, \Sigma_\theta, \hat{\mathrm{x}}_t^-, \Sigma_{\hat{\mathrm{x}}_t^-}) = \Sigma_{\hat{\mathrm{x}}_t^-} - \Sigma_{\hat{\mathrm{x}}_t^-}C_t^\mathsf{T}\mu_\theta(\mu_\theta^\mathsf{T}\hat{\phi}_t^-\mu_\theta + \mathcal{S}(\Sigma_\theta) + \sigma_\mathrm{v}^2)^{-1}\mu_\theta^\mathsf{T}C_t\Sigma_{\hat{\mathrm{x}}_t^-}^\mathsf{T},\end{cases} \tag{6-5}$$

Equations (6-4) and (6-5) provide the recursive counterpart to Equation (3-18), enabling the sequential update of the states estimates as new measurements become available.

Motivated by Lemma 3-5.1, the estimated DBS $(\hat{\phi}_t^-, \Sigma_{\hat{\phi}_t^-})$ is used to condition the recursive affine MMSE parameter update, resulting in a nonlinear parameter estimator with improved performance:

$$\begin{cases}\hat{\theta}_t^{\mathrm{nl}}(\mathrm{y}_t \,|\, \hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \hat{\phi}_t^-, \Sigma_{\hat{\phi}_t^-}) = \Psi^\star_{\theta_t}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \hat{\phi}_t^-, \Sigma_{\hat{\phi}_t^-})\mathrm{y}_t + \psi^\star_{\theta_t}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \hat{\phi}_t^-, \Sigma_{\hat{\phi}_t^-}),\\ \Sigma_{\hat{\theta}_t}^{\mathrm{nl}}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \hat{\phi}_t^-, \Sigma_{\hat{\phi}_t^-}) = \Sigma_{\hat{\theta}_{t-1}} - \Sigma_{\hat{\theta}_{t-1}}\hat{\phi}_t^-(\hat{\phi}_t^{-\mathsf{T}}\Sigma_\theta\mu_{\phi_t} + \mathcal{M}(\Sigma_{\hat{\phi}_t^-}) + \sigma_\mathrm{v}^2)^{-1}\hat{\phi}_t^{-\mathsf{T}}\Sigma_{\hat{\theta}_{t-1}}.\end{cases} \tag{6-6}$$

Likewise, to further improve state estimation, we can replace the prior statistics $(\mu_\theta, \Sigma_\theta)$ with parameter estimates $(\hat{\theta}_t, \Sigma_{\hat{\theta}_t})$ in (6-5) to obtain the following nonlinear state estimator:

$$\begin{cases}\hat{\mathrm{x}}_t^{\mathrm{nl}}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \hat{\mathrm{x}}_t^-, \Sigma_{\hat{\mathrm{x}}_t^-}) = \hat{\mathrm{x}}_t^- + \Sigma_{\hat{\mathrm{x}}_t^-}C_t^\mathsf{T}\hat{\theta}_{t-1}(\hat{\theta}_{t-1}^\mathsf{T}\hat{\phi}_t^-\hat{\theta}_{t-1} + \mathcal{S}(\Sigma_{\hat{\theta}_{t-1}}) + \sigma_\mathrm{v}^2)^{-1}(\mathrm{y}_t - \hat{\phi}_t^{-\mathsf{T}}\hat{\theta}_{t-1}),\\ \Sigma_{\hat{\mathrm{x}}_t}^{\mathrm{nl}}(\hat{\theta}_{t-1}, \Sigma_{\hat{\theta}_{t-1}}, \hat{\mathrm{x}}_t^-, \Sigma_{\hat{\mathrm{x}}_t^-}) = \Sigma_{\hat{\mathrm{x}}_t^-} - \Sigma_{\hat{\mathrm{x}}_t^-}C_t^\mathsf{T}\hat{\theta}_{t-1}(\hat{\theta}_{t-1}^\mathsf{T}\hat{\phi}_t^-\hat{\theta}_{t-1} + \mathcal{S}(\Sigma_{\hat{\theta}_{t-1}}) + \sigma_\mathrm{v}^2)^{-1}\hat{\theta}_{t-1}^\mathsf{T}C_t\Sigma_{\hat{\mathrm{x}}_t^-}^\mathsf{T},\end{cases} \tag{6-7}$$

Algorithm 6 summarizes the recursive dual state-parameter estimation procedure.

Algorithm 6 Parallel Point Filter Dual State-Parameter Estimator

1: $(\hat{\theta}_0, \Sigma_{\hat{\theta}_0}) \leftarrow (\mu_\theta, \Sigma_\theta)$
2: $(\hat{\mathrm{x}}_0^-, \Sigma_{\hat{\mathrm{x}}_0^-}) \leftarrow (\mu_{\mathrm{x}_0}, \Sigma_{\mathrm{x}_0})$
3: **for** $t = 0, 1, \ldots, T$ **do**
4: $\quad (\hat{\phi}_t^-, \Sigma_{\hat{\phi}_t^-}) \leftarrow \mathrm{DBS}(\hat{\mathrm{x}}_t^-, \Sigma_{\hat{\mathrm{x}}_t^-})$
5: $\quad r_t \leftarrow \mathrm{Tr}(\Sigma_{\hat{\phi}_t^-}(\Sigma_{\hat{\theta}_t} + \hat{\theta}_t\hat{\theta}_t^\top)) + \sigma_\mathrm{v}$
6: $\quad \hat{\theta}_{t+1} \leftarrow \hat{\theta}_t + \Sigma_{\hat{\theta}_t}\hat{\phi}_t^- \frac{\mathrm{y}_t - \hat{\phi}_t^{-\top}\hat{\theta}_t}{\hat{\phi}_t^{-\top}\Sigma_{\hat{\theta}_t}\hat{\phi}_t^- + r_t}$
7: $\quad \Sigma_{\hat{\theta}_{t+1}} \leftarrow \Sigma_{\hat{\theta}_t} - \Sigma_{\hat{\theta}_t}\frac{\hat{\phi}_t^-\hat{\phi}_t^{-\top}}{\hat{\phi}_t^{-\top}\Sigma_{\hat{\theta}_t}\hat{\phi}_t^- + r_t}\Sigma_{\hat{\theta}_t}$
8: $\quad c_t \leftarrow C(\hat{\mathrm{x}}_t^-, \Sigma_{\hat{\mathrm{x}}_t^-})^\top\hat{\theta}_t$ With $C(\cdot)$ according to Equation (3-16)
9: $\quad \hat{\mathrm{x}}_t \leftarrow \hat{\mathrm{x}}_t^- + \Sigma_{\hat{\mathrm{x}}_t^-}c_t^\top\hat{\phi}_t^- \frac{\mathrm{y}_t - \hat{\phi}_t^{-\top}\hat{\theta}_t}{\hat{\phi}_t^{-\top}\Sigma_{\hat{\theta}_t}\hat{\phi}_t^- + r_t}$
10: $\quad \Sigma_{\hat{\mathrm{x}}_t} \leftarrow \Sigma_{\hat{\mathrm{x}}_t^-} - \Sigma_{\hat{\mathrm{x}}_t^-}c_t^\top\frac{1}{\hat{\phi}_t^{-\top}\Sigma_{\hat{\theta}_t}\hat{\phi}_t^- + r_t}c_t\Sigma_{\hat{\mathrm{x}}_t^-}$
11: $\quad \hat{\mathrm{x}}_{t+1}^- \leftarrow \mathrm{A}_t\hat{\mathrm{x}}_t + \mathrm{B}_t\mathrm{u}_t$
12: $\quad \Sigma_{\hat{\mathrm{x}}_{t+1}^-} \leftarrow \mathrm{A}_t\Sigma_{\hat{\mathrm{x}}_t}\mathrm{A}_t^\top + \Sigma_\mathrm{w}$
13: **end for**

Sequential processing of the dataset significantly simplifies the DBS computation, in particular the evaluation of $\Sigma_{\hat{\Phi}_\mathrm{s}}^{(k)}$ (step 11 of Algorithm 1). In the batch formulation, each iteration requires evaluating the cross-covariances between all basis functions at all pairs of time steps, i.e., computing $\Sigma_\phi^{tt'(k)}$ for all $t, t' = 1, \ldots, T$. This computation is greatly simplified in the recursive implementation, which only computes the DBS at the current time step and therefore requires evaluating a single covariance per update, namely Σ_ϕ^{tt}. Consequently, the computations of $\mathcal{S}^{tt'}(\Sigma_{\hat{\theta}_\mathrm{nl}}^{(k-1)})$ and $\mathcal{M}^{tt'}(\Sigma_{\hat{\Phi}}^{(k)})^{(k)}$ (steps 4 and 7 of Algorithm 1, respectively) are also significantly reduced. However, the recursive implementation ignores the correlations between past and future measurements, which might be detrimental to algorithm performance. To investigate the performance of the proposed recursive algorithm another Monte-Carlo experiment is conducted.

6-4 Monte Carlo Experiment

Experimental Configuration

The same seabed configuration from Section 5-1 was used. The number of unknown parameters is set to $N = 11$, the ten different frequency vectors are, $f_0 = [0, 0]^\mathsf{T}$, $f_n = [n\frac{2\pi}{10}, 0]^\mathsf{T}$ for $n \in \{1, 2, 3\}$, and $f_n = [(n-7)\frac{2\pi}{10}, \frac{2\pi}{6}]^\mathsf{T}$ for $n \in \{4, \ldots, 10\}$.

As specified by Equation (2-6), the unknown parameters follow a uniform distribution. The lower bound is set to $a = 2$ and the upper bound is $b = 8$. Therefore, the distribution for the unknown parameters is $p(\theta) = \mathcal{U}([2, 8]^{11})$, which has mean $\mu_{\theta_n} = 5$ and covariance $\sigma_{\theta_n}^2 = 3$.

The state-transition and input matrices are given by $\mathrm{A}_t = \mathbb{I}$ and $\mathrm{B}_t = \Delta t, \mathbb{I}$, with $\Delta t = 0.1$. As defined in Section 2-1, the process and measurement noise are Gaussian. Two different variances for the process noise are tested, the values are $\Sigma_{\mathrm{w}} = \sigma_{\mathrm{w}}^2 \mathbb{I}$ where, $\sigma_{\mathrm{w}}^2 = \{0.001, 0.01\}$. The measurement noise variance was kept consistent at $\sigma_{\mathrm{v}}^2 = 0.01$ across all experiments. Finally, the initial position of the Autonomous Underwater Vehicle (AUV) is specified using a Gaussian distribution with mean $\mu_{\mathrm{x}_0} = [3.2, 2.8]^{\mathsf{T}}$ and covariance Σ_{w}.

A sequence of inputs $\bar{\mathrm{u}}^*$ of length $T = 10^5$ was selected such that $\mathrm{u}_t^* = 4.5 \sum_{\upsilon \in \Upsilon} [\cos(\upsilon t),\ \sin(\upsilon t)]^{\mathsf{T}}$, with $\Upsilon \in \{3, 5, 10, 20, 100\}$. To generate 100 different state trajectories, 100 different realizations of process noise $\{\bar{\mathrm{w}}^i\}_{i=1}^{100}$ were sampled for both values of σ_{w}^2. The state trajectories were created according to Equation (3-3), i.e.

$$\bar{\mathrm{x}}^i = \bar{\mathrm{A}}(\bar{\mathrm{B}}\bar{\mathrm{u}}^* + \bar{\mathrm{w}}^i)$$

The same 100 realizations of θ from Section 5-1 were used.

Finally, 100 different realizations of measurement noise were generated, i.e $\{\bar{\mathrm{v}}^i\}_{i=1}^{100}$. These were combined into 10,000 different observation sequences according to

$$\bar{\mathrm{y}}^{i,j} = \Phi(\bar{\mathrm{x}}^i)^\top \theta^j + \bar{\mathrm{v}}^i, \quad i, j = 1, \ldots, 100,$$

That is, for each state trajectory i, a single measurement noise realization $\bar{\mathrm{v}}^i$ was reused across all parameter realizations j. Having specified how the synthetic dataset was created, the configuration of the algorithms is now provided.

Benchmark Algorithm Configuration

- **dual state-parameter parallel-point filter:** Algorithm 6 does not require any tuning parameters.
- **dual BPF-KF filter:** As explained in Appendix A-1, the BPF (Algorithm 7) requires tuning of the number of particles, N_f. This was set to 6000 to ensure an accurate approximation.

Results

	Method	Mean	Std	Median	Q1	Q3	IQR
$\sigma_{\mathrm{w}}^2 = 0.001$	dual state-parameter	0.0005	0.0013	0.0002	0.0001	0.0005	0.0004
	dual BPF-KF	0.9983	19.4941	0.0285	0.0109	0.0854	0.0745
$\sigma_{\mathrm{w}}^2 = 0.01$	dual state-parameter	9.2179	51.7310	0.0037	0.0018	0.0081	0.0063
	dual BPF-KF	99.5941	161.3111	2.7550	0.7559	166.5449	165.7890

Table 6-1: Comparison of final parameter estimation accuracy between dual state-parameter and dual BPF-KF filters.

Figure 6-4 shows the evolution of $||\theta - \hat{\theta}_t||^2$ as a function of t across all 10,000 simulations. The solid lines represent the mean error at each time step t, the shaded regions show the central 96% percentile interval (2nd–98th percentiles). Figure 6-4 shows the distribution of

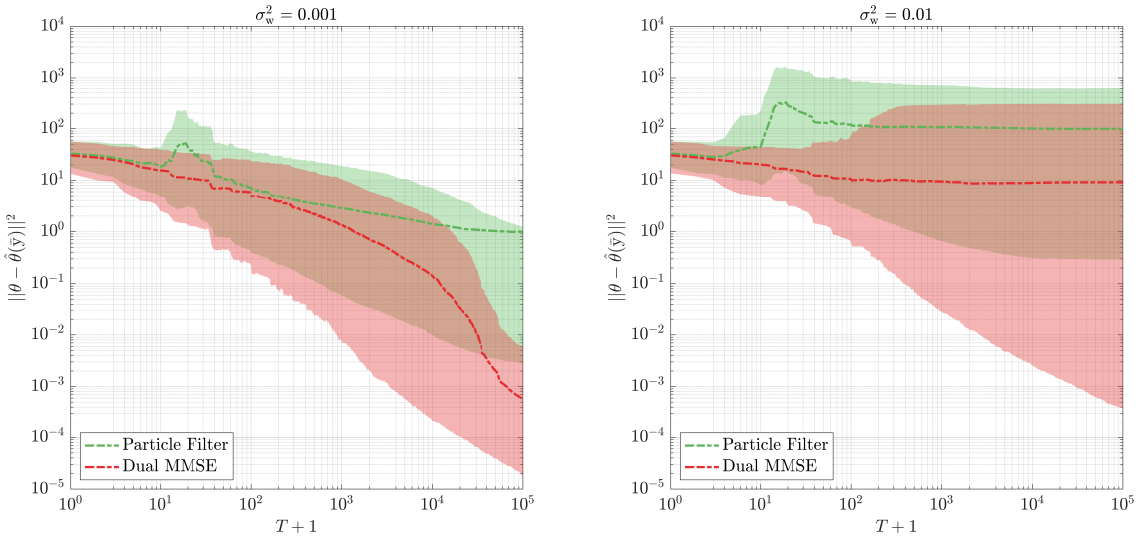

Figure 6-3: Temporal evolution of squared parameter estimation error across 10,000 Monte Carlo simulations for the dual state-parameter parallel-point filter (red) and dual BPF-KF filter (green).

final errors at time $t = T$ using box plots. The box represents interquartile range, the centre line indicates the median, the cross notes the mean.

In the low process-noise case $\sigma_w^2 = 0.001$, the dual state-parameter parallel-point filter demonstrates consistent convergence. The mean error decreases monotonically from approximately 10^1 at $t = 1$ to 10^{-3} by $t \approx 10^5$. For the final estimates, the interquartile range is narrow (0.0004), indicating that the vast majority of realizations converge. Additionally, the median (0.0002) and mean (0.0005) are close and the standard deviation is small (0.0013) indicating that nearly all experiments have converged. As shown in Figure 6-4, the dual state-parameter estimator produces a tightly concentrated error distribution.

In contrast, the dual BPF-KF filter exhibits substantially worse performance. While the mean error decreases initially, it plateaus around 10^0 after $t \approx 10^3$. More critically, the interquartile range is much wider, indicating significant variability across realizations. This is reflected in Table 6-1, where the mean error of 0.9983 is nearly three orders of magnitude larger than the median of 0.0285, with a standard deviation of 19.4941. This heavy-tailed distribution indicates that while some BPF-KF runs achieve reasonable accuracy, a substantial fraction diverge. As shown in Figure 6-4, the BPF-EKF has a significant amount trajectories that diverge.

When process noise increases to $\sigma_w^2 = 0.01$, the performance gap between the two methods widens significantly. However, the dual-state parameter is also subject the diverging trajectories, consequently the mean error is quite large (9.2179). The number of trajectories diverging is limited, indicated by the Q3 (0.0081) and median (0.0037). Despite these occasional failures, the dual filter remains effective for the vast majority of cases.

In contrast, the dual BPF-KF filter suffers systematic degradation in the high-noise regime. The mean error diverges, reaching 10^2 by $t = 10^5$. The interquartile range expands dramatically, spanning multiple orders of magnitude, and the distribution becomes extremely heavy-tailed with a mean of 99.5941, median of 2.7550, and IQR of 165.7890.

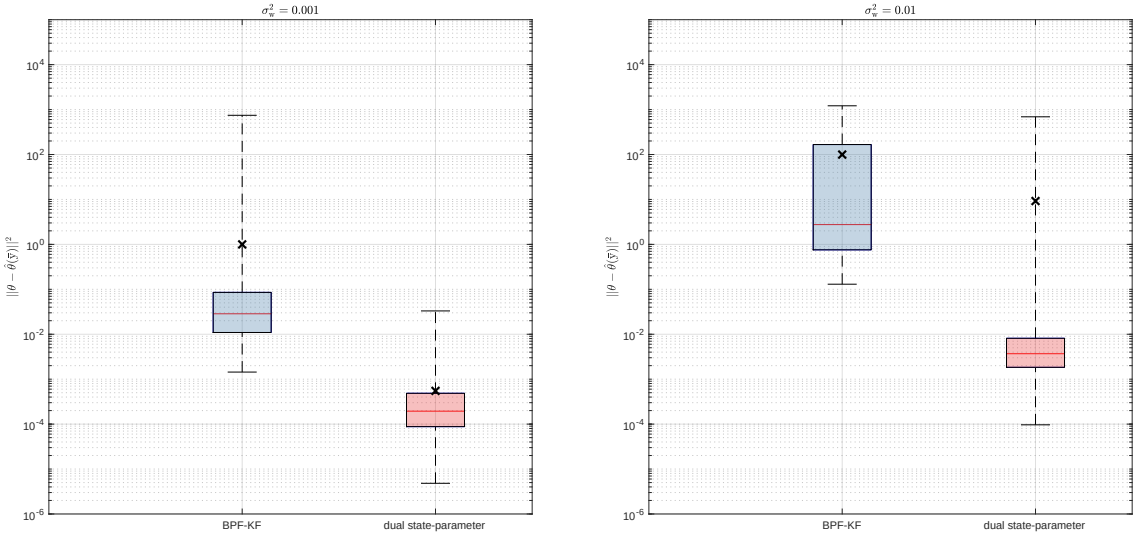

Figure 6-4: Distribution of final squared error (at $t = 10^5$) across 10,000 simulations.

The fundamental difference in performance stems from the structural advantages of the dual state-parameter parallel-point filter. The dual state-parameter parallel-point filter performs better because it recursively accounts for uncertainty in both state and parameter estimates, providing consistent updates over time. In contrast, the dual BPF-KF does not propagate parameter uncertainty adequately, and its particle-based state approximation degrades over long sequences, leading to variable or diverging estimates.

Overall, the Monte Carlo study demonstrates that the dual state-parameter parallel-point filter achieves superior performance compared to the dual BPF-KF filter for long-horizon online estimation. Across all 10,000 simulations and both noise regimes, the parallel-point filter exhibits consistent convergence, low final errors, and narrow error distributions (Table 6-1). While the dual BPF-KF filter occasionally achieves comparable accuracy in the low-noise case, its performance is highly variable and degrades catastrophically as either process noise or time horizon increases.

Beyond accuracy, the dual state-parameter filter offers substantial computational advantages: it requires only constant per-step operations with $\mathcal{O}(1)$ memory, whereas the BPF-KF filter requires propagating and resampling N_f particles at each time step. This combination of superior accuracy, and computational efficiency establishes the dual state-parameter parallel-point filter as the preferred method for long-horizon online joint state-parameter estimation in this application domain.

6-5 Chapter Conclusion

This chapter addressed the scalability limitations of the dual state-parameter estimator by developing a recursive formulation. The resulting parallel point filter processes measurements sequentially, reducing complexity from $\mathcal{O}(T^3)$ to $\mathcal{O}(T)$.

Monte Carlo experiments on long data records ($T = 10^5$) demonstrated that the parallel point filter achieves superior performance compared to the particle filter-based dual BPF-KF

approach. In the low process-noise regime ($\sigma_{\mathrm{w}}^2 = 0.001$), the parallel point filter exhibited consistent convergence with median final error of 0.0002, while the dual BPF-KF displayed high variability (median 0.0285, mean 0.9983). This performance gap widened substantially in the high process-noise regime ($\sigma_{\mathrm{w}}^2 = 0.001$), where the dual BPF-KF experienced systematic degradation with mean error exceeding 99, compared to a median error of 0.0037 for the parallel point filter.

Overall, the recursive formulation enables the dual state-parameter estimator to be applied efficiently to large data records. However, when increasing the process noise the method appears to diverge for some cases. Investigating a method to prevent this issue is addressed in the next chapter in the context of future work.

Chapter 7

Discussion

The final chapter of this thesis reflects on the content presented in the previous chapters. The research goals, as defined in Section 1-1, are revisited in Section 7-1. The most promising directions for future work are outlined in Section 7-2.

7-1 Reflection on Research Goals

1) Model-Driven Representation of the Seabed and Problem Formulation

The first objective of this thesis was to provide a rigorous formulation of the bathymetric learning problem using a model-driven representation of the seabed, which was the topic of Chapter 2.

The seabed model was formulated as a linear combination of $N + 1$ known basis functions, with the Fourier basis investigated in this thesis. This approach provides a flexible and interpretable representation, allowing variations at different scales through the choice of frequency components. However, this formulation also has inherent limitations. Accurately representing complex seabeds requires a large number of basis functions N, which increases the dimensionality of the estimation problem. Moreover, this thesis only explored Fourier features, which, while convenient for analytical computation, may not be optimal for all seabed geometries. Alternative basis functions, such as wavelets, could potentially offer more compact representations, particularly for localized or highly irregular seabed structures.

The integration of the seabed model with the AUV dynamics through a Wiener model provides a convenient state-space abstraction for linking control inputs, latent states, and observations. The linear time-varying assumption for the AUV process model may be reasonable in many scenarios, but its accuracy was largely unexamined in this thesis. In practice, the true AUV dynamics can exhibit nonlinearities, unmodeled disturbances, and variations in hydrodynamic effects that are not captured by this model class. The probabilistic model, which assumes additive Gaussian process noise with known and constant covariance, further simplifies reality, as process noise may be time- or state-dependent, and the assumed covariance may not reflect

true dynamic uncertainties. These simplifications can propagate through the state estimates and impact the quality of the inferred bathymetric map. As currently formulated, the model does not include mechanisms to adapt or correct for such mismatches.

The model based representation of the seabed and the Wiener model structure enables the bathymetric learning problem to be cast as a Bayesian inference problem. The Bayesian inference perspective provides access to a rich and well-established body of methodological tools and highlights the flexibility and expressive power of state-space models. Overall, the proposed formulation offers a principled and extensible foundation for addressing seabed estimation problems. With this, the first research goal, has been addressed.

2) Development of dual state-parameter estimator

The development of the dual state-parameter estimator constitutes the core contribution of this thesis. The central idea, decomposing the augmented estimation problem into tractable affine MMSE subproblems for states and parameters, exploits the structure of the Wiener model in a principled way, enabling a practical solution to an otherwise intractable Bayesian inference problem.

Despite its empirical effectiveness, several theoretical questions remain unresolved. Most importantly, the relationship between the fixed-point solution produced by Algorithm 1 and the true MMSE estimator is unknown. Although the algorithm alternates between affine MMSE updates, their composition does not generally correspond to the MMSE estimator of the original problem. The existence and uniqueness of fixed points have not been established, and formal convergence guarantees are absent beyond empirical observations. At the time of writing, it remains unclear under which conditions the dual state-parameter accurately represents the true posterior mean and in which cases it fails.

Finally, the recursive formulation in Chapter 6 addresses the computational limitations of the batch estimator by reducing complexity from $\mathcal{O}(T^3)$ to $\mathcal{O}(T)$ through the use of current-time DBS only. This efficiency gain comes at the cost of estimation accuracy, the high-noise filtering experiments reveal occasional divergence. For the recursive algorithm important theoretical questions also remain unknown.

3) Validation and Benchmarking in Simulation

To assess the efficacy of the proposed estimation framework, three simulation experiments were conducted. The batch estimation approach was evaluated in Chapter 5, while the recursive method was evaluated in Chapter 6. Overall, the simulation results are promising. However, the scope of the experimental validation remains limited.

In Section 5-1, Algorithm 1 was compared against two state-of-the-art methods. In this experiment, the proposed approach clearly outperformed the benchmark algorithms. While this result represents a meaningful first step, the experimental setup was limited in scope. To better assess performance in practical seabed mapping scenarios, the methods should be evaluated for larger parameter dimensions ($N >> 11$), under a wider range of input trajectories (varying $\bar{\mathrm{u}}$), and across more diverse noise conditions. Similar limitations apply to the experiment presented in Section 6-5. More extensive experiments, even under idealized

conditions, would provide deeper insight into the relative algorithmic performance. Similar critique applies for the recursive experiment in Section 6-4.

A comparable limitation applies to the simulator-based experiment discussed in Section 5-2. In particular, the evaluation relies on simplified simulation assumptions. Ocean currents were neglected, perfect orientation knowledge was assumed, and the sensor measurement models were only approximate. Furthermore, plant-model mismatch was not investigated.

A practical direction for future research is therefore to enhance the realism of the simulation environment. This could include implementing a realistic pose estimator, testing on more complex seabed geometries that outside the model class $h_\theta(\cdot)$ and incorporating more accurate IMU and depth sensor models. Such simulator enhancements would enable a more rigorous assessment of the proposed framework's suitability for practical deployment in autonomous underwater exploration.

7-2 Future Work

The developed dual state-parameter estimator demonstrates the potential to be used alongside existing Bayesian inference techniques (when applied to Wiener models). Building on these findings, the next section outlines directions for future research.

Efficient Implementation using Tensor Networks.

Recall that the original algorithm presented in Chapter 3 is ill-suited for large data records, i.e., when T is large. In particular, its computational complexity is $\mathcal{O}(T^3)$. Chapter 6 introduced a recursive version of the algorithm that alleviates this computational burden. This improvement in efficiency, however, comes at the cost of reduced performance, since the recursive formulation cannot exploit correlations across the entire dataset. Importantly, recursive estimation is not the only means of achieving scalability.

As an alternative, scalability can be achieved by constructing a low-rank tensor approximation of the kernel weights, as proposed in [55] and later applied in a Bayesian inference setting in [30]. A promising avenue for future research is to investigate whether the proposed dual state-parameter estimator can be extended through low-rank tensor approximations of the output map. Such an extension would require the development of an efficient tensor-based computation of the Dynamic Basis Statistics (DBS).

A low-rank tensor approximation of the kernel weights offers two key advantages. First, it preserves the performance benefits of batch algorithms by retaining access to long-range correlations in the data. Second, it enables efficient estimation in models with a large number of parameters, thereby addressing another fundamental limitation of the presented method.

Extension to Distributed AUVs

An interesting extension of the proposed estimation framework is to enable distributed bathymetric mapping using multiple AUVs operating simultaneously. This distributed formulation offers several practical advantages: increased coverage and parallel data collection. However,

extending the dual state-parameter estimator to a distributed setting introduces significant technical challenges.

Recursive Active Learning

In the state-space formulation of the bathymetric learning problem, the ultimate goal is to estimate the unknown parameters of the seabed model. This thesis focused exclusively on the estimation methodology, the algorithms used to compute the parameter vector. While the choice of estimation algorithm is arguably the single most important factor affecting performance, it is not the only one.

Another crucial factor is experiment design. Improving the quality of the seabed estimate does not necessarily require a better estimation algorithm; it can also be achieved by increasing the information content of the data. In the context of this thesis, this involves selecting an input sequence that maximizes the performance of existing algorithms. This process is referred to as active learning.

Due to the closed-form solution of the covariance matrix, the affine parameter estimator enables the optimization of an input sequence that minimizes the MMSE criterion [51, Section 5]. However, this approach is inherently batch-based and does not scale well to long horizons. Moreover, it remains unclear whether a similar strategy can be adapted to the proposed dual estimation framework. Investigating this extension could open the door to a recursive, adaptive active learning procedure, an advantage not available in other estimation frameworks.

Benchmarking Against Variational Inference

Chapter 4 positioned the proposed method within the data augmentation framework, with comparisons to the Gibbs sampler, the Expectation Maximization (EM) algorithm, and dual smoothing approaches. While these methods have long been central to Bayesian inference for state-space models, recent years have seen a growing interest in Variational Inference (VI) techniques, driven by their favorable scalability and optimization-based formulation. An important direction for future research is therefore to benchmark the proposed method against state-of-the-art VI approaches.

The central idea of VI is to approximate an intractable target distribution by a tractable, parameterized family of distributions. The parameters of this family are obtained by minimizing the Kullback–Leibler (KL) divergence between the variational approximation and the true posterior [5]. In the context of state-space models, both the filtering and smoothing distributions can be jointly parameterized and optimized, yielding scalable inference algorithms [40, 17, 16].

The philosophy of approximating a complex distribution by a simpler, tractable representation is also central to the proposed method. However, while VI relies on KL-divergence minimization, the proposed approach is derived from a restriction on the Minimum Mean Square Error (MMSE) estimator. This fundamental difference raises an interesting research question regarding the precise relationship between the proposed method and variational inference. In particular, it would be valuable to investigate whether the proposed framework can be interpreted as a variational method under a specific choice of variational family or divergence measure, or whether it represents a fundamentally distinct approximation paradigm.

Appendix A

SMC

The goal of this Appendix is to provide a self-contained reference for the Sequential Monte Carlo (SMC) methods required by the benchmark algorithms presented in Chapter 4. Both benchmark algorithms utilize SMC techniques: the Particle Gibbs (PG) algorithm (Algorithm 3) is based on the Conditional Particle Filter (CPF) (Algorithm 8), whereas the Particle Smoothing EM (PSEM) algorithm (Algorithm 5) builds upon the Forward Filtering Backward Simulation (FFBS) smoother (Algorithm 9).

SMC methods, are among the most extensively studied nonlinear filtering/smoothing algorithms, with numerous rigorous treatments available [20, 14, 49, 11]. Accordingly, this appendix emphasizes the algorithmic mechanics rather than theoretical properties, directing to references for in-depth analysis.

A-1 Bootstrap Particle Filtering

The foundation of all presented SMC methods is the Bootstrap Particle Filter (BPF), which approximates the filtering distribution of nonlinear state-space models when it cannot be computed in closed form.The BPF exploits the temporal structure of state-space models to recursively approximate the filtering equations, which are stated here for completeness.

$$p(\mathrm{x}_t \mid \mathrm{y}_{0:t-1}) = \int p(\mathrm{x}_t \mid \mathrm{x}_{t-1})\, p(\mathrm{x}_{t-1} \mid \mathrm{y}_{0:t-1})\, d\mathrm{x}_{t-1}, \tag{A-1}$$

$$p(\mathrm{x}_t \mid \mathrm{y}_{0:t}) = \frac{p(\mathrm{y}_t \mid \mathrm{x}_t)\, p(\mathrm{x}_t \mid \mathrm{y}_{0:t-1})}{\int p(\mathrm{y}_t \mid \mathrm{x}_t)\, p(\mathrm{x}_t \mid \mathrm{y}_{0:t-1})\, d\mathrm{x}_t}. \tag{A-2}$$

The underlying principle of SMC methods is to approximate intractable continuous densities with empirical distributions of weighted particles. For instance, the filtering distribution, $p(\mathrm{x}_t \mid \mathrm{y}_{0:t})$, is approximated by an empirical distribution of N_f weighted particles:

$$\hat{p}(\mathrm{x}_t \mid \mathrm{y}_{0:t}) = \sum_{i=1}^{N_f} \nu_t^{(i)}\, \delta_{\mathrm{x}_t^{(i)}}(\mathrm{x}_t), \tag{A-3}$$

where each particle $\mathrm{x}_t^{(i)}$ represents a candidate state at time t, and its weight $\nu_t^{(i)}$ reflects the likelihood of the observations given that state. By propagating these particles according to the filtering equations in (A-1), it is possible to recursively extend this approach to sequentially solve the filtering problem at subsequent time steps, thereby obtaining an approximation of the filtering distribution at each time. Under standard regularity conditions, this empirical approximation converges to the true filtering distribution as the number of particles N_f approaches infinity.

At the initial time $t = 0$, the particle set is generated by drawing N_f independent samples from the prior distribution $p(\mathrm{x}_0)$. The corresponding importance weights are then computed as

$$\nu_0^{(i)} = p(\mathrm{y}_0 \mid \mathrm{x}_0^{(i)}), \quad \bar{\nu}_0^{(i)} = \frac{\nu_0^{(i)}}{\sum_{j=1}^{N_f} \nu_0^{(j)}}, \tag{A-4}$$

where $\bar{\nu}_0^{(i)}$ denotes the normalized weight of particle i. Normalization is required to ensure that the weighted particles form a proper probability distribution, i.e., that the weights sum to one. This guarantees that the empirical approximation is a valid representation of the filtering distribution.

Using the empirical approximation from Equation A-3, the filtering distribution at $t = 0$ is then represented by the weighted particle set:

$$\hat{p}(\mathrm{x}_0 \mid \mathrm{y}_0) = \sum_{i=1}^{N_f} \bar{\nu}_0^{(i)} \, \delta_{\mathrm{x}_0^{(i)}}(\mathrm{x}_0). \tag{A-5}$$

These particles are propagated through the filtering equations recursively. Suppose that at time $t-1$ the filtering distribution has been approximated by a weighted particle set:

$$\hat{p}(\mathrm{x}_{t-1} \mid \mathrm{y}_{0:t-1}) = \sum_{i=1}^{N_f} \bar{\nu}_{t-1}^{i} \, \delta_{\mathrm{x}_{t-1}^{i}}(\mathrm{x}_{t-1}). \tag{A-6}$$

Substituting this approximation into the prediction step of (A-1) gives an empirical representation of the predictive distribution:

$$\hat{p}(\mathrm{x}_t \mid \mathrm{y}_{0:t-1}) = \sum_{i=1}^{N_f} \bar{\nu}_{t-1}^{i} \, p(\mathrm{x}_t \mid \mathrm{x}_{t-1}^{i}). \tag{A-7}$$

To concentrate the particles to the most probable regions of the state space, a resampling step is performed. Ancestor indices a_t^i are drawn according to the normalized weights:

$$\mathbb{P}(a_t^i = j) = \bar{\nu}_{t-1}^{j}, \quad j = 1, \ldots, N_f. \tag{A-8}$$

The resampled particles are then propagated through the system dynamics and re-weighted by the likelihood of the current observation:

$$\mathrm{x}_t^i \sim p(\mathrm{x}_t \mid \mathrm{x}_{t-1}^{a_t^i}), \quad \nu_t^i = p(\mathrm{y}_t \mid \mathrm{x}_t^i), \quad \bar{\nu}_t^i = \frac{\nu_t^i}{\sum_{j=1}^{N_f} \nu_t^j}. \tag{A-9}$$

This procedure yields a particle-based approximation of the filtering distribution at time t:

$$\hat{p}(\mathrm{x}_t \mid \mathrm{y}_{0:t}) = \sum_{i=1}^{N_f} \bar{\nu}_t^i \, \delta_{\mathrm{x}_t^i}(\mathrm{x}_t). \tag{A-10}$$

The BPF, which implements these steps, is summarized in Algorithm 7. The computational complexity of the filter is $\mathcal{O}(TN_f)$, where T is the length of the data record and N_f is the number of particles, since each particle must be propagated, weighted, and potentially resampled at every time step.

The number of particles N_f is a critical tuning parameter: increasing N_f improves the approximation of the filtering distribution but also increases computational cost. A fundamental limitation of SMC methods is the curse of dimensionality, which causes particle degeneracy and makes it challenging to accurately represent the filtering distribution in high-dimensional state spaces. As a result, SMC methods can struggle with large data records or complex models, since most particles eventually carry negligible weight and only a few effectively contribute to the approximation.

Algorithm 7 Bootstrap Particle Filter (BPF)

Ensure: Weighted particle approximation $\{(\mathrm{x}_t^i, \bar{\nu}_t^i)_{i=1}^{N_f}\}_{t=0}^{T}$ of $\{p(\mathrm{x}_t \mid \mathrm{y}_{1:t})\}_{t=0}^{T}$

$\mathrm{x}_0^i \sim p(\mathrm{x}_0)$
$\nu_0^i \leftarrow p(\mathrm{y}_0 \mid \mathrm{x}_0^i)$
$\bar{\nu}_0^i \leftarrow \frac{\nu_0^i}{\sum_{j=1}^{N_f} \nu_0^j}$
for $t = 1 \dots T$ **do**
    Sample a_i^t with $\mathbb{P}(a_t^i = j) = \bar{\nu}_{t-1}^j$
    $\mathrm{x}_t^i \sim p(\mathrm{x}_t \mid \mathrm{x}_{t-1}^{a_t^i})$
    $\nu_t^i \leftarrow p(\mathrm{y}_t \mid \mathrm{x}_t^i)$
    $\bar{\nu}_t^i \leftarrow \frac{\nu_t^i}{\sum_{j=1}^{N_f} \nu_t^j}$
end for

A-2 Conditional Particle Filter Update

The CPF is a simple extension of the BPF in which one particle trajectory is fixed to a reference path throughout the filtering process. This is a technical requirement in the PG algorithm to maintain proper Markov chain invariance.

Formally, given a reference trajectory $\bar{\mathrm{x}}'$, the CPF constructs a particle set at each time step such that the N-th particle is always equal to the corresponding state of the reference trajectory, $\mathrm{x}_t^N = \mathrm{x}_t'$, while the remaining $N-1$ particles are sampled and weighted as in the standard BPF. At the final time, a new trajectory is selected by sampling one particle according to the normalized weights. The full trajectory is then reconstructed by tracing

back through the ancestor indices a_t^i used during resampling, effectively following the lineage of the chosen particle back to time $t = 0$.

Algorithm 8 summarizes the procedure. Its computational complexity is the same as the BPF.

Algorithm 8 CPF Update

Require: Reference trajectory $\bar{\mathrm{x}}'$, parameters θ, observations $\bar{\mathrm{y}}$, particles N_f

Ensure: New reference trajectory $\bar{\mathrm{x}}^*$

$\mathrm{x}_0^i \sim p(\mathrm{x}_0)$

$\nu_0^i \leftarrow p(\mathrm{y}_0 \mid \mathrm{x}_0^i)$

$\bar{\nu}_0^i \leftarrow \frac{\nu_0^i}{\sum_{j=0}^{N_f} \nu_0^j}$

for $t = 1$ to T **do**

  $a_t^i \sim \bar{\nu}_{t-1}$, $i = 1, \ldots, N_f - 1$

  $\mathrm{x}_t^i \sim p(\mathrm{x}_t \mid \mathrm{x}_{t-1}^{a_t^i})$, $i = 1, \ldots, N_f - 1$

  $\mathrm{x}_t^{N_f} = \mathrm{x}_t'$

  $\nu_t^{N_f} \leftarrow p(\mathrm{y}_t \mid \mathrm{x}_t^{N_f})$

  $\bar{\nu}_t^i \leftarrow \frac{\nu_t^i}{\sum_{j=1}^{N_f} \nu_t^j}$

end for

Sample $i_T \sim \bar{\nu}_T$ $\mathbb{P}(i = j) = \bar{\nu}_T^j$

$\mathrm{x}_T^* = \mathrm{x}_T^{i_T}$; reconstruct x_t^* from ancestors

A-3 Forward Filtering Backward Smoothing

Particle smoothers extend the SMC framework to approximate the smoothing distribution of state-space models. While particle filters such as the BPF provide an empirical approximation of the filtering distribution, particle smoothers aim to approximate the marginal smoothing distributions

$$p(\mathrm{x}_t \mid \bar{\mathrm{y}}), \quad t = 0, \ldots, T, \tag{A-11}$$

which incorporate information from the entire sequence of observations, including future measurements.

The FFBS smoother represents the smoothing distribution using an empirical distribution of N_s weighted particles:

$$\hat{p}(\mathrm{x}_t \mid \mathrm{y}_{0:T}) = \sum_{i=1}^{N_s} \bar{\nu}_{t|T}^{(i)} \, \delta_{\mathrm{x}_t^{(i)}}(\mathrm{x}_t), \tag{A-12}$$

where each particle $\mathrm{x}_t^{(i)}$ represents a candidate state at time t, and $\bar{\nu}_{t|T}^{(i)}$ is the normalized smoothing weight.

The smoothing recursion leverages both the forward filtering approximation and the system dynamics. Formally, the smoothing density at time t can be written as

$$p(\mathrm{x}_t \mid \mathrm{y}_{0:T}) = p(\mathrm{x}_t \mid \mathrm{y}_{0:t}) \int \frac{p(\mathrm{x}_{t+1} \mid \mathrm{x}_t)\, p(\mathrm{x}_{t+1} \mid \mathrm{y}_{0:T})}{p(\mathrm{x}_{t+1} \mid \mathrm{y}_{0:t})}\, d\mathrm{x}_{t+1}, \tag{A-13}$$

which shows that the smoothing distribution depends on: (i) the filtering distribution at time t, (ii) the system dynamics, and (iii) the smoothing distribution at time $t+1$.

In practice, the FFBS algorithm implements this recursion using the particles obtained from the forward filtering pass. At the final time $t = T$, the filtering and smoothing distributions coincide. Backward smoothing is then performed recursively by computing backward importance weights for each particle:

$$\nu_{t|t+1}^{(i)} = \bar{\nu}_t^{(i)}\, p(\mathrm{x}_{t+1}^* \mid \mathrm{x}_t^{(i)}), \qquad \bar{\nu}_{t|t+1}^{(i)} = \frac{\nu_{t|t+1}^{(i)}}{\sum_{j=1}^{N_s} \nu_{t|t+1}^{(j)}}, \tag{A-14}$$

where x_{t+1}^* is a sampled particle from time $t+1$. Sampling from these normalized weights yields a smoothed particle x_t^*, which is then used in the next step of the backward recursion.

Repeating this backward sampling for $t = T-1, \ldots, 0$ generates a full trajectory approximately drawn from the joint smoothing distribution. By performing multiple backward passes, one can generate multiple independent trajectories.

The FFBS algorithm is summarized in Algorithm 9. Its computational complexity is $\mathcal{O}(TN_fN_s)$, where T is the length of the time series, N_f is the number of forward filtering particles, and N_s is the number of smoothed trajectories, reflecting the cost of computing backward weights and sampling at each time step. As with the BPF, increasing the number of particles improves the approximation, but particle degeneracy and computational cost must be considered, especially in high-dimensional state spaces or long time series.

Algorithm 9 Forward Filtering Backward Smoothing (FFBS)

Require: Weighted particles from forward filtering $\{(\mathrm{x}_t^i, \bar{\nu}_t^i)_{i=1}^{N_f}\}_{t=0}^{T}$

Ensure: Weighted particles $\{(\mathrm{x}_t^i, \bar{\nu}_{t|T}^i)_{i=1}^{N_f}\}_{t=0}^{T}$ approximating $p(\mathrm{x}_t \mid \mathrm{y}_{0:T})$

1: $\bar{\nu}_{T|T}^i \leftarrow \bar{\nu}_T^i$ for $i = 1, \ldots, N_f$

2: **for** $t = T-1, \ldots, 0$ **do**

3: **for** $i = 1, \ldots, N_f$ **do**

4: Compute backward weight numerator:

$$\nu_{t|T}^{(i)} \leftarrow \bar{\nu}_t^i \sum_{j=1}^{N_f} \bar{\nu}_{t+1|T}^{(j)}\, p(\mathrm{x}_{t+1}^j \mid \mathrm{x}_t^i)$$

5: **end for**

6: Normalize backward weights:

$$\bar{\nu}_{t|T}^{(i)} \leftarrow \frac{\nu_{t|T}^{(i)}}{\sum_{j=1}^{N_f} \nu_{t|T}^{(j)}}, \quad i = 1, \ldots, N_f$$

7: **end for**

Bibliography

[1] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(3):269–342, June 2010.

[2] Stephen Barkby, Stefan Williams, Oscar Pizarro, and Michael Jakuba. Incorporating prior maps with bathymetric distributed particle slam for improved auv navigation and mapping. In *OCEANS 2009*, pages 1–7. IEEE, 2009.

[3] Robert Bassett and Julio Deride. Maximum a Posteriori Estimators as a Limit of Bayes Estimators. *Mathematical Programming*, 174(1-2):129–144, March 2019. arXiv:1611.05917 [math].

[4] Matthew J Beal and Zoubin Ghahramani. The Variational Bayesian EM Algorithm for Incomplete Data: With Application to Scoring Graphical Model Structures. In V Lindley, J M Bernardo, M J Bayarri, J O Berger, A P Dawid, D Heckerman, A Fm Smith, and M West, editors, *Bayesian Statistics 7*, pages 453–463. Oxford University PressOxford, July 2003.

[5] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017. arXiv:1601.00670 [stat].

[6] Pete Bunch and Simon Godsill. Improved particle approximations to the joint smoothing distribution using markov chain monte carlo. *IEEE Transactions on Signal Processing*, 61(4):956–963, 2013.

[7] Andrew Campbell, Yuyang Shi, Thomas Rainforth, and Arnaud Doucet. Online variational filtering and parameter learning. *Advances in Neural Information Processing Systems*, 34:18633–18645, 2021.

[8] G. Canepa, O. Bergem, and N.G. Pace. A new algorithm for automatic processing of bathymetric data. *IEEE Journal of Oceanic Engineering*, 28(1):62–77, 2003.

[9] Olivier Cappe. Online sequential Monte Carlo EM algorithm. In *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, pages 37–40, August 2009. ISSN: 2373-0803.

[10] Olivier Cappé. Online EM Algorithm for Hidden Markov Models. *Journal of Computational and Graphical Statistics*, 20(3):728–749, January 2011.

[11] Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in hidden Markow models*. Springer series in statistics. Springer, New York, 2005.

[12] Chris K Carter and Robert Kohn. On gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

[13] Nicolas Chopin, Pierre E. Jacob, and Omiros Papaspiliopoulos. SMC^2: an efficient algorithm for sequential analysis of state-space models, January 2012. arXiv:1101.1528 [stat].

[14] Nicolas Chopin and Omiros Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer Series in Statistics. Springer International Publishing, Cham, 2020. ISSN: 0172-7397, 2197-568X.

[15] Nicolas Chopin and Sumeetpal S. Singh. On particle Gibbs sampling. *Bernoulli*, 21(3), August 2015. arXiv:1304.1887 [stat].

[16] Jarrad Courts, Adrian Wills, Thomas Schön, and Brett Ninness. Variational System Identification for Nonlinear State-Space Models, September 2022. arXiv:2012.05072 [stat].

[17] Jarrad Courts, Adrian G. Wills, and Thomas B. Schön. Gaussian Variational State Estimation for Nonlinear State-Space Models. *IEEE Transactions on Signal Processing*, 69:5979–5993, 2021.

[18] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

[19] Clemens Deutsch, Lázaro Moratelli, Sebastian Thuné, Jakob Kuttenkeuler, and Filip Söderling. Design of an auv research platform for demonstration of novel technologies. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–8. IEEE, 2018.

[20] Arnaud Doucet, Nando Freitas, and Neil Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York, NY, 2001.

[21] Italo Oliveira Ferreira, Laura Coelho de Andrade, Victoria Gibrim Teixeira, and Felipe Catão Mesquita Santos. State of art of bathymetric surveys. *Boletim de Ciências Geodésicas*, 28(01):e2022002, 2022.

[22] Enric Galceran and Marc Carreras. Planning coverage paths on bathymetric maps for in-detail inspection of the ocean floor. In *2013 IEEE International Conference on Robotics and Automation*, pages 4159–4164. IEEE, 2013.

[23] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[24] Zoubin Ghahramani and Sam Roweis. Learning nonlinear dynamical systems using an em algorithm. *Advances in neural information processing systems*, 11, 1998.

[25] Simon J Godsill, Arnaud Doucet, and Mike West. Monte carlo smoothing for nonlinear time series. *Journal of the american statistical association*, 99(465):156–168, 2004.

[26] Simon Haykin. *Kalman filtering and neural networks*. John Wiley & Sons, 2004.

[27] Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear Estimation*. Prentice Hall, New Jersey, 2000.

[28] Nikolas Kantas, Arnaud Doucet, Sumeetpal S. Singh, Jan Maciejowski, and Nicolas Chopin. On Particle Methods for Parameter Estimation in State-Space Models. *Statistical Science*, 30(3):328 – 351, 2015. Publisher: Institute of Mathematical Statistics.

[29] Mart Kartašev, David Dörner, Özer Özkahraman, Petter Ögren, Ivan Stenius, and John Folkesson. Smarcsim: Maritime robotics simulation modules. *arXiv preprint arXiv:2506.07781*, 2025.

[30] Afra Kilic and Kim Batselier. Interpretable bayesian tensor network kernel machines with automatic rank and feature selection. *arXiv preprint arXiv:2507.11136*, 2025.

[31] Stanisław Konatowski, Piotr Kaniewski, and Jan Matuszewski. Comparison of estimation accuracy of ekf, ukf and pf filters. *Annual of Navigation*, (23):69–87, 2016.

[32] Yifang Li and Sujit K Ghosh. Efficient sampling methods for truncated multivariate normal and student-t distributions subject to linear inequality constraints. *Journal of Statistical Theory and Practice*, 9(4):712–732, 2015.

[33] Zhuoxiao Li, Zitian Peng, Zheng Zhang, Yijie Chu, Chenhang Xu, Shanliang Yao, Ángel F García-Fernández, Xiaohui Zhu, Yong Yue, Andrew Levers, et al. Exploring modern bathymetry: A comprehensive review of data acquisition devices, model accuracy, and interpolation techniques for enhanced underwater mapping. *Frontiers in Marine Science*, 10:1178845, 2023.

[34] Fredrik Lindsten, Michael I. Jordan, and Thomas B. Schön. Particle Gibbs with Ancestor Sampling, January 2014. arXiv:1401.0604 [stat].

[35] Bin Liu, Yan Huang, Hao Feng, Jiancheng Yu, Jianan Qiao, and Xiaolong Ju. A review of auv-based bathymetric slam technology. *Ocean Engineering*, 342:122858, 2025.

[36] Lennart Ljung and Torsten Söderström. *Theory and practice of recursive identification*. MIT press, 1983.

[37] Alessandro Mastrototaro and Jimmy Olsson. Online variational sequential monte carlo. *arXiv preprint arXiv:2312.12616*, 2023.

[38] Geoffrey J McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2008.

[39] Steven A Murawski, Sherryl Gilbert, Matthew Hommeyer, Yonggang Liu, Chad Lembke, Sarah Grasty, David English, Chuanmin Hu, Tim Dixon, and Taha Sadeghi Chorsi. Bathymetric mapping in the coastal zone: Approaches, challenges, and opportunities. *Marine Technology Society Journal*, 59(2):62–77, 2025.

[40] Christian A. Naesseth, Scott W. Linderman, Rajesh Ranganath, and David M. Blei. Variational Sequential Monte Carlo, February 2018. arXiv:1705.11140 [stat].

[41] A.T. Nelson and E.A. Wan. A two-observation Kalman framework for maximum-likelihood modeling of noisy time series. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, volume 3, pages 2489–2494 vol.3, May 1998. ISSN: 1098-7576.

[42] Lawrence Nelson and Edwin Stear. The simultaneous on-line estimation of parameters and states in linear systems. *IEEE Transactions on automatic Control*, 21(1):94–98, 1976.

[43] Ornella Nonnis, Chiara Maggi, Pasquale Lanera, Raffaele Proietti, Alessia Izzi, Pietro Antonelli, and Massimo Gabellini. The management of information to identify the submarine cable route: The case study of campania islands. *Journal of Coastal Research*, (75):1002–1006, 2016.

[44] Jimmy Olsson and Johan Westerborn. Efficient particle-based online smoothing in general hidden markov models: The paris algorithm. 2017.

[45] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[46] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, New York, NY, 2004. ISSN: 1431-875X.

[47] Thomas B. Schön, Adrian Wills, and Brett Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, January 2011.

[48] Jianzhong Sun, Hongfu Zuo, and Michael G Pecht. Advances in sequential monte carlo methods for joint state and parameter estimation applied to prognostics. In *2011 Prognostics and System Health Managment Confernece*, pages 1–7. IEEE, 2011.

[49] Simo Särkkä. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013.

[50] Martin A Tanner and Wing Hung Wong. The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398):528–540, 1987.

[51] Sasan Vakili, Manuel Mazo Jr., and Peyman Mohajerin Esfahani. Optimal bayesian affine estimator and active learning for the wiener model, 2025.

[52] Sasan Vakili, Daniël Woonings, and Peyman Mohajerin Esfahani. Nonlinear estimator: Dual bayesian affine estimators for the wiener model, 2025.

[53] Eric Wan and Alex Nelson. Dual Kalman Filtering Methods for Nonlinear Prediction, Smoothing and Estimation. In *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.

[54] Lei Wang, Hongxing Liu, Haibin Su, and Jun Wang. Bathymetry retrieval from optical images with spatially distributed support vector machines. *GIScience & remote sensing*, 56(3):323–337, 2019.

[55] Frederiek Wesel and Kim Batselier. Large-scale learning with fourier features and tensor decompositions. *Advances in Neural Information Processing Systems*, 34:17543–17554, 2021.

[56] Adrian Wills, Thomas B. Schön, Lennart Ljung, and Brett Ninness. Identification of hammerstein–wiener models. *Automatica*, 49(1):70–81, 2013.

[57] CF Jeff Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.

[58] Russell B. Wynn, Veerle A.I. Huvenne, Timothy P. Le Bas, Bramley J. Murton, Douglas P. Connelly, Brian J. Bett, Henry A. Ruhl, Kirsty J. Morris, Jeffrey Peakall, Daniel R. Parsons, Esther J. Sumner, Stephen E. Darby, Robert M. Dorrell, and James E. Hunt. Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience. *Marine Geology*, 352:451–468, 2014. 50th Anniversary Special Issue.

Glossary

List of Acronyms

GPS	Global Positioning System
AUV	Autonomous Underwater Vehicle
MAP	Maximum a Posteriori
MMSE	Minimum Mean Square Error
MSE	Mean Square Error
EM	Expectation Maximization
PG	Particle Gibbs
SMC	Sequential Monte Carlo
PGAS	Particle Gibbs with Ancestor Sampling
FFBS	Forward Filtering Backward Simulation
BPF	Bootstrap Particle Filter
DBS	Dynamic Basis Statistics
PSEM	Particle Smoothing EM
ROS	Robot Operating System
MQTT	Message Queuing Telemetry Transport
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
CPF	Conditional Particle Filter
VI	Variational Inference