Dynamic rEvolution
Adaptive state estimation via Gaussian processes and iterative filtering
Anton Kullberg
Dynamic rEvolution

Adaptive state estimation via Gaussian processes and iterative filtering

Anton Kullberg
**Cover illustration:** A Gaussian process model of the lateral acceleration experienced by a car driving around the Nürburgring, Germany. The acceleration profile is illustrated for a part of the track, specifically Kallenhard-Wehrseifen-Ex-Mühle.

**Title:** A play on words relating to the theme of the thesis. Dynamic - the thesis studies dynamical systems; Evolution - dynamical systems evolve over time and the contributions are an evolution of the research field itself; rE(volution) - iterated algorithms are at the core of parts of the contributions.

---

Linköping studies in science and technology. Dissertations.  
No. 2391  
**Dynamic rEvolution: Adaptive state estimation via Gaussian processes and iterative filtering**  
Anton Kullberg  
anton.kullberg@liu.se  
www.control.isy.liu.se  
Division of Automatic Control  
Department of Electrical Engineering  
Linköping University  
SE–581 83 Linköping  
Sweden  

ISBN 978-91-8075-673-0 (print)  
ISBN 978-91-8075-674-7 (PDF)  
ISSN 0345-7524

Unless otherwise stated, this work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit [http://creativecommons.org/licenses/by/4.0/](http://creativecommons.org/licenses/by/4.0/).  

Copyright © 2024 Anton Kullberg  

Printed by LiU-Tryck, Linköping, Sweden 2024
“The circle is now complete.  
When I left you, I was but the learner. 
Now I am the master.”
Abstract

For virtually every area of science and engineering, state estimation is ubiquitous. Accurate state estimation requires a moderately precise mathematical model of the system, typically based on domain expertise. These models exist for a plethora of applications and available state estimators can generally produce accurate estimates. However, the models usually ignore hard-to-model phenomena, either due to the cost or the difficulty of modeling these characteristics. Further, the most widely used state estimator for nonlinear systems is still the extended Kalman filter (EKF), which may suffer from divergence for complex models, which essentially restricts the complexity of the usable models. Generally speaking, this thesis investigates ways of improving state estimation. Firstly, existing state-space models (SSMs) for target tracking are augmented with a Gaussian process (GP) in order to learn hard-to-model system characteristics online. Secondly, improved linearization-based state estimators are proposed that exhibit favorable robustness properties to the parameters of the noise processes driving the SSM.

The first part of the thesis explores joint state estimation and model learning in partially unknown SSMs, where some a priori domain expertise is available, but parts of the model need to be learned online. Paper A combines a linear, a priori identified, SSM with an approximate GP. An EKF is applied to this GP-augmented SSM in order to jointly estimate the state of the system and learn the, a priori, unknown dynamics. This empirically works well and substantially reduces the prediction error of the dynamical model as compared to a non-augmented SSM. Paper B explores ways of reducing the computational complexity of the method of Paper A. Crucially, it uses a compact kernel in the GP, which admits an equivalent basis function (BF) representation where only a few BFs are non-zero at any given system state. This enables a method that is essentially computationally invariant to the number of parameters, where the computational complexity can be tuned by hyperparameters of the BFs.

The second part explores iterated filters as a means to increase robustness to improper noise parameter choices. As the nonlinearities in the model are mainly contained in the dynamics, standard iterated filters such as the iterated extended Kalman filter (IEKF) can not be used. Papers C and D develop dynamically iterated filters (DIFs), which is a unified framework for linearization-based iterated filters that deal with nonlinearities in both the dynamics as well as the measurement model. The DIFs are shown to be robust toward improper noise parameter tuning and improve the mean square error (MSE) as compared to their corresponding non-iterated baselines.

The third and final part of the thesis considers an alternative BF representation of the GP model, the Hilbert-space Gaussian process (HGP), which is essentially a sinusoidal representation on a compact domain. Paper E identifies previously unutilized Hankel-Toeplitz structure in the HGP, which enables a time complexity for learning that is linear in the number of BFs, without further approximation. Lastly, Paper F improves the computational complexity of prediction in the HGP, by adaptively choosing the most important BFs for prediction in a certain region of the input.
Populärvetenskaplig sammanfattning


Den allra mest välkända metoden för tillståndsskattning är det så kallade Kalmanfiltret. En anledning till Kalmanfiltrets popularitet är att det kan visas vara optimalt om den matematiska modellen besitter specifika egenskaper. Kalmanfiltret är dessutom beräkningseffektivt och kan användas på väldigt enkla mikroprocessorer. T.ex. körs det varianter av Kalmanfilter lokalt på din mobiltelefon för att uppskatta hur den är roterad och vart den befinner sig. I de fall där den matematiska modellen saknar dessa specifika egenskaper så behöver man ta till approximativa metoder, som dessvärre inte längre är optimala, men som praktiskt sett fungerar i väldigt många fall. I vissa fall kan man stöta på så kallad divergens, där tillståndsskattningen blir så pass dålig att den kan orsaka en mjukvarukrasch, som i värsta fall kan leda till att människor tar skada om metoderna till exempel används i säkerhetskritiska system, så som flygplan.

Med breda drag kan man säga att denna avhandling behandlar olika metoder för att förbättra just tillståndsskattning, för att förbättra vår uppfattning om
Populärvetenskaplig sammanfattning


Den andra delen av avhandlingen studerar *direkta* förbättringar av metoder för tillståndsskattning. Här utvecklas nya metoder för tillståndsskattning som generaliserar flera tidigare metoder. Fördelen med de nya metoderna är att de dels förbättrar själva tillståndsskattningen men också att de löper mycket mindre risk att divergera. Detta innebär att de är väl lämpade för användning i säkerhetskritiska system, speciellt då de är nästan lika beräkningseffektiva som Kalmanfiltret.

Tredje och sista delen av avhandlingen tar ett steg tillbaka och tar en närmre titt på den maskininlärningsmodell som nyttjades i första delen. Här går arbetet framförallt ut på att göra modellen mer beräkningseffektiv för att möjliggöra dess användning i mer komplexa scenarier. Slutsatsen är att modellerna går att skala upp till mycket större problem med ingen eller liten försämring av kvaliteten på modellprediktionerna.
Acknowledgments

I would first like to express my gratitude to my supervisor Gustaf Hendeby and co-supervisor Isaac Skog. You have provided me with guidance and support for the technical work contained within this thesis as well as career advice moving forward, which I am ever grateful for. Even though we have had differences in opinion, I always value and appreciate your comments, regardless of the frustration they cause at times.

Thanks to the head of division, Martin Enqvist, for maintaining a friendly and welcoming work environment. Your unique ability to see things from multiple perspectives without projecting your own views is a great source of inspiration. Also, to our administrator, Ninna Stensgård, thank you for taking care of all practicalities during my time at the division.

To all of my colleagues, past and present, at the division of Automatic Control, thank you for the friendly environment, weird discussions, and occasional beers. My time as a PhD student would have been very dull without you! I am especially grateful to Daniel Arnesby, Gustav Zetterqvist, Johanna Wilroth, Magnus Malmström, and Robin Forsling, for proofreading this thesis and helping me improve it. Any remaining errors are my own. A special thanks also goes out to Daniel Arnesby, Kristin Nielsen, and Magnus Malmström. We have, in different constellations, collaborated in a variety of ways, through coursework, reading groups, etc. Our discussions and joint work have taught me a lot. To Filipe Barbosa, our first collaboration managed to severely crash the fika group economy, a great success! My time at the division would have been a lot duller without you.

To all of my collaborators: Arno Solin, Daniel Axehill, Filipe Barbosa, Frederik Wesel, Fredrik Gustafsson, Frida Viset, Henrik Nilsson, Joakim Rydell, Johan Löfberg, Magnus Malmström, Manon Kok, Martin Skoglund, and Rudy Helmons, thank you for fruitful and interesting collaborations, I have learned a lot from all of you. A special thanks to Arno Solin who welcomed me as a visiting PhD student at Aalto University. Even though the work I did while at Aalto remains dormant in my desk drawer, the time at Aalto directly influenced a lot of the work I have done after my licentiate thesis. To the PhD students and postdocs at Aalto: thank you for welcoming me into your group, it was a fun and highly educational time for me. A special thanks also to Frida Viset: our collaboration proved extremely fruitful and I hope it can continue in the future. This thesis would have taken a completely different spin without our joint work.

This work was supported by the Wallenberg AI and Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Thank you partly for financing my pursuit of a PhD, but mainly for providing ample opportunity to network with a fantastic group of people.

To my family, you have supported me throughout my 11 years in Linköping and I hope that continues in my future endeavors. Any time I get to spend with you is precious to me, even though physical distance has separated us for a long time now.
To my second family: FB, TF, JG (JAG), KG, JK (JE), KK, DS, CS, CW, TW, and LÅ. Your endless support and friendship are invaluable and irreplaceable. Our times together have indirectly improved this thesis immensely.

Lastly, the biggest appreciation goes out to my wonderful sambo Lovisa. You have withstood my frustration, complaints, rants, stress, and weird shenanigans for all of these years. You’re the best!

Linköping, May 2024
Anton Kullberg
Contents

Notation xiii
Acronyms xv

I Background

1 Introduction 3
  1.1 Background and Motivation 3
  1.1.1 Data-Driven Modeling of Dynamical Systems 5
  1.1.2 Nonlinear State Estimation 7
  1.2 Problem Statement 8
  1.3 Thesis Outline 9
    1.3.1 Publications and Contributions 9
    1.3.2 Complete List of Publications 12
    1.3.3 Source Code Availability 13

2 Function Approximation 15
  2.1 Basis Function Expansions 16
  2.2 Gaussian Processes 19
    2.2.1 Computational Aspects 22

3 State Estimation 31
  3.1 State-Space Models 31
    3.1.1 Probabilistic Modeling 32
  3.2 State Estimation 33
    3.2.1 Nonlinear State Estimation 35
    3.2.2 Iterated Filtering 39

4 Joint State Estimation and Function Approximation 43
  4.1 Learning in State-Space Models 43
  4.2 Modeling 45

5 Concluding Remarks 49
5.1 Summary of Contributions ........................................ 49
  5.1.1 Online Learning in Augmented Grey-box Models .... 49
  5.1.2 Iterated Linearization-based Filtering ................. 50
  5.1.3 Computational Efficiency in Gaussian Processes ... 51
5.2 Future Research Directions ..................................... 52

Bibliography ............................................................. 55

II Publications

A Learning Driver Behaviors Using a GPASSM ............... 71
B Online Joint State Inference and Learning of PUSSM .... 89
C Iterated Statistical Linearization and Quasi-Newton .... 123
D Dynamically Iterated Filters .................................... 137
E Exploiting Hankel-Toeplitz Structure ....................... 165
F Adaptive Basis Function Selection ............................ 197
## Notation

### Mathematical notation

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{R}$</td>
<td>Set of real numbers</td>
</tr>
<tr>
<td>$\mathbb{R}^n$</td>
<td>Set of real $n$ vectors</td>
</tr>
<tr>
<td>$\mathbb{R}^{n \times n}$</td>
<td>Set of real $n \times n$ matrices</td>
</tr>
<tr>
<td>$x$</td>
<td>Scalar</td>
</tr>
<tr>
<td>$\mathbf{x}$</td>
<td>Vector</td>
</tr>
<tr>
<td>$X$</td>
<td>Matrix</td>
</tr>
<tr>
<td>cov$(\mathbf{x})$</td>
<td>Covariance of the elements of $\mathbf{x}$</td>
</tr>
<tr>
<td>diag$(X)$</td>
<td>Diagonal of $X$</td>
</tr>
<tr>
<td>diag$(\mathbf{x})$</td>
<td>Diagonal matrix with diagonal given by $\mathbf{x}$</td>
</tr>
<tr>
<td>$\mathcal{N}(\mathbf{x}; \mu, S)$</td>
<td>(Multivariate) Normal distribution of $\mathbf{x}$ with mean $\mu$ and covariance $S$</td>
</tr>
<tr>
<td>$\mathbb{E}[\cdot]$</td>
<td>Expected value</td>
</tr>
<tr>
<td>$\mathbb{E}_{p(\cdot)}[\cdot]$</td>
<td>Expected value w.r.t. $p(\cdot)$</td>
</tr>
<tr>
<td>$\sim$</td>
<td>Distributed as</td>
</tr>
</tbody>
</table>
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF</td>
<td>basis function</td>
</tr>
<tr>
<td>BNN</td>
<td>Bayesian neural network</td>
</tr>
<tr>
<td>BRUF</td>
<td>Bayesian recursive update filter</td>
</tr>
<tr>
<td>CG</td>
<td>conjugate gradient</td>
</tr>
<tr>
<td>CKF</td>
<td>cubature Kalman filter</td>
</tr>
<tr>
<td>CSRBF</td>
<td>compact support radial basis function</td>
</tr>
<tr>
<td>CV</td>
<td>constant velocity</td>
</tr>
<tr>
<td>DFP</td>
<td>Davidon-Fletcher-Powell</td>
</tr>
<tr>
<td>DIEKF</td>
<td>dynamically iterated extended Kalman filter</td>
</tr>
<tr>
<td>DIF</td>
<td>dynamically iterated filter</td>
</tr>
<tr>
<td>DIPLF</td>
<td>dynamically iterated posterior linearization filter</td>
</tr>
<tr>
<td>DIUKF</td>
<td>dynamically iterated unscented Kalman filter</td>
</tr>
<tr>
<td>EKF</td>
<td>extended Kalman filter</td>
</tr>
<tr>
<td>EKS</td>
<td>extended Kalman smoother</td>
</tr>
<tr>
<td>ELBO</td>
<td>evidence lower bound</td>
</tr>
<tr>
<td>EM</td>
<td>expectation maximization</td>
</tr>
<tr>
<td>FIC</td>
<td>fully independent conditional</td>
</tr>
<tr>
<td>GN</td>
<td>Gauss–Newton</td>
</tr>
<tr>
<td>GP</td>
<td>Gaussian process</td>
</tr>
<tr>
<td>GPASSM</td>
<td>Gaussian process augmented state-space model</td>
</tr>
<tr>
<td>GPS</td>
<td>global positioning system</td>
</tr>
<tr>
<td>GPSSM</td>
<td>Gaussian process state-space model</td>
</tr>
<tr>
<td>HGP</td>
<td>Hilbert-space Gaussian process</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>ICKF</td>
<td>iterated cubature Kalman filter</td>
</tr>
<tr>
<td>IEKF</td>
<td>iterated extended Kalman filter</td>
</tr>
<tr>
<td>IEKS</td>
<td>iterated extended Kalman smoother</td>
</tr>
<tr>
<td>IPLF</td>
<td>iterated posterior linearization filter</td>
</tr>
<tr>
<td>IPLS</td>
<td>iterated posterior linearization smoother</td>
</tr>
<tr>
<td>IUKF</td>
<td>iterated unscented Kalman filter</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman filter</td>
</tr>
<tr>
<td>KL</td>
<td>Kullback–Leibler</td>
</tr>
<tr>
<td>KS</td>
<td>Kalman smoother</td>
</tr>
<tr>
<td>LASSO</td>
<td>least absolute shrinkage and selection operator</td>
</tr>
<tr>
<td>LSDIEKF</td>
<td>line–search dynamically iterated extended Kalman filter</td>
</tr>
<tr>
<td>MAP</td>
<td>maximum-a-posteriori</td>
</tr>
<tr>
<td>MC</td>
<td>Monte Carlo</td>
</tr>
<tr>
<td>MCMC</td>
<td>Markov chain Monte Carlo</td>
</tr>
<tr>
<td>MLL</td>
<td>marginal log-likelihood</td>
</tr>
<tr>
<td>MSE</td>
<td>mean square error</td>
</tr>
<tr>
<td>NLPD</td>
<td>negative log predictive density</td>
</tr>
<tr>
<td>ODE</td>
<td>ordinary differential equation</td>
</tr>
<tr>
<td>PF</td>
<td>particle filter</td>
</tr>
<tr>
<td>PMF</td>
<td>point mass filter</td>
</tr>
<tr>
<td>PS</td>
<td>particle smoother</td>
</tr>
<tr>
<td>PSB</td>
<td>Powell symmetric Broyden</td>
</tr>
<tr>
<td>QN</td>
<td>quasi–Newton</td>
</tr>
<tr>
<td>QN–IEKF</td>
<td>quasi–Newton iterated extended Kalman filter</td>
</tr>
<tr>
<td>RBF</td>
<td>radial basis function</td>
</tr>
<tr>
<td>RBPF</td>
<td>Rao–Blackwellized particle filter</td>
</tr>
<tr>
<td>RFF</td>
<td>random Fourier features</td>
</tr>
<tr>
<td>RMSE</td>
<td>root mean square error</td>
</tr>
<tr>
<td>RTS</td>
<td>Rauch-Tung-Striebel</td>
</tr>
<tr>
<td>SE</td>
<td>squared exponential</td>
</tr>
<tr>
<td>SGD</td>
<td>stochastic gradient descent</td>
</tr>
<tr>
<td>SKI</td>
<td>structured kernel interpolation</td>
</tr>
<tr>
<td>SL</td>
<td>statistical linearization</td>
</tr>
<tr>
<td>SLAM</td>
<td>simultaneous localization and mapping</td>
</tr>
<tr>
<td>SMC</td>
<td>sequential Monte Carlo</td>
</tr>
</tbody>
</table>
**Acronyms**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSGP</td>
<td>sparse spectrum GP</td>
</tr>
<tr>
<td>SSM</td>
<td>state-space model</td>
</tr>
<tr>
<td>SVGP</td>
<td>stochastic variational GP</td>
</tr>
<tr>
<td>TDOA</td>
<td>time difference of arrival</td>
</tr>
<tr>
<td>UKF</td>
<td>unscented Kalman filter</td>
</tr>
<tr>
<td>UKS</td>
<td>unscented Kalman smoother</td>
</tr>
<tr>
<td>VFF</td>
<td>variational Fourier features</td>
</tr>
<tr>
<td>VGP</td>
<td>variational GP</td>
</tr>
</tbody>
</table>
Part I

Background
1

Introduction

This thesis is primarily concerned with obtaining accurate estimates of a system’s state while keeping the computational requirements of the employed algorithms low. The thesis is structured as a compilation, where the introductory part, Chapters 1–5, contains background information seeking to provide intuition into the mathematical techniques used in the second part, which consists of published papers by the author of this thesis.

This chapter is structured as follows: Section 1.1 provides some perspectives on the topics that are dealt with in Part II in order to motivate the particular topics under study. It further provides some background for the specific algorithm choices in the publications. Section 1.2 explicitly specifies the problems that this thesis aims to address. Lastly, Section 1.3 describes the published papers contained in the thesis and in particular, explains the author’s contribution to each of the papers.

1.1 Background and Motivation

The study of dynamical systems dates back to the early days of studies on planetary motion, which laid the foundation for how we understand the mechanics of the universe. Since then, dynamical systems have been studied in, for instance, mechanics [1], fluid dynamics [2], vehicles [3], neuroscience [4], and many more. Simply put, a dynamical system is an entity that changes over time in some manner, e.g., planet Earth revolves around the sun as well as its own axis, without which we would not have day and night cycles, nor would we have seasons.

The field of automatic control, in which this thesis can be categorized, is primarily concerned with controlling systems, i.e., making systems behave in a prescribed manner, e.g., as simple as controlling the temperature of an oven so that the bread you are baking does not burn. For automatic control, dynamical sys-
Introduction

(a) Target tracking (and one case of sensor management) – a sensor (black circle) provides some measurements of targets (colored circles) and the system estimates the historical trajectories (translucent lines) of each target. The translucent yellow area illustrates the field of view of the sensor.

(b) A simplified view of SLAM – the platform estimates its position (orange circle) as well as the landmarks (black squares) by, e.g., measuring the relative distance to each landmark (dashed black lines and translucent blue lines). The landmark estimates are illustrated with translucent blue circles to illustrate the uncertainty.

Figure 1.1: Two problems where accurate state estimates are crucial. Improving the dynamical models as well as the estimation algorithms directly influences the achievable performance. Further, both of these examples critically require a reasonably accurate dynamical model ahead of seeing data. Finally, algorithms for these problems necessarily need to be computationally efficient, as typical applications require them to run on embedded hardware.

Systems are a central concept, without which control would not be possible, nor would it be required (if nothing changes, what would we be controlling?). To control a system, automatic control engineers construct a mathematical entity known as a controller, which calculates an input to the system that is designed to bring the system state to some desired reference; for the oven example, the state could be the temperature and the reference would thus be the desired temperature. Typically, control engineers prefer so-called feedback controllers, as these are generally more robust toward noise or external influence [5]. The basic idea of feedback controllers is that we apply some input, measure the outcome, and then adapt our input according to the discrepancy between the measured outcome and the reference.

In many cases, the states of the system that we would like to control are not directly measurable, or they may be impractical to measure. For instance, the orientation of your smartphone is not directly measurable, but it sure seems to know when you rotate it to landscape mode (for the most part...). In order to be able to control systems when we do not have access to direct measurements of all the states, control engineers require so-called state estimation. Note that we will here also characterize, for instance, target tracking as a control application, even though this application does not necessarily imply that any system is to be “controlled”. The reason for this characterization is simply because these topics are taught at the division of Automatic Control at Linköping University. Further, the methods and techniques that are used within these topics also greatly overlap with more classical control applications. In contrast to classical control applications, it is the state of the system that is the primary interest in, e.g., target track-
ing, which naturally requires state estimation. State estimation is simply the process of estimating the state(s) of interest of the system, based on measurements of some quantities related to the system; the orientation of your smartphone is partly estimated based on measurements of the acceleration of the phone caused by the gravitational forces of the Earth. Two examples of problems that control engineers may face that require state estimation are illustrated in Figure 1.1.

State estimation is ubiquitous not only for automatic control but for virtually every area of science and engineering. It is a necessary component for target tracking [6, 7], motion planning [8, 9], simultaneous localization and mapping (SLAM) [10], sensor management [11], model-based control such as model predictive control [12], inertial navigation [13], and more. Regardless of the application, state estimation provides us with a way of extracting additional information from various measurements, e.g., we can succinctly express that velocity is the time derivative of the position, enabling us to estimate both the position as well as the velocity from only measurements of the position. The basic enabler of this information extraction is a mathematical model of the system in question. In the context of this thesis, the mathematical models we use for state estimation are central, as more precise models will typically allow us to obtain better estimates of the system states. However, as the models become more and more complex, i.e., “more” nonlinear, the state estimation algorithms we employ generally need to be more advanced to deal with these nonlinearities. In this thesis, we consider both the problem of learning more precise mathematical models of dynamical systems online as well as the development of more advanced state estimation algorithms.

1.1.1 Data-Driven Modeling of Dynamical Systems

Mathematical models of dynamical systems have historically been based on domain knowledge, such as simple Newtonian mechanics, allowing the engineer to express various physical relationships based on insights into the system. The bicycle (or single track) model for a car-like vehicle is one such example [14]. Models based on domain knowledge are sometimes referred to as white-box, or if they depend on some unknown parameters as grey-box [15]. White-box and grey-box models both enjoy physical interpretability as well as the ability to incorporate prior domain knowledge. For instance, the unknown parameters in a grey-box model could relate to hydrodynamic forces [16], or dynamic stiffness parameters in robotic manipulators [17]. The most “flexible” model is the black-box model, which typically is neither physically interpretable nor does it incorporate prior knowledge. Instead, it relies on very general mathematical structures with unknown parameters that are identified through, e.g., system identification [15, 18]. The advantage of data-driven models is that they can more easily capture hard-to-model phenomena such as driver interactions [19]. Further, as we show in Papers A and B, incorporating some data-driven components into otherwise physically derived models can yield more accurate short- and long-term predictions. A highly simplified view of three types of models of importance to this thesis is illustrated in Figure 1.2.

Data-driven modeling has achieved remarkable success, e.g., as a component
Before seeing data  

After seeing data

![Simplified view of three types of models. Standard white-box models can capture the main properties of a system's behavior without requiring data from the system. In contrast, purely data-driven models are usually unable to provide any reasonable predictions without access to system data. The "augmented" models that we study in this thesis enjoy the best of both worlds, capturing the main behavior of the system a priori, but can adapt to data online.](image)

in reinforcement learning, it has contributed to beating the (human) world champion(s) at Go [20] and Dota 2 [21]. Most of the modern data-driven models are still based on neural networks, which typically have limited to no possibility of incorporating prior domain knowledge, nor can they typically report the uncertainty in their predictions. Incorporation of prior domain knowledge is of high interest, as it can turn an otherwise hard problem into an easier one and can immediately provide a “minimal performance”; there are also centuries’ worth of expert domain knowledge that will otherwise go unnecessarily lost. Furthermore, accurately quantifying the uncertainty in the model predictions is imperative when the systems are to be incorporated into the “real world” through, e.g., decision support systems. If the uncertainty is not accurately quantified, it cannot be taken into account for subsequent decisions, which is especially important in safety-critical applications, such as autonomous driving [22]. Research efforts have been put into both of these issues to make data-driven models more easily applicable to a wider range of problems. For instance, neural ordinary differential equations (ODEs) [23, 24] embed neural networks within an ODE, which automatically respects the differential constraints of most physical systems. More extensive incorporation of prior domain knowledge is also possible through the use of, e.g., Gaussian processes (GPs) within the ODE as a replacement of the neural network [25], or through recent work on function space representations of neural networks [26]. Uncertainty quantification in neural networks primarily revolves around the so-called Bayesian neural networks (BNNs) [27, 28], which attempt to find a full posterior distribution over the weights of the network itself. This enables BNNs to produce a predictive distribution, rather than a point estimate, which inherently captures the predictive uncertainty.

The de facto gold standard of uncertainty modeling is the GP [29]. GPs provide a principled way of reasoning about uncertainty and they further enjoy closed-form solutions for the standard regression case [29]. Due to these advanta-
1.1 Background and Motivation

gorous properties, GPs have for instance been used for Bayesian optimization [30–32], reinforcement learning [33–35], data-driven control [36, 37], motion planning [38], system identification [39–41], experiment design [42], target tracking [43, 44], SLAM [45, 46], etc. The primary drawback of standard GPs is that the computational complexity scales with the number of data points included in the model. Much effort has gone into reducing this complexity, some of which is detailed in Section 2.2, which has made it possible to apply GPs to increasingly large problems. Still, incorporating GPs into control applications in a computationally efficient manner remains a largely unsolved problem, especially as the problems are scaled up beyond toy examples. Resolving this problem may lead to the wide adoption of partly data-driven models, as these are physically interpretable, build upon existing prior domain knowledge, and may yield more accurate long-term predictions with principled uncertainty quantification. Note that, for many control applications, such as target tracking or model-based control, the sensor model is typically well-defined by the problem geometry and/or physics, whereas the dynamical model can be seen as an assumption about the dynamics of the system in question. In this thesis, the main concern thus lies with data-driven modeling of the dynamical model, even though we touch upon sensor model learning as well. Thus, we will typically assume simple linear sensor models and focus our attention on learning (parts of) the system dynamics online.

1.1.2 Nonlinear State Estimation

Once a model has been constructed, the second component of accurate state estimation is the algorithms that are used to estimate the system’s state. The typical process of (discrete-time) state estimation is schematically depicted in Figure 1.3. The arguably most well-known algorithm for state estimation is the Kalman filter (KF) [47]. The reason for this popularity is its computational efficiency and that it is the optimal estimator in the mean square error (MSE) sense for affine models with additive Gaussian noise. However, most systems are only accurately described using a nonlinear model, where the KF is no longer applicable. More-
over, most data-driven models are nonlinear, necessitating other types of state estimation algorithms. The de facto standard algorithm for nonlinear state estimation is the (Schmidt) extended Kalman filter (EKF) \[48\], which simply linearizes the model locally in each time-step around the current state estimate. While this (more or less) retains the computational efficiency of the KF and typically works well, state estimate divergence has been widely observed for complex models, which may be particularly problematic in safety-critical applications \[49\]. This observation has spurred the development of a plethora of different approximate nonlinear filtering algorithms to tackle this, such as the unscented Kalman filter (UKF) \[50\], the cubature Kalman filter (CKF) \[51\], the iterated posterior linearization filter (IPLF) \[52\], and various iterated versions of all of these \[53–56\]. These methods, particularly the iterated algorithms, have primarily focused on nonlinearities in the sensor model, i.e., the measurement update in Figure 1.3, and still (mostly) retain the computational efficiency of the KF. However, as alluded to in Section 1.1.1, we primarily assume linear sensor models in this thesis and particularly focus on learning complex dynamical models. Hence, the main nonlinearities in the problems we study are contained in the dynamical model, which previous algorithms typically do not deal with, even though some exceptions exist, see e.g. \[57\]. Therefore, it is of interest to develop new algorithms that inherently deal with nonlinearities in the dynamical evolution of the system, such that filter divergence can be avoided while retaining the computational efficiency of previously proposed methods. While we are primarily focused on linearization-based estimators due to their computational efficiency, we briefly detail other approaches to nonlinear state estimation in Section 3.2 as well.

### 1.2 Problem Statement

This thesis is concerned with obtaining accurate estimates of a system’s state, partly through online data-driven adaptation of the dynamical model of the underlying system and partly through developing new state estimators. As the intended use of the models and algorithms is control applications, such as sensor management or model predictive control, the models and algorithms must remain computationally efficient, even for large-scale domains. To that end, this thesis focuses on:

(i) The combination of domain knowledge-based dynamical models with GPs in a computationally efficient manner.

(ii) Development of computationally efficient nonlinear state estimation algorithms for models where the main nonlinearities are contained in the dynamics.

(iii) Reduction of the computational complexity of GPs in low-dimensional applications, such as target tracking or geostatistics.
1.3 Thesis Outline

The thesis is split into two parts. Part I aims to give an intuitive understanding of the basic methodology used throughout the author’s publications. The background is by no means extensive and instead provides references to other publications where the technical material can be found. Particularly, the background aims to give an intuitive understanding rather than a technical one. The publications included in the thesis are contained in Part II. A summary of each publication and the author’s particular contributions to each work is listed in Section 1.3.1. Finally, a complete list of works that the author has contributed to during his time as a PhD student is provided at the end of this section.

1.3.1 Publications and Contributions

The included publications are listed here with a short summary of each paper. The contributions of this thesis’ author are highlighted in connection to each publication. The following abbreviations are used for the authors of each paper: Anton Kullberg (AK), Arno Solin (AS), Frederiek Wesel (FW), Frida Viset (FV), Gustaf Hendeby (GH), Isaac Skog (IS), and Martin Skoglund (MS).


Summary: The class of dynamical models studied in the thesis is introduced. This class is both physically interpretable and allows for data-driven learning of hard-to-model phenomena, through a GP component which is restricted to only a subspace of the entire state-space. The first large-scale example illustrating a potential application of the proposed model class is presented. In this example, the model class and associated inference scheme are shown to reduce the overall estimation error as compared to using only a physically derived state-space model, indicating the usefulness of the new model.

Background: GH and IS had the original idea for the paper. AK developed the methodology, implemented and ran the numerical experiments, and wrote the paper. GH and IS provided technical advice and editorial comments on the paper.


Summary: The computational aspects of the model and inference scheme in Paper A are studied. An approximate inference scheme that enables online learning in large-scale models, which is essentially computationally invariant to the number of model parameters, is developed. By using compactly supported radial basis functions and an approximate Kalman gain, the computational complexity is considerably reduced and is essentially determined by the support of the basis functions. The inference scheme and associated model are shown to have competitive estimation performance as compared to the method from Paper A while being real-time applicable even if the number of model parameters grows large.

Background: AK had the original idea for the paper, developed the methodology, implemented and ran the numerical experiments, and wrote the paper. GH and IS provided technical advice and editorial comments on the paper.


Anton Kullberg, Martin A. Skoglund, Isaac Skog, and Gustaf Hendeby.

Summary: A relationship between nonlinear filtering algorithms based on iterated statistical linearization and quasi–Newton (QN) algorithms is discovered. Two such relationships are described in detail which allows for a QN interpretation of the UKF, CKF, IPLF, and similar algorithms. The implications of the discovery are new insights into the behavior of iterated statistical linearization algorithms and possible avenues for developing more efficient damping schemes.

Background: AK and MS had the original idea for the paper. AK derived the original theoretical results with input from MS. IS and GH noted weaknesses in the first draft of the theoretical derivations whereupon AK re-derived the main results. AK wrote the paper and MS, IS, and GH provided editorial comments.

Paper D: Dynamically Iterated Filters: A unified framework for improved iterated filtering via smoothing.

Anton Kullberg, Martin A. Skoglund, Isaac Skog, and Gustaf Hendeby.

Summary: A complete framework for dynamically iterated filters is proposed. The framework is particularly developed for efficient inference in models with nonlinear dynamics. The framework constitutes an easy way of deriving multiple new filters, depending on the chosen linearization strategy, which relies on
a simple mechanism of “feedback” in linearization-based filtering. Three such algorithms are particularly investigated and are shown to improve performance over their non-iterated counterparts even in “simple” cases where standard non-linear filtering algorithms such as the EKF are widely used. The algorithms are empirically demonstrated to have superior robustness properties with respect to the noise parameter tuning. The paper is an extension of the author’s previous work [62], which was the runner up for best student paper award at FUSION 2023.

**Background:** AK had the original idea for the paper, developed the methodology, implemented the numerical experiments, and wrote the paper. MS provided technical insights through discussions and connections to related work as well as wrote some minor parts. IS and GH provided editorial comments and advice on numerical experiments.

**Paper E: Exploiting Hankel-Toeplitz Structures for Fast Computation of Kernel Precision Matrices.**


**Summary:** The precision matrices of several widely used basis function expansion models are shown to have some type of Hankel/Toeplitz, block-Hankel, or block-Toeplitz structure. Two theorems are provided that make it particularly easy to check whether the user’s favorite choice of basis function exhibits this structure. The structure enables (approximate) GP inference at an \(O(NM)\) cost instead of \(O(NM^2)\), significantly improving the practical usability of, e.g., the Hilbert-space Gaussian process (HGP) for larger problems.

**Background:** FV first discovered the hidden structures in the HGP. FV, AK, and FW developed an initial proof for the HGP. FV and FW further discovered similar structures in other basis function expansion models. The formalized proof in the paper was developed in collaboration between all co-authors. AK and FV independently implemented the method. AK ran two out of three numerical experiments and FV ran the third. AK and FV wrote the main parts of the paper with help from FW and AS. FW and AS further provided connections to other work, technical advice, and advice for the experimental section. AK and FV contributed equally to the paper.

**Paper F: Adaptive Basis Function Selection for Computationally Efficient Predictions.**


**Summary:** An a posteriori method for identifying the most important basis functions in a basis function expansion is developed. The method allows for
prediction-point-dependent basis function selection at a low cost, enabling fast predictions in large models with little loss in accuracy. By exploiting the structure of the basis function expansion, the selection method does not require any additional data, otherwise common in similar methods in, e.g., the neural network literature. Further, the method may also be used for fast approximate model compression which could be beneficial from a distributed estimation point of view.

**Background:** The idea was born through discussions between AK and FV. AK developed the methodology, implemented the numerical experiments, and wrote the paper. FV provided technical discussions, helped clarify the core contribution, and provided insightful connections to other work. IS and GH provided editorial comments.

### 1.3.2 Complete List of Publications

For completeness, a full list of publications that the author has contributed to during his time as a PhD student is given below in chronological order. Publications included in Part II are marked with ⋆.


1.3.3 Source Code Availability

To aid in reproducing the results of the papers, the source code for most of the papers has been released publicly on the author’s GitHub, https://github.com/AOKullberg. Source code for Papers A and B can be made available upon request. The source code for this thesis is also released in its entirety (excluding papers) on https://github.com/AOKullberg.
This chapter discusses two fundamental models that are commonly used to represent general functions. Before presenting these two models, we shall discuss function approximation, which is what these two models are technically used for.

Here, we assume that a function $f^0 : \mathbb{R}^n \rightarrow \mathbb{R}$ exists, that takes $x \in \mathbb{R}^n$ as input and produces a real-valued response. To learn a representation of the function $f^0$, we have access to noisy observations of function values at particular inputs, i.e.,

$$y_i = f^0(x_i) + e_i,$$

(2.1)

where $y_i \in \mathbb{R}$ is an observation, and $e_i$ is a random noise affecting the observations. We will assume that we have access to $N$ such observations, which we collect in a vector $y_{1:N} = [y_1^\top \ y_2^\top \ \cdots \ y_N^\top]^\top$. The corresponding inputs are collected in the matrix $X = [x_1 \ x_2 \ \cdots \ x_N]^\top$. Our objective is now to find a model $f$ of the function $f^0$, that resembles the true function as closely as possible. The closeness can for instance be defined by the MSE

$$\frac{1}{N} \sum_{i=1}^{N} (f^0(x_i) - f(x_i))^2,$$

(2.2)

over the inputs in $X$. Once the representation $f$ has been found, it can then readily be used to predict function values at new inputs $x^*$, which is the main interest here.

In this chapter, we will primarily focus on Bayesian representations, as this fits well into the state estimation framework presented in Chapter 3. Practically, this means that we will place probabilistic priors $p(\theta)$ on the parameters $\theta$ defining the function $f$. This can be interpreted as implicitly placing a prior on the
function itself, loosely “p(f)”.

With this “functional” prior, we inform our learning process about what types of functions we expect to see from data, which acts as a form of regularization, implicitly biasing the function approximation toward particular representations. We shall also consider the GP, which provides a convenient way of constructing p(f) immediately, without resorting to parametric priors.

The chapter is organized as follows. Section 2.1 covers the basis function (BF) expansion model, which is a flexible model commonly used in many different fields, primarily in the signal processing community. Section 2.2 introduces the GP, the gold standard of uncertainty quantifying models, which is used in a wide variety of disciplines, primarily in the machine learning community.

2.1 Basis Function Expansions

The BF expansion model can be viewed as a generalization of a standard linear model and is given by

\[ f(x) = \sum_{j=1}^{M} \phi^j(x)\theta^j = \Phi(x)\theta, \quad (2.3) \]

where \( \phi^j : \mathbb{R}^n \rightarrow \mathbb{R} \) is the \( j \)th so-called basis function and \( \theta^j \) is the corresponding weight. Further, \( \Phi(x) = [\phi^1(x) \ \phi^2(x) \ \cdots \ \phi^M(x)] \) and \( \theta = [\theta^1 \ \theta^2 \ \cdots \ \theta^M]^\top \).

Note that the standard linear model is obtained by selecting \( \Phi(x) = x^\top \). Example 2.1 provides a visual interpretation of this model for a particular choice of BFs, the radial basis function (RBF).

---

Example 2.1: RBF Expansion Representation

Assume that the true function \( f^0 \) is given by

\[ f^0(x) = x + \sin\left(\frac{3\pi}{5}x\right). \]

The function is then modeled with a BF expansion (2.3) with BFs given by

\[ \phi^j(x) = \exp\left(-\frac{1}{2l^2}(x - c^j)^2\right), \]

where \( c^j \in \{-2, -1, 0, 1, 2\} \) and \( l = 0.5 \). The weights of the expansion are found through least-squares on 100 noiseless observations in \([-2, 2] \), meaning that we are only interested in capturing the function on this domain.
Given the representation (2.1), the question of how to find the weights $\theta$ remains. One standard way of obtaining them is through least-squares regression, which minimizes the MSE (2.2). To fit into the state estimation framework presented in Chapter 3, we will here assume a prior on the parameters $p(\theta)$, which will allow us to find the posterior distribution $p(\theta|y_{1:N})$, which inherently captures the uncertainty of the parameters. To that end, assume that the observation model is given by (2.1) and that the noise $e_i$ is i.i.d. Gaussian, i.e., $e_i \sim N(e_i; 0, \sigma^2)$. The data likelihood is thus given by

$$p(y_{1:N}|\theta, X) = \prod_{i=1}^{N} p(y_i|\theta) = N(y; \Phi(x)\theta, \sigma^2 I).$$

Henceforth, we will drop the dependence on $X$, unless it is explicitly relevant to the discussion, but still assume $X$ is given. We shall assume that the parameter prior is given by

$$p(\theta) = N(\theta; 0, \Lambda),$$

which corresponds to a prior on the function values $f$ at inputs $X$ given by

$$p(f) = \int_{\mathbb{R}} p(f|\theta)p(\theta) \, d\theta = N(f; 0, \Phi\Lambda\Phi^\top).$$

The parameter posterior is then given by Bayes’ rule as

$$p(\theta|y_{1:N}) = \frac{p(y_{1:N}|\theta)p(\theta)}{p(y_{1:N})} = N(\theta; m, S)$$

$$m = \frac{1}{\sigma^2} S\Phi^\top y_{1:N}$$

$$S^{-1} = \frac{1}{\sigma^2} \Phi^\top \Phi + \Lambda^{-1}. $$
Note that the dependence on $x$ has been dropped for notational convenience and, with abuse of notation, $\Phi = \begin{bmatrix} \Phi(x_1)^\top & \Phi(x_2)^\top & \ldots & \Phi(x_N)^\top \end{bmatrix}^\top$. We can now also define the posterior distribution of the function values at particular test locations $x^\ast$, which is given by

$$p(f^\ast | y_{1:N}, x^\ast) = \mathcal{N}(f^\ast; \Phi(x^\ast)m, \Phi(x^\ast)S\Phi^\top(x^\ast)).$$

(2.7)

One way of understanding this “functional” posterior distribution, is that it inherently captures the uncertainty that remains about the true function, even after seeing the data. The hope is that the uncertainty accurately captures the general ‘region’ in which the true function resides. This uncertainty is also implicitly biased by the prior that we have placed on the parameters. Example 2.2 considers the same underlying function as in Example 2.1 but now finds the posterior distribution of the function given some noisy observations.

--- Example 2.2: Posterior Distribution of RBF Expansion ---

We reuse the model from Example 2.1 and now assume a prior on the parameters given by

$$p(\theta) = \mathcal{N}(\theta; 0, I).$$

We then generate noisy observations of $f^0$ by drawing $N = 5$ uniform samples from the input space and corrupting the corresponding function values with noise given by $e_i \sim \mathcal{N}(e_i; 0, 0.01)$. The parameter posterior is found through (2.6) and we visualize the posterior distribution (2.7) against the true function $f^0$ in Figure 2.2.

![Figure 2.2: The posterior distribution (2.7) of the RBF expansion from Example 2.1 when trained with 5 noisy observations. The shaded blue area represents the uncertainty of the posterior at three standard deviations. In principle, the uncertainty reflects what we still do not know about the function.](image)

Choosing a reasonable prior on the weights $p(\theta)$ of the BF expansion can be tricky. One way of remedying this is by making the prior diffuse and uninformative, but that may defeat the purpose of the prior to begin with. Another way is by considering the implicit prior on the function $p(f)$ that the weight prior
corresponds to. This makes it easier to understand what types of functions that the weight prior corresponds to. However, this begs another question: instead of choosing a weight prior \( p(\theta) \) and a set of BFs \( \Phi \) and seeing what functional prior \( p(f) \) this corresponds to, can we not just place a prior on the function immediately?

### 2.2 Gaussian Processes

GPs can be viewed as a general framework for placing priors on functions \( p(f) \). Formally, we will denote a GP as [29]

\[
f(x) \sim \mathcal{GP}(\mu(x), \kappa(x, x')),
\]

where \( \mu : \mathbb{R}^n \rightarrow \mathbb{R} \) is the mean function and \( \kappa : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R} \) is the kernel, or covariance function. The mean function \( \mu(\cdot) = \mathbb{E}[f(\cdot)] \) we will take to be \( 0 \), without loss of generality as we can, e.g., zero-mean center the data [29]. The covariance function, or kernel, \( \kappa(\cdot, \cdot) = \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))] \) describes the covariance between any pair of inputs \( x \) and \( x' \). The defining property of the GP is that any subset of variables produced by the GP are jointly Gaussian distributed [29]. Formally, this means that for a given set of inputs \( X \), the corresponding function outputs are distributed as

\[
p(f|X) = \mathcal{N}(f; 0, K_f).
\]

Here, \( f = \begin{bmatrix} f(x_1) & f(x_2) & \cdots & f(x_N) \end{bmatrix}^\top \) and \( [K_f]_{ij} = \kappa(x_i, x_j) \).

The construction (2.8) provides us with a convenient way of expressing our prior assumptions on the function \( f(\cdot) \) directly through the mean function \( \mu(\cdot) \) and the kernel \( \kappa(\cdot, \cdot) \). Typically, for regression, the observations are assumed to be generated by (2.1), i.e., \( p(y_{1:N}|f, X) = \mathcal{N}(y_{1:N}; f, \sigma^2 I) \). If we assume that \( \mu(\cdot) = 0 \) and marginalize out the function values \( f \), we find that the (marginal) likelihood is given by [29]

\[
p(y_{1:N}|X) = \int_{\mathbb{R}} p(y_{1:N}|f, X)p(f|X) \, df = \mathcal{N}(y_{1:N}; 0, K_f + \sigma^2 I).
\]

With this in mind and in light of (2.9), we can construct the joint distribution of the observations \( y_{1:N} \) and the function values \( f^* \) at the test inputs \( X^* \) as

\[
p(f^*, y_{1:N}|X^*, X) = \mathcal{N}\left( \begin{bmatrix} f^* \\ y_{1:N} \end{bmatrix}; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{ss} & K_{sf} \\ K_{fs} & K_f + \sigma^2 I \end{bmatrix} \right).
\]

Here, e.g., \( [K_{sf}]_{ij} = \kappa(\{X\}_i, \{X^*\}_j) \). The corresponding test posterior can then be readily found through standard Gaussian conditioning as [29]

\[
p(f^*|y_{1:N}, X, X^*) = \mathcal{N}\left( f^*; \mathbb{E}[f^*], \text{cov}(f^*) \right)
\]

\[
\mathbb{E}[f^*] = K_{sf} (K_f + \sigma^2 I)^{-1} y_{1:N}
\]

\[
\text{cov}(f^*) = K_{ss} - K_{sf} (K_f + \sigma^2 I)^{-1} K_{fs}.
\]
Example 2.3: Test Posterior of GP

We reuse the same function \( f^0 \) as in Example 2.1 and the observation model as in Example 2.2. We place a GP prior on the function \( f \), with a squared exponential (SE) kernel (2.12) with parameters \( l = 0.5 \) and \( \sigma_f^2 = 1 \). This prior is visualized in Figure 2.3, where three samples of the prior are also shown in blue.

![Figure 2.3: GP prior on the function \( f \). The shaded area represents the uncertainty associated with the prior at three standard deviations. Three function samples are shown in blue and represent some possible functions that correspond to this prior.](image)

We then condition our GP on the same observations as in Example 2.2 through (2.11) and visualize the test posterior of the GP in Figure 2.4, where three samples of the posterior are also shown in blue.

![Figure 2.4: GP posterior on the function \( f \), conditioned on the observations \( y_{1:5} \). The uncertainty has now drastically decreased around the observations but the uncertainty about the function is still high elsewhere. Three function samples are shown in blue and represent some functions that are possible under the posterior. Notice that all of the possible functions now have to pass close to the observations.](image)
Notice that the posterior only depends on the kernel evaluated at the training inputs $X$ and the test inputs $X^*$ and that there are no parameters in the model, i.e., it is a so-called non-parametric model. Thus, as soon as a kernel $\kappa(\cdot, \cdot)$ has been specified, the test posterior is immediately available, without having to find any parameters as an intermediate step. Further, we can view (2.11) as a smoothing method that interpolates the available data to produce predictions at new input locations. Henceforth, we will typically leave out the explicit conditioning on $X$ and $X^*$ for notational convenience.

**Kernel Choice:** The kernel can generally be chosen freely, under the restriction that it needs to be positive semi-definite, which in practice means that the resulting covariance matrix $K_{ff}$ is positive semi-definite. The de facto standard choice is the SE kernel, given by

$$
\kappa(x, x') = \sigma_f^2 \exp \left( -\frac{1}{2l^2} \|x - x'\|^2 \right),
$$

(2.12)

where $l$ is called the characteristic lengthscale and $\sigma_f^2$ is the signal variance. The SE kernel generally promotes very smooth functions that interpolate well but have poor extrapolation properties far from the training data. Example 2.3 provides a visual interpretation of the GP model and the test posterior that it produces, given a specific choice of kernel.

**Hyperparameter Optimization**

While the GP is typically described as a non-parametric method, the kernel $\kappa(\cdot, \cdot)$ typically contains a number of hyperparameters. These hyperparameters affect the types of functions that the GP can represent. For instance, the SE kernel (2.12) is parametrized by a lengthscale $l$ and a signal variance $\sigma_f^2$. From a frequency perspective, the lengthscale is inversely proportional to the “width” of the spectral density of the function, where a smaller lengthscale can capture higher frequencies. Conversely, a larger lengthscale biases the GP toward slowly varying functions. On the other hand, the signal variance directly changes the amplitude of the spectral density, meaning that larger fluctuations are expected.

The hyperparameters can either be chosen manually, which is common in online settings, such as in Paper B [59] or for magnetic field SLAM [46]. Otherwise, the most common alternative is through marginal log-likelihood (MLL) optimization [29]. For the GP (2.9) with the additive observation model (2.1), the MLL is given by the logarithm of (2.10), which is explicitly given by

$$
\log p(y_{1:N}) = -\frac{1}{2} \left( N \log(2\pi) + \log \det (K_{ff} + \sigma^2 I) + y^\top \left( K_{ff} + \sigma^2 I \right)^{-1} y \right). 
$$

(2.13)

Typically, maximization of (2.13) is performed over all of the kernel hyperparameters as well as the observation noise standard deviation $\sigma$. Note that the hyperparameters of the kernel are typically few, meaning that this is a low-dimensional optimization problem and that optimizing (2.13) generally works well for finding reasonable parameters. Still, the objective is non-convex with a lot of local
Function Approximation

minima and care needs to be taken to initialize the optimization reasonably well. Lastly, note that evaluating the objective generally costs $O(N^3)$, due to the need to compute the inverse of $(K_{ff} + \sigma^2 I)$ and the log-determinant.

### 2.2.1 Computational Aspects

As we have established in Section 2.2, the GP is a non-parametric model and its expressivity and flexibility are completely determined by the choice of kernel and the data itself. However, this flexibility comes with a cost in terms of computational complexity. Notice that in the test posterior (2.11), we need to compute the inverse of $K_{ff} + \sigma^2 I$, with a computational cost of $O(N^3)$, since $K_{ff} \in \mathbb{R}^{N \times N}$ [29]. Hence, the computational complexity grows with the number of measurements $N$ that the GP is conditioned on. Remedies to this complexity have been widely studied, but in the context of this thesis, only two such approaches will be discussed.

#### Inducing Points

Conceptually, the inducing point approach can be thought of as parametrizing the otherwise non-parametric GP [71]. This is achieved by defining a set of auxiliary variables $u = [u_1, u_2, \ldots, u_M]$ that are commonly referred to as inducing points [71–74]. In the context of this thesis, the inducing outputs $u$ represent function values at particular input locations $z$, the inducing inputs, i.e., $u^i = f(z^i)$. The inducing outputs $u$ are assumed to be generated from the GP, i.e., they are jointly Gaussian distributed with $f$ and $f^\star$. We can thus define the joint distribution between $f$, $f^\star$, and $u$ as

$$p(f, f^\star, u) = \mathcal{N}\left(\begin{bmatrix} f \\ f^\star \\ u \end{bmatrix}; \begin{bmatrix} 0 \\ K_{ff} & K_{f^\star} & K_{fu} \\ 0 & K_{f^\star} & K_{u^\star} \\ 0 & K_{fu} & K_{uu} \end{bmatrix}\right).$$

(2.14)

Notice that it is possible to recover the exact joint $p(f, f^\star)$ by marginalization

$$p(f, f^\star) = \int_{\mathbb{R}} p(f, f^\star, u) \, du = \int_{\mathbb{R}} p(f, f^\star|u) p(u) \, du.$$  

(2.15)

Nothing has yet been gained, as $u$ are only additional prior function values that do not change the posterior (2.11). The crucial step in achieving computational benefits is by assuming that the joint conditional prior $p(f, f^\star|u)$ can be approximated as

$$p(f, f^\star|u) \approx q(f, f^\star|u) = q(f|u) q(f^\star|u).$$

(2.16)

The assumption (2.16) crucially implies that the test function values $f^\star$ are assumed conditionally independent of the training function values $f$, conditioned on $u$. The inducing points $u$ can now be interpreted as a parametrization of the otherwise non-parametric GP. The problem has now turned into that of finding
the posterior of the inducing points \( p(\mathbf{u}|y_{1:N}) \) which can subsequently be used to compute the test posterior, now given by

\[
q(f^*|y_{1:N}) = \int_{\mathbb{R}} q(f^*|\mathbf{u})p(\mathbf{u}|y_{1:N})\,d\mathbf{u}.
\] (2.17)

---

**Example 2.4: Inducing Points in a GP**

We consider the same problem as Example 2.3 but with inducing inputs placed at locations \( z_j \in \{-2, -1, 0, 1, 2\} \). The posterior over the inducing points is found through (2.19) and the test posterior through (2.18), visualized in Figure 2.5. The information flows from the observations \( y_{1:5} \) to the inducing points \( \mathbf{u} \) that act as parameters of the model. Note that both the mean \( m \) and covariance \( S \) of the inducing points \( \mathbf{u} \) are captured, signified by the error bars in the top figure. After finding the posterior over the inducing points, the observations can be discarded and the test posterior can be constructed solely from the posterior over the inducing points \( p(\mathbf{u}|y_{1:5}) = \mathcal{N}(\mathbf{u}; m, S) \).

---

*Figure 2.5: The top figure illustrates the learning process of the inducing point approach, where information flows from the observations \( y_{1:5} \) to the inducing points \( \mathbf{u} \). The bottom figure is the test posterior that can be computed from the posterior over the inducing points alone, without access to \( y_{1:5} \). Three posterior samples are visualized.*
This still begs the question of how this affects the computational complexity. In turn, this can be decomposed into two subproblems, the computational complexity of computing the inducing point posterior \( p(u|y_{1:N}) \), and the complexity of subsequently computing the test posterior (2.17). Before diving into those details, Example 2.4 provides a visual interpretation of what inducing points mean in the context of this thesis.

**Computational Complexity**: In order to understand the computational complexity of computing the test posterior, notice that the conditional \( q(f^*|u) \) is readily available from (2.14) and is given by

\[
q(f^*|u) = \mathcal{N}(f^*; K_{uu}K_{uu}^{-1}u, K_{*u}K_{*u}^{-1}K_{uu}).
\]

Assuming that a posterior over the inducing inputs \( p(u|y_{1:N}) = \mathcal{N}(u; m, S) \) is available, the test posterior is then simply given by

\[
q(f^*|y_{1:N}) = \mathcal{N}(f^*; E[f^*], \text{cov}(f^*))
\]

From here, we can conclude that once the posterior \( p(u|y_{1:N}) \) is available, computing the predictions \( f^* \) costs \( O(M^3) \), which is due to the matrix multiplications for computing the variance.

To find the posterior over the inducing inputs \( p(u|y_{1:N}) \), we follow the same reasoning as for the full GP, i.e., we consider the joint

\[
p(u, y_{1:N}) = \mathcal{N}\left( \begin{bmatrix} u \\ y_{1:N} \end{bmatrix}; \begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} K_{uu} & K_{uf} \\ K_{fu} & K_{ff} + \sigma^2 I \end{bmatrix} \right). \]

The posterior is then readily given by

\[
p(u|y_{1:N}) = \mathcal{N}\left( u; K_{uf}(K_{ff} + \sigma^2 I)^{-1}y_{1:N}, K_{uu} - K_{uf}(K_{ff} + \sigma^2 I)^{-1}K_{fu} \right). \]

Evidently, the inverse \((K_{ff} + \sigma^2 I)^{-1}\) is still present and the posterior thus costs \( O(N^3) \) to compute. This can be remedied in a few ways, but the most commonly used is that of the *fully independent conditional* (fic) approximation [71, 72], which crucially assumes that the training function values are independent, given the inducing inputs, i.e.,

\[
q(f|u) = \prod_{i=1}^{N} p(f_i|u) = \mathcal{N}(f; K_{fu}K_{uu}^{-1}u, \text{diag}(K_{ff} - K_{fu}K_{uu}^{-1}K_{uf})).
\]

Under this assumption, the likelihood is given by

\[
p(y_{1:N}) = \int p(y_{1:N}|f)q(f|u)p(u) \, df\,du = \mathcal{N}(y_{1:N}; 0, Q_{ff} + \text{diag}(K_{ff} - Q_{ff} + \sigma^2 I)),
\]
2.2 Gaussian Processes

where \( Q_{ab} = K_{uu}^{-1} K_{ub} \). The joint distribution of \( y_{1:N} \) and \( u \) is now given by

\[
p(u, y_{1:N}) = \mathcal{N} \left( \begin{bmatrix} u \\ y_{1:N} \end{bmatrix}; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{uu} & K_{uf} \\ K_{fu} & Q_{ff} + \Lambda \end{bmatrix} \right),
\]

where \( \Lambda = \text{diag} [K_{ff} - Q_{ff}] + \sigma^2 I \) is a diagonal matrix. Thus, under the FIC assumption, the posterior over \( u \) is given by

\[
p(u | y_{1:N}) = \mathcal{N} \left( \begin{bmatrix} u \end{bmatrix}; m, S \right) \tag{2.19a}
\]

\[
m = K_{uf} (Q_{ff} + \Lambda)^{-1} y_{1:N} \tag{2.19b}
\]

\[
S = K_{uu} - K_{uf} (Q_{ff} + \Lambda)^{-1} K_{fu} \tag{2.19c}
\]

Now, notice that

\[
(Q_{ff} + \Lambda)^{-1} = \Lambda^{-1} - \Lambda^{-1} K_{fu} \left( K_{uf} \Lambda^{-1} K_{fu} + K_{uu} \right)^{-1} K_{uf} \Lambda^{-1}
\]

Since \( \Lambda \) is a diagonal matrix, the dominating cost is associated with the product (\( \ast \)) which costs \( O(NM^2) \) to evaluate. This cost is generally common to most inducing point approximations without further assumptions on the kernel.

Basis Function Representations

In Section 2.1, we saw that the BF expansion could be interpreted as a prior on the function values \( p(f) \), similarly to how the GP defines such a prior immediately. We can also move in the opposite direction by approximating the GP prior on the function values as \([71, 75]\)

\[
p(f) = \mathcal{N}(f; 0, K_{ff}) \approx \mathcal{N}(f; 0, \Phi \Lambda \Phi^\top).
\]

In light of Section 2.1, we can immediately recognize this in terms of an equivalent BF expansion

\[
f(x) = \Phi(x) \theta, \quad p(\theta) = \mathcal{N}(\theta; 0, \Lambda).
\]

The crucial approximation step can thus be viewed as a low-rank approximation of the kernel matrix \( K_{ff} \) as

\[
K_{ff} \approx \Phi \Lambda \Phi^\top, \tag{2.20}
\]

similarly to the inducing point approach \([71]\). The technicalities in selecting \( \Phi \) and \( \Lambda \) are not of particular interest for this thesis and we shall only mention one such choice and roughly how it is derived. With the approximation (2.20), the parameter posterior \( p(\theta | y_{1:N}) \) is now given by (2.6) and the test posterior is similarly given by (2.7).
Hilbert-space GP: Note that any kernel \( \kappa(\cdot,\cdot) \) may be associated with a covariance operator \( \mathcal{K} \) defined as \[ \mathcal{K}g = \int \kappa(\cdot,x')g(x')\,dx'. \]

In [75], it was shown that this operator, under the assumption of an isotropic covariance function \( \kappa(\cdot,\cdot) \), may be expressed in terms of a series as
\[ \mathcal{K} = a_0 + a_1(-\nabla^2) + a_2(-\nabla^2)^2 + \ldots, \] (2.21)
where \( \nabla^2 \) is the Laplace operator. They use this (truncated) representation to derive an approximation of the kernel given by
\[ \kappa(x,x') = \sum_{j=1}^{M} S\left(\sqrt{\lambda^j}\right)\phi^j(x)\phi^j(x'). \] (2.22)

Here, \( S(\cdot) \) is the spectral density of the covariance function, and \( \lambda^j \) and \( \phi^j(\cdot) \) are the eigenvalues and eigenfunctions of the Laplace operator on a domain \( \Omega \) that are found by solving
\[ \begin{cases} -\nabla^2 \phi^j(x) = \lambda^j \phi^j(x), & x \in \Omega \\ \phi^j(x) = 0, & x \in \partial \Omega. \end{cases} \] (2.23)

We will henceforth refer to this approach as the HGP and an example of this approximation can be found in Example 2.5.

Other BF Representations: Note that while the HGP defines one particular BF representation of a GP, there are many such correspondences, some of which are discussed here. Assume a BF expansion given by (2.3) with a parameter prior given by (2.4), i.e.,
\[ f(x) = \sum_{j=1}^{M} \phi^j(x)\theta^j = \Phi(x)\theta, \quad p(\theta) = \mathcal{N}(\theta; 0, \Lambda). \]

Then, this may be viewed as a GP with a kernel given by
\[ \kappa(x,x') = \Phi(x)\Lambda\Phi(x')^\top. \]

If the prior covariance \( \Lambda \) is assumed diagonal, this simplifies to
\[ \kappa(x,x') = \sum_{j=1}^{M} \phi^j(x)[\Lambda]_{jj}\phi^j(x'). \]

Given the choice of \( \phi^j(\cdot) \) as an RBF, with \( \Lambda = \sigma_f^2 I \), the corresponding kernel approaches the standard SE kernel in the limit of an infinite number of BFS on a grid with infinitesimal grid spacing [29].
Example 2.5: 1D HGP on a Rectangular Domain

We consider the same function and GP prior as in Example 2.3. The kernel is approximated by (2.22) where the BFs are found by (2.23) on the domain $\Omega = [-L, L]$ with $L = 4$. The BFs are given by [75]

$$
\phi_j(x) = \frac{1}{\sqrt{L}} \sin \left( \frac{j \pi x + L}{2L} \right),
$$

where the first five are visualized in the top plot of Figure 2.6. The test posterior is given by (2.7) and is visualized in the bottom of Figure 2.6. Similarly to Examples 2.2 and 2.4, the observations $y_{1:5}$ can be discarded after learning.

---

**Figure 2.6**: The BF expansion defined by (2.23) on a rectangular domain $\Omega = [-4, 4]$. The BFs are essentially a Fourier-type basis on a fixed domain. Three posterior samples are also visualized.
Even the inducing point approaches may be viewed in some sense as a BF expansion with a specific choice of BF's [71]. To see this, note that the inducing point test posterior (2.18) for one test point \( \mathbf{x}^* \) can be written as

\[
\mathbb{E}[f^*] = \kappa_{\mathbf{u}^*} \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1} \mathbf{m} = \begin{bmatrix} \kappa_{\mathbf{u}^*} \\ \kappa_{\mathbf{u}^*} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1} \\ 0 \end{bmatrix} \mathbf{m} \tag{2.24a}
\]

\[
\text{cov}(f^*) = \kappa_{\mathbf{u}^*} - \kappa_{\mathbf{u}^*} \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}(\mathbf{K}_{\mathbf{u}\mathbf{u}} - \mathbf{S}) \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1} \kappa_{\mathbf{u}^*}
\]

\[
= \begin{bmatrix} \kappa_{\mathbf{u}^*} & \kappa_{\mathbf{u}^*} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1} & 0 \\ 0 & \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}(\mathbf{K}_{\mathbf{u}\mathbf{u}} - \mathbf{S}) \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1} \end{bmatrix} \begin{bmatrix} \kappa_{\mathbf{u}^*} \\ \kappa_{\mathbf{u}^*} \end{bmatrix}, \tag{2.24b}
\]

where \( \kappa_{\mathbf{u}^*} \) is defined similarly to \( \mathbf{K}_{\mathbf{u}^*} \) but with only one test point. The test posterior (2.24) can now be identified as a BF expansion test posterior (2.7) with the basis

\[
\Phi = \begin{bmatrix} \kappa_{\mathbf{u}^*} \\ \kappa_{\mathbf{u}^*} \end{bmatrix},
\]

and the weight posterior

\[
p(\theta) = \mathcal{N}(\theta; \bar{m}, \bar{S}).
\]

One can note that the basis of the expansion is “adaptive”, in the sense that it places a “basis function” \( \kappa_{\mathbf{u}^*} \) on the test point \( \mathbf{x}^* \). This is essentially exactly the augmentation approach discussed in [71], originally proposed in [77, 78] to “heal” the predictive variance of BF expansion models. To understand what “healing” practically means, the reader may, e.g., compare the predictive variance (shaded blue areas) of Examples 2.2 and 2.4.

**Other Approximate GPs**

The standard inducing point approach, such as the FIC method, approximates the GP model and then performs standard Bayesian inference. A qualitatively different approach is the variational GP (VGP) which leaves the prior GP intact and instead aims to approximate the posterior [73, 79]. In the VGP, the Kullback–Leibler (KL) divergence between an approximate and the true posterior is minimized. Note that this is equivalent to maximizing the evidence lower bound (ELBO), which in turn is a lower bound on the MLL (2.10). In [79], the ELBO is maximized analytically over the inducing point distribution \( q(\mathbf{u}) \) which leads to a test posterior that is identical to that of the projected process approach from [74, 80], which is a standard inducing point approach. The difference instead lies with the optimization objective, which turns out to be more suitable for selecting both kernel hyperparameters as well as the inducing inputs \( \mathbf{z} \) themselves [73]. While [73] in effect marginalized out \( q(\mathbf{u}) \) by analytical maximization, [81] maintains an explicit representation of \( q(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S}) \), where \( \mathbf{m} \) and \( \mathbf{S} \) are treated as additional variational parameters. This leads to an ELBO which can be optimized w.r.t. the inducing inputs \( \mathbf{z} \), the parameters \( (\mathbf{m}, \mathbf{S}) \) of the variational distribution \( q(\mathbf{u}) \) as well as the kernel hyperparameters. In particular, the primary novelty
of [81] is that this approach enables the use of stochastic gradient descent (SGD) for optimization, effectively scaling GPs to significantly larger datasets. This approach, the stochastic variational GP (SVGP), has since gained a lot of attention and is the de facto standard approach for scalable GPs, with implementations in GPFlow [82], GPyTorch [83] and GPJax [84].

Similarly to Section 2.2.1, a simple approach to scalable GPs is by approximating the covariance matrix $K_{ff}$ with a rank $M$ approximation $K_{ff} \approx Q_{ff}$. An optimal rank $M$ matrix $Q_{ff}$ can be obtained by finding the $M$ most significant eigenvalues and eigenvectors of $K_{ff}$. By considering the limiting case of infinite data, where $\frac{1}{N}K_{ff}$ approaches the kernel operator, i.e., $\frac{1}{N}K_{ff} \rightarrow K$, the problem turns into finding the eigenvalues and eigenfunctions of $K$ [85]. Mercer’s theorem can then be used to express the kernel in terms of an infinite series of the eigenvalues and eigenfunctions of $K$ [76]. Truncation of this series to the $M$ most significant eigenvalues then leads to an optimal rank $M$ representation of the kernel [86].

A similar trigonometric basis is explored in the sparse spectrum GP (SSGP) [87] and the random Fourier features (RFF) approach [88], where the kernel is approximated with a cosine basis, leading to a finite-dimensional basis function model in terms of sines and cosines. In the SSGP, the frequencies of the basis are found through MLL optimization, whereas in the RFF approach, the frequencies are randomly sampled. Similar ideas can be found in the variational Fourier features (VFF) approach [89], which combines the variational framework of [81] with the Fourier representations of [87, 88]. Particularly, inducing features [90], are defined as a Fourier basis for the kernel on a compact domain. Since [75, 89] use harmonic frequencies on the domain, e.g., $\omega_m = \frac{2\pi m}{b-a}$, $m = 0, \ldots, M$ on $[a,b]$, the methods typically require a large $M$ to capture high-frequency content, similar to the inducing point approaches [91].
In this thesis, we are primarily interested in dynamical systems and their state. Dynamical systems can generally be thought of as systems that in some ways change over time, i.e., their state changes. Here, we primarily focus on systems that have a physically interpretable state, such as a vehicle, where the state could, for instance, represent the position and orientation of the vehicle. In this chapter, the necessary basics of dynamical systems and state estimation are covered. The material is by no means exhaustive but provides a solid foundation for understanding the publications relating to dynamical systems in this thesis. The material is based on [66, 92–94] and Paper D.

The chapter is organized as follows. Section 3.1 introduces the state-space model (SSM) that is a standard way of modeling dynamical systems, both from a functional perspective and also from a probabilistic viewpoint. Section 3.2 then describes the general process of state estimation in SSMs and in particular focuses on filtering.

### 3.1 State-Space Models

Dynamical systems can be modeled in a variety of ways. We will here focus our attention on the well-known SSM, as this fits well into the state estimation algorithms presented later on. Even though some systems are better described in continuous time, we will here work only with discrete-time systems. Here, the discrete-time SSM is assumed to be given by

\[
\begin{align*}
    x_{k+1} &= f_\theta(x_k, u_k, w_k) \\
    y_k &= h_\theta(x_k, u_k, e_k).
\end{align*}
\]

(3.1a)

(3.1b)

Here, \(x_k \in \mathbb{R}^{n_x}\), \(u_k \in \mathbb{R}^{n_u}\), and \(y_k \in \mathbb{R}^{n_y}\) denote the state, input, and observation at time-instance \(k\), respectively. Further, \(w_k \sim \mathcal{N}(w_k; 0, Q_k)\) and \(e_k \sim \mathcal{N}(e_k; 0, R_k)\)
are the process noise and observation noise, respectively. They are further assumed to be mutually uncorrelated. Lastly, $f_\theta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$ is the transition function and $h_\theta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_y}$ is the observation function. Both $f_\theta(\cdot)$ and $h_\theta(\cdot)$ are parametrized by a parameter $\theta$, which is either assumed to be known a priori or it needs to be identified from data. For notational convenience, the subscript $\theta$ will be suppressed in the following, if the context does not require it.

The transition model (3.1a) summarizes the dynamical behavior of the system in a compact mathematical construct. It is generally used to capture the physical assumptions of the system at hand, e.g., the effect of gravity on an object. The observation model (3.1b) relates the state of the system at time $k$, $x_k$, to the observed quantities $y_k$. In some systems and models, $y_k = x_k$ and $h(\cdot)$ is then simply the identity function. In applications such as target tracking, $y_k$ could be the range and bearing to the target, while $x_k$ could be the kinematic state of the target. The observation model (3.1b) then simply maps between the kinematic state and the range and bearing. Note that we will interchangeably use both observation and measurement to refer to some observed quantity according to (3.1b).

In the field of automatic control, the general SSM (3.1) is most often used to construct a principled way of selecting an input $u_k$ such as to achieve some goal [5]. This could for instance be to keep a vehicle at a certain speed, control the temperature in a building, or similar. For some applications, such as target tracking, the object of interest is the state of the system $x_k$. The model (3.1) is then used to construct algorithms that can estimate $x_k$ given only the observations $y_k$, so-called state estimation [92]. In general, both $f(\cdot)$ and $h(\cdot)$ are nonlinear functions, but in many applications, they are assumed to be linear, as this simplifies analysis and also leads to more efficient, often closed-form, algorithms for state estimation.

### 3.1.1 Probabilistic Modeling

The SSM (3.1) can equivalently be described in a probabilistic sense as [92]

$$
\begin{align*}
    x_{k+1} &\sim \rho_\theta(x_{k+1} | x_k, u_k) \\
y_k &\sim \rho_\theta(y_k | x_k, u_k).
\end{align*}
$$

(3.2a)

(3.2b)

The system dynamics are then expressed in terms of the transition density (3.2a) and the observations are expressed in terms of the observation density (3.2b). This alternative form lends itself well to deriving a variety of state estimation algorithms as well as parameter identification schemes. Similar results can be obtained from the functional form (3.1) but this is a matter of pure preference. One benefit of the probabilistic formulation is that the uncertainty of the resulting state estimate is captured completely by the posterior distribution, which is (theoretically) easily obtained through Bayes’ rule. In the following, the input $u_k$ will also be suppressed for notational convenience, unless explicitly needed.
State estimation refers to the problem of computing an estimate of the state of a system given observations \( y_{1:K} \). Here, \( y_{1:K} = [y_1^T \ y_2^T \ \cdots \ y_K^T] \) are the (ordered) observations from time \( k = 1 \) to time \( k = K \). Thus, given a sequence of observations of a system, the problem is now to compute an estimate of the system’s state at some time-instance. The most general form of state estimation considers the problem of estimating the entire history of system states, i.e., \( x_{0:K} = [x_0^T \ x_1^T \ \cdots \ x_K^T] \). The solution to this problem can be obtained by simple application of Bayes’ theorem. We thus wish to compute the conditional distribution of \( x_{0:K} \) given \( y_{1:K} \), i.e.,

\[
p(x_{0:K} | y_{1:K}) = \frac{p(y_{1:K} | x_{0:K}) p(x_{0:K})}{p(y_{1:K})}.
\]  

(3.3)

The prior distribution \( p(x_{0:K}) \) is simply given by the dynamics of the system, i.e.,

\[
p(x_{0:K}) = \prod_{k=0}^{K-1} p(x_{k+1} | x_k)p(x_0).
\]  

(3.4)

For the conditional distribution of \( y_{1:K} \) given \( x_{0:K} \), observe that a fundamental assumption in SSMs is that the observations are conditionally independent given the state trajectory of the system. Each observation \( y_k \) is further only dependent on the state of the system at the same time-instance, note from the model:

\[
p(y_k | x_k).\]

Hence, \( p(y_{1:K} | x_{0:K}) \) is simply the product of the observation density at each time-instance, i.e.,

\[
p(y_{1:K} | x_{0:K}) = \prod_{k=1}^{K} p(y_k | x_k).
\]  

(3.5)

Lastly, the (marginal) likelihood \( p(y_{1:K}) \) is given by marginalization over \( x_{0:K} \), i.e.,

\[
Z_{1:K} \triangleq p(y_{1:K}) = \int_{x_{0:K}} \prod_{k=1}^{K} p(y_k | x_k)p(x_{k+1} | x_k)p(x_0) \; dx_0 \cdots dx_K.
\]  

(3.6)

Note that \( p(x_{k+1} | x_k) \) is shifted to \( p(x_k | x_{k-1}) \), but it still represents the same fundamental construct. The posterior distribution over the state trajectory \( x_{0:k} \) is then given by

\[
p(x_{0:K} | y_{1:K}) = \frac{1}{Z_{1:K}} \prod_{k=1}^{K} p(y_k | x_k)p(x_{k+1} | x_k)p(x_0).
\]  

(3.7)

For many real-world applications, or more specifically for many control applications, the main interest lies with the state at the current time, i.e., the (marginal) posterior of the state \( x_K \) given \( y_{1:K} \). This is the distribution of interest in this thesis, and we will solely focus on that henceforth. The marginal posterior can be
obtained from (3.7) by marginalization, i.e.,
\[ p(x_k|y_{1:K}) = \int_{\mathbb{R}^n} p(x_{0:K}|y_{1:K}) \, dx_0 \cdots dx_{K-1}. \] (3.8)

At first sight, this may not seem convenient, as the posterior (3.7) is required. However, note that by inserting (3.7), after some manipulation, this can be rewritten as
\[ p(x_k|y_{1:K}) = \frac{\int_{\mathbb{R}^n} p(y_K|x_K) \int_{\mathbb{R}^n} p(x_K|x_{K-1}) \, dx_{K-1} \, dx_K \, p(x_{K-1}|y_{1:K-1}) \, dx_{K-1}}{Z_K}, \] (3.9)

where
\[ Z_K = \int_{\mathbb{R}^n} p(y_K|x_K) \int_{\mathbb{R}^n} p(x_K|x_{K-1}) \, dx_{K-1} \, dx_K. \] (3.10)

### Example 3.1: Kalman filter
To intuitively understand the KF, consider a dynamical model given by
\[
\begin{align*}
p(x_{k+1}|x_k) &= \mathcal{N}(x_{k+1}; 2x_k, 0.1) \\
p(y_k|x_k) &= \mathcal{N}(y_k; x_k, 0.2),
\end{align*}
\]
with a posterior at time \( k - 1 \) given by \( p(x_{k-1}|y_{1:k-1}) = \mathcal{N}(x_{k-1}; 1.0, 0.1) \). The measurement at time \( k \) is given by \( y_k = 3.0 \). One step of the KF applied to this model is visualized in Figure 3.1. The uncertainty is naturally captured through the “width” of each respective distribution, wider corresponding to a more uncertain estimate.

![Figure 3.1](image.png)

**Figure 3.1**: One step of the KF. The previous posterior \( p(x_{k-1}|y_{1:k-1}) \) is propagated through the time update to the predictive distribution \( p(x_k|y_{1:k-1}) \). The predictive distribution is then combined with the observation density \( p(y_k|x_k) \) through the measurement update to obtain the posterior \( p(x_k|y_{1:k}) \).
Note that (3.9) requires the posterior of the state at time $K-1$, i.e., $p(x_{K-1}|y_{1:K-1})$. From this, we note the two fundamental equations of state estimation. Firstly, the time update given by the Chapman-Kolmogorov equation

$$p(x_K|y_{1:K-1}) = \int_{\mathbb{R}^n} p(x_K|x_{K-1}) p(x_{K-1}|y_{1:K-1}) \, dx_{K-1}. \quad (3.11)$$

Secondly, the measurement update given by Bayes’ theorem

$$p(x_K|y_{1:K}) = p(y_K|x_K) p(x_K|y_{1:K-1}) \int_{\mathbb{R}^n} p(y_K|x_K) p(x_K|y_{1:K-1}) \, dx_K. \quad (3.12)$$

These two steps together form the Bayesian recursive filter and are typically alternated, i.e., first a time update is performed, then a measurement update, a time update, and so on.

Even though we now have the general filtering solution (3.9), it currently does not give us a means of actually computing the distribution. Depending on what densities we have chosen for $p(x_{k+1}|x_k)$ and $p(y_k|x_k)$, the posterior can be challenging to compute. A widely used assumption is that both $p(x_{k+1}|x_k)$ and $p(y_k|x_k)$ are affine Gaussian, i.e.,

$$p(x_{k+1}|x_k) = \mathcal{N}(x_{k+1}; F_k x_k + b_k, Q_k) \quad (3.13a)$$
$$p(y_k|x_k) = \mathcal{N}(y_k; H_k x_k + d_k, R_k). \quad (3.13b)$$

This corresponds to the two fundamental assumptions on the functional form (3.1):

1. $f(\cdot)$ and $h(\cdot)$ are affine and are given by $f(x_k) = F_k x_k + b_k$ and $h(x_k) = H_k x_k + d_k$.

2. The process noise $w_k$ and the observation noise $e_k$ are additive (in addition to being zero-mean Gaussian as we have already assumed).

With these two assumptions, the marginal posterior distribution (3.9) can be computed analytically and is given by the well-known KF, starting at some prior $p(x_0) = \mathcal{N}(x_0; \hat{x}_0|0, P_0|0)$. Note that the KF can be derived in a variety of other ways as well [48]. For completeness, the KF is given by Algorithm 1 and illustrated for intuition in Example 3.1.

### 3.2.1 Nonlinear State Estimation

Unfortunately, (3.13) does not hold in general, and more complex systems are better described by nonlinear dynamics and observation models. This complicates state estimation as it generally is no longer analytically tractable to compute the posterior distribution (3.9). A plethora of algorithms have been developed with numerous creative ways of approaching this problem. Here, we will only provide brief details on some of these approaches, as they relate to Papers C and D.
discuss some other alternatives in Section 3.2.2. Here, we focus on linearization-based approaches, that effectively replace the nonlinear models with linear surrogates by linearizing the models locally at each time-instance. For control engineers, this typically means analytical linearization, but there are other means of linearization as well.

**Linearization-based Approaches**

Linearization can be thought of quite generally as replacing a nonlinear function \( g(x) \) with an affine (possibly stochastic) approximation

\[
g(x) \approx Ax + b + w, \tag{3.14}
\]

with \( w \sim \mathcal{N}(\tilde{w}; 0, \Omega) \). The most widely used approximation utilizes analytical linearization, whereby the parameters \( A, b \) and \( \Omega \) are chosen as

\[
A = \frac{\partial}{\partial x} g(x)|_{x = \hat{x}}, \quad b = g(x)|_{x = \hat{x}} - Ax, \quad \Omega = 0, \tag{3.15}
\]

where \( \hat{x} \) is the linearization point. This produces an affine surrogate model that typically works well locally around \( \hat{x} \). Using this approximation strategy in the filtering posterior (3.9) directly leads to the well-known and widely used EKF [47].

A lesser-known strategy is that of statistical linearization, which instead of a point \( \hat{x} \) requires a full distribution \( p(x) = \mathcal{N}(x; \hat{x}, P) \). Statistical linearization
Algorithm 2 Linearization-based filter

Require: $\hat{x}_{0|0}, P_{0|0}, y_1:K, \{Q_k\}_{k=1}^{K}, \{R_k\}_{k=1}^{K}$

for $k = 1 : K$ do
  Compute $F_k, b_k, \Omega_k$ by linearizing $f(\cdot)$
  $\hat{x}_{k|k-1}, P_{k|k-1} \leftarrow \text{Time Update}(\hat{x}_{k-1|k-1}, P_{k-1|k-1}, F_k, b_k, Q_k + \Omega_k)$
  Compute $H_k, d_k, \Omega_k$ by linearizing $h(\cdot)$
  $\hat{x}_{k|k}, P_{k|k} \leftarrow \text{Measurement Update}(\hat{x}_{k|k-1}, P_{k|k-1}, H_k, d_k, R_k + \Omega_k)$
end for

return $\hat{x}_{k|k}, P_{k|k}$ ∀ $k \in [1, K]$

then finds the affine approximation (3.14) by minimizing the MSE between the true function $g(x)$ and its corresponding approximation, i.e., by solving [95]

$$\min_{A, b} E \left[ \|g(x) - (Ax + b)\|_2^2 \right].$$

(3.16)

Here, the expectation is taken w.r.t. $p(x)$ and $\|\cdot\|_2^2$ is the squared 2-norm. The solution to (3.16) is given by

$$A = \Psi^\top P^{-1}$$

(3.17a)

$$b = z - \bar{x}$$

(3.17b)

$$z \triangleq E [g(x)]$$

(3.17c)

$$\Psi \triangleq E \left[ (x - \bar{x})(g(x) - z)^\top \right].$$

(3.17d)

Further, the linearization error is given by

$$\Omega = \Phi - APA^\top$$

(3.18a)

$$\Phi \triangleq E \left[ (g(x) - z)(g(x) - z)^\top \right].$$

(3.18b)

Note that the expectations in (3.17) and (3.18) are generally not analytically available and are usually computed using some form of numerical integration [96], such as cubature, unscented transform, etc. The specific numerical integration technique is not integral to this thesis and will therefore not be detailed any further. The important takeaway is that choosing for instance the unscented transform [97] leads to the well-known UKF [98], while cubature leads to various versions of the CKF. Even the Gaussian particle filter (PF) [99] can be interpreted in this respect [100]. Example 3.2 considers both analytical and statistical linearization to provide intuition for how they differ qualitatively from one another.

The standard linearization-based filter utilizes either analytical or statistical linearization in each time instance, i.e., it first linearizes the dynamical model and uses the linearized model in the time update (3.11). Since the model is now linear, the time update is identical to that of the KF, see Algorithm 1. The filter then proceeds with linearizing the observation model and uses the linearized model in the measurement update (3.12), obtaining the KF measurement update,
see Algorithm 1. The qualitative difference between statistical and analytical linearization in the filtering context is the presence of the linearization error $\Omega$. If present, $\Omega$ can be viewed as a modified process and observation noise covariance, i.e., it is simply added to $Q_k$ and $R_k$, depending on the update at hand. The linearization-based filter is given in Algorithm 2, where the time and measurement update are given in Algorithm 1. It encapsulates the EKF, UKF, CKF, and other sigma-point-based filters, as these can be viewed as approximate KFs based on statistical linearization [98].

--- Example 3.2: Analytical and Statistical Linearization

To illustrate the difference between analytical and statistical linearization, we shall consider a 1D nonlinear function given by

$$g(x) = -x^2.$$  

Linearization is performed w.r.t. $p(x) = N(x; 0, 1, 0)$ with both first-order Taylor expansion as well as statistical linearization using unscented transform with parameters $\alpha = 1.732, \beta = 2, \kappa = -0.5$, corresponding to a weight of 1/3 on the central sigma point; see e.g. [101] for details on the parameters. The two linearization strategies end up in the two affine approximations

$$g_{\text{analytical}} = Ax + b$$
$$g_{\text{unscented}} = Cx + d,$$

where $A$ and $b$ are calculated using (3.15) and $C$ and $d$ are calculated using the unscented transform in (3.17). Figure 3.2 illustrates the two different linearized models that are obtained. The two models differ only in the obtained offsets $b$ and $d$, whereas $A = C$.

---

**Figure 3.2:** Analytical and statistical linearization based on the unscented transform of the function $g(x) = -x^2$. The blue dot is the mean around which the function is linearized. The orange triangles are the sigma points of the unscented transform.
3.2 State Estimation

3.2.2 Iterated Filtering

The approaches briefly detailed in Section 3.2.1 all rely on computing the posterior distribution through the lens of the time update (3.11) and the measurement update (3.12). By linearizing the nonlinear dynamics and observation models, this approach led to the EKF and various versions of the sigma-point filter, e.g., the UKF. To gain insight into the inner workings of the resulting algorithms, it is instructive to rather view the measurement update as minimizing

\[ \min_x \frac{1}{2} \left( \| y_k - h(x) \|_{R_k}^2 + \| \hat{x}_{k|k-1} - x \|_{P_{k|k-1}}^2 \right) = \min_x \frac{1}{2} r^T(x) r(x), \quad (3.19) \]

where \( \| A \|_W^2 = A^T W^{-1} A \) and

\[ r(x_k) = \begin{bmatrix} R_k^{-1/2}(y_k - h(x)) \\ P_{k|k-1}^{-1/2}(\hat{x}_{k|k-1} - x) \end{bmatrix}. \]

By minimizing (3.19) using analytical or statistical linearization, it is possible to show that the EKF and UKF are Gauss–Newton (GN) methods, performing one GN step in each measurement update [54, 55, 102]. Here, we will only briefly detail the EKF correspondence to build intuition and note that similar results hold for the UKF.

The EKF as GN

The GN method is a first-order iterative optimization method for nonlinear least squares problems, of the form (3.19) [103]. The GN method initializes an iterate \( x^0 \) and then updates this iterate by performing the update [94]

\[ x^{i+1} = x^i - \left( J_i^T J_i \right)^{-1} J_i^T r_i, \quad (3.20) \]

where \( r_i = r(x^i) \) and

\[ J_i \triangleq \frac{\partial r(x)}{\partial x} \bigg|_{x=x^i} = \begin{bmatrix} R_k^{-1/2} H_i \\ P_{k|k-1}^{-1/2} H_i \end{bmatrix}, \quad H_i \triangleq \frac{\partial h(x)}{\partial x} \bigg|_{x=x^i}. \quad (3.21) \]
The iterative process is visualized in Figure 3.3. The top plot shows the diﬀerential linearization of the observation model around the current iterate. This process can be interpreted as performing the measurement update multiple times, each time utilizing statistical linearization instead. These results can be obtained for the iterated extended Kalman filter (IEKF) and the iterated unscented Kalman filter (IUKF), and other so-called iterated ﬁlters [52–55]. These ﬁlters can be interpreted as performing the measurement update multiple times, each time re-linearizing the observation model around the current iterate $x^i$. Example 3.3 provides some graphical intuition about these algorithms.

--- Example 3.3: Iterated Filtering – IEKF

To illustrate how iterated ﬁltering works, we consider the observation model

$$y = h(x) + v = ax^3 + e,$$

with $a = 0.01$ and $e \sim \mathcal{N}(0, 0.1)$. The prior is assumed to be given by $p(x) = \mathcal{N}(3, 4)$ and the measurement is given by $y = 1.5$. The prior is updated multiple times through the EKF measurement update, i.e., an IEKF measurement update. The iterative process is visualized in Figure 3.3. The top plot shows the diﬀerent posteriors obtained as the ﬁlter iteratively reﬁnes the estimate. The middle plot shows the measurement function $h(x)$ and the linear approximations used in each iteration. Finally, the bottom plot shows the cost function (3.19) and the iterates produced in the updates. The iteration number is indicated in each plot, with 0 corresponding to the prior.
3.2 State Estimation

The iterative process refines the estimate by re-linearizing the observation model at each iterate. The plot visualizes the various approximate posteriors obtained during the iterative process. The middle plot illustrates the corresponding affine approximations used during the iterations together with the original nonlinear model \( h(x) \). Lastly, the bottom plot shows the cost function being optimized. Note that iteration 1 corresponds to a standard EKF.

Alternative State Estimation Approaches

While we only detail linearization-based approaches, the posterior (3.9) can be computed in other ways. One simple such approach is by approximating the previous posterior as

\[
p(x_{K-1}|y_{1:K-1}) \approx \sum_{i=1}^{M} w^i \delta(x^i - x_{K-1}),
\]

where \( \delta \) is the Dirac-delta function (a point mass) and \( w^i \) is a weighting of the \( i \)th point mass with \( \sum_{i=1}^{M} w^i = 1 \). The posterior is thus represented using a weighted sum of point masses, essentially “discretizing” the density. The weight \( w^i \) loosely...
represents the probability that the true $x_{K-1}$ is at $x^i$. This fundamentally simple approximation is the foundation of the point mass filter [104] as well as the PF [105, 106]. These methods have since been successfully used for, e.g., target tracking [107, 108] and learning of dynamical systems [109–111]. One of the primary reasons for using the linearization-based approaches we consider is that the point-mass-based methods suffer from the curse of dimensionality, meaning that their computational complexity grows exponentially in the dimension of the state. In the PF, the root cause for this problem is what is known as particle degeneracy [112], where the particle distribution “degenerates” to a single point mass, hindering its ability to represent arbitrary distributions. The computational complexity of point-mass-based methods has somewhat been remedied by the development of the Rao-Blackwellized/Marginalized PF [113], but this may still require a large number of “particles” to perform well. The particle degeneracy has instead been tackled by the particle flow filter [114], which “flows” particles from the prior to the posterior via an ordinary differential equation. This eliminates particle degeneracy, avoids resampling altogether, and is thus embarrassingly parallelizable. The flow is quite similar to likelihood tempering from sequential Monte Carlo methods [115–117], where the likelihood is, in a sense, progressively introduced.

Another alternative, which has recently gained attention, is the variational approach to state estimation. This approach aims to minimize some divergence measure, typically the KL divergence, between the true and an assumed posterior [118, 119]. This approach has been used to develop variational Gaussian filters [119], but has also recently been extended to Gaussian sum filters [120]. These approaches are generally applicable to any SSM but typically involve complex high-dimensional optimization problems. This is usually tackled through simple mean-field approximations, which on the other hand restricts the expressivity of the posterior.
Joint State Estimation and Function Approximation

This chapter presents one approach to combining the methods from Chapters 2 and 3 for learning parts of the system dynamics online, via standard state estimation methods. To that end, the chapter is organized as follows: Section 4.1 presents a general view of parameter identification (or model learning) in SSMs and details the particular variant exploited in this thesis. Section 4.2 then introduces the specific model that the thesis considers.

4.1 Learning in State-Space Models

We consider the probabilistic view of the SSM from Section 3.2, reiterated here for clarity, i.e., the model is given in terms of a transition and observation density

\[ x_{k+1} \sim p(x_{k+1}|x_k, \theta) \]
\[ y_k \sim p(y_k|x_k, \theta). \]

Here, the model is now explicitly parametrized by a parameter \( \theta \). Now, similar to Section 3.2, given an observation sequence \( y_{1:K} \), the problem amounts to finding an estimate of the state \( x \) as well as the parameter \( \theta \). We can approach the problem in the same way as in Section 3.2, i.e., we aim to find

\[
p(x_{0:K}, \theta|y_{1:K}) = \frac{p(y_{1:K}|x_{0:K}, \theta)p(x_{0:K}, \theta)}{p(y_{1:K})}.
\]  

(4.1)

Just like in Section 3.2, we are primarily interested in completely online solutions, which in the context of this thesis amounts to being able to discard an observation \( y_k \) after incorporation at time-instance \( k \). Note that other authors have other definitions of “online”. Hence, we are interested in the marginal posteriors

\[
p(x_K, \theta|y_{1:K}) = \frac{p(y_K|x_K, \theta)p(x_K, \theta|y_{1:K-1})}{Z_K},
\]

where \( Z_K \) is the normalizing constant.

43
where
\[ Z_K \triangleq \int_{[0,2\pi]^{2n} \times \mathbb{R}^n} p(y_K | x_K, \theta) p(x_K, \theta | y_{1:K-1}) \, d\theta \, dx_K. \]

A pragmatic solution to this problem is to treat the parameters \( \theta \) as a part of the state, the so-called state augmentation approach [92]. Then any of the filtering algorithms in Chapter 3 can be applied to estimate both the state \( x \) as well as the parameter \( \theta \). This is the approach that we pursue in Papers A and B. We do so primarily since it is an inherently online approach. Further, for a completely static model, this corresponds to a recursive least-squares estimate of the parameter \( \theta \) [92], identical to the batch solution, which lends credibility to the approach as such. Furthermore, this approach is widely used and has empirically worked well for numerous applications.

Other Approaches

While we use the pragmatic state augmentation approach, the arguably “correct” way of doing parameter identification in nonlinear SSMs is by considering the joint smoothing posterior (4.1). This can be tackled in a variety of ways, some of which we mention here. Technically, the joint posterior can be tackled directly by Markov chain Monte Carlo (MCMC) methods [121], which results in an extremely challenging high-dimensional sampling problem. One such approach, the particle MCMC methods [122], exploits the inherent Markovian structure of the SSM, by employing sequential Monte Carlo (SMC) methods to sample a state trajectory [106], and uses sampling methods such as Metropolis-Hastings for parameter inference [123, 124]. These methods, for instance, include particle marginal Metropolis-Hastings and particle Gibbs [122, 125].

A similar approach is the expectation maximization (EM) family of methods [126, 127], which optimizes a lower bound of the MLL. The expectation step infers a (smoothed) state trajectory given a certain parameter and can be performed with a variety of smoothers, e.g., an extended Kalman smoother [128], unscented Kalman smoother [129], particle smoother [111], etc. The maximization step maximizes the MLL w.r.t. the parameters and may offer analytical maximization for some special cases, e.g., for linear-in-the-parameters models [129]. In the general case, maximization is performed numerically or even partially through the Generalized EM algorithm [126]. A gradient-based method is explored in [130] and a type of block-optimization is pursued in [131], leading to the expectation-conditional-maximization method. The EM family of methods is extremely wide and new such methods can be obtained by replacing either the E-step or the M-step accordingly.

Lastly, direct MLL or maximum-a-posteriori (MAP) optimization is also sometimes a feasible approach, e.g., through the sensitivity equation approach [92, 132, 133]. This approach basically differentiates the standard filtering equations w.r.t. the parameters of interest and can thereby optimize the MLL.
4.2 Modeling

The general model under consideration here is given by

\[ x_{k+1} = f(x_k, w_k, g(x_k)) \quad (4.2a) \]
\[ y_k = h(x_k) + e_k. \quad (4.2b) \]

The observation model \( h(\cdot) \) is assumed to be known and is generally chosen as a linear model, even though this is not strictly necessary. The overall dynamics \( f(\cdot) \) of the system are assumed to be partly known through domain knowledge and \( g(\cdot) \) models any unknown dynamics that need to be inferred from data. The unknown dynamics \( g(\cdot) \) can technically be used to model anything unknown about the system and the inference procedures described in Papers A and B are not specific to the examples studied therein. The idea of the construction is essentially to utilize both prior domain knowledge as well as a data-driven component. Crucially, the data-driven component \( g(\cdot) \) is not restricted to be a mapping \( g: \mathbb{R}^{n_x} \to \mathbb{R}^{n_x} \), but can rather be restricted to a lower-dimensional subspace. This in turn yields substantial computational savings as compared to constructing a data-driven \( f(\cdot) \) directly.

In Papers A and B, we study a special case of this model which with a slight abuse of notation is given by

\[ x_{k+1} = f(x_k) + g_f(x_k)(w_k + g(x_k)) \quad (4.3a) \]
\[ y_k = h(x_k) + e_k. \quad (4.3b) \]

Here, \( f(\cdot) \) now represents the a priori known dynamics. Further, the mapping \( g_f(\cdot) \) is assumed to be known and is technically dependent on the choice of \( f(\cdot) \). This model can be interpreted as either: (i) \( g(\cdot) \) models the noise process and learns a “colored” noise or (ii) \( g(\cdot) \) learns an unknown input to the system, where the exact interpretation depends on \( f(\cdot) \) and \( g_f(\cdot) \). It is now also clear from (4.3) that \( g(\cdot) \) is restricted to the same subspace as the noise process. Example 4.1 provides a simple example of a possible model that adheres to the structure (4.3) and an explicit interpretation of \( g(\cdot) \).

**Example 4.1: Augmented constant velocity model**

Let the transition model \( f(\cdot) \) be given by a 1D constant velocity model and \( x = [p, v]^T \), where \( p \) is the position and \( v \) is the velocity. The mapping \( g_f(\cdot) \) is then linear and the complete transition model is given by

\[ x_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} (w_k + g(x_k)). \quad (4.4) \]

In (4.4), the function \( g(\cdot) \) is interpretable as the (deterministic) acceleration of the 1D system. Note that the process noise in this case will still deal with the stochasticity of the system.
Modeling the function $g(\cdot)$: The last question to resolve is how to model the function $g(\cdot)$. As alluded to in Section 4.1, we want to pursue a state augmentation approach to learn any parameters of the dynamics online. Thus, we require a parametric model for $g(\cdot)$. In this thesis, we particularly pursue approximate GPs as our go-to representation of $g(\cdot)$. The primary motivation for this is that GPs are naturally probabilistic models that fit well into the Bayesian frameworks of Chapter 3. In Paper A we use a FIC approach and in Paper B we instead use a BF expansion approach. These two models, in combination with the state augmentation approach, automatically yield an online algorithm for joint state estimation and model learning, through the nonlinear filtering techniques described in Chapter 3.

Discussion: We are primarily interested in online learning. The reasons for this are: (i) we do not assume access to historical data nor do we assume the possibility to influence the input to the system, (ii) even if we had historical data, this may not be entirely representative of the system behavior, necessitating online adaptation to new data, and (iii) we are not primarily interested in system identification, in the sense that we do not necessarily want to recover the “true” system, in a system identification sense [15]. These requirements necessitate models that are suitable for online updates and computationally efficient algorithms such that updates may be performed at a rate close to the data rate; in control applications, the data rate is often $\geq 100$ Hz. It is thus worthwhile to mention that the methods in Section 4.1 are primarily offline methods, even though pragmatic approximations with, e.g., sliding-window optimization exist [134]. The state augmentation approach on the other hand is inherently online, even though it, in general, may not converge to the “true” model, in a system identification sense [15]. Further, the model structures that we choose are inherently suited for online adaptation and fit well into the Bayesian filtering frameworks of Chapter 3, in contrast to, e.g., a neural network structure.

We end this chapter with Example 4.2, where we consider a simple 2D problem and apply our joint state estimation and model learning framework, where the unknown dynamics $g(\cdot)$ are modeled as a BF expansion.
Example 4.2

We assume that the true dynamics of the system are given by

\[ x_{k+1} = x_k + \Delta t A x_k, \quad A = \begin{bmatrix} -0.0375 & -1.0 \\ 1.0 & -0.0425 \end{bmatrix}, \quad \Delta t = 0.05. \]

The system is then modeled with

\[ x_{k+1} = x_k + \Delta t g(x_k) + w_k \\
\]

\[ g(x_k) = \Phi(x_k) \theta \\
\]

\[ y_k = x_k + e_k, \]

with \( w_k \sim N(w_k; 0, 0.01 I) \) and \( e_k \sim N(e_k; 0, 0.05^2 I) \). Further, the BFs are chosen as RBFs similar to Example 2.2, with 25 equispaced RBFs placed within \([-2, 2] \times [-2, 2]\), see Figure 4.1. The true system is simulated for \( N = 400 \) time-instances from an initial state drawn from \( x_0 \sim N(x_0; [-1, -1]^T, 0.1 I) \). The augmentation approach from Section 4.1 is used for the parameters of the expansion and an EKF is applied to the augmented model to learn the (partly) unknown dynamics. Note that the vector field is 2D, and therefore \( g(\cdot) \) is a vector-valued function. The ground truth vector field and learned vector fields are visualized in Figure 4.1. The norm of the vector field is also visualized in Figure 4.2(a). The ground truth is shown in the left plot of Figures 4.1 and 4.2(a) and the estimated system trajectory in the right, respectively. The RBF centers are also visualized in the right plot of Figures 4.1 and 4.2(a).

Figure 4.1: Left: the ground truth vector field and system trajectory. Note that the trajectory slowly approaches the origin. Right: the estimated vector field and system trajectory.
Figure 4.2: Norm of the unknown part of the dynamics. Note that the estimated field norm can only be expected to be accurate close to the observed system trajectory. Further, the norm is clipped to 2.5 for visualization purposes.

(a) Left: ground truth vector field norm and system trajectory. Right: estimated vector field norm and system trajectory.

(b) The recursive learning procedure over time. The stages illustrate the learned field from $k = 0$ to $k = 300$. The estimated trajectory is also shown in blue along with the BF centers in black.
Concluding Remarks

We conclude Part I by summarizing the main contributions of the publications contained in Part II. We further provide some suggested future research directions for the covered topics.

5.1 Summary of Contributions

The main contributions of each of the papers in Part II are explicitly listed here. The contributions themselves can be broadly categorized into three distinct topics, which are all treated separately. The three topics separately and directly address Problems (i)–(iii), respectively in order.

5.1.1 Online Learning in Augmented Grey-box Models

Papers A and B are both concerned with online learning of parts of the system dynamics. The specific contributions of the two papers are as follows.

Paper A

(A1) A grey-box model consisting of an SSM augmented with a GP is proposed for modeling systems with partly known dynamics.

(A2) An inference scheme, based on an EKF, is detailed, which allows for online learning of the unknown system dynamics.

(A3) The model and inference scheme is shown to reduce the estimation error as compared to using only the a priori specified part of the grey-box model, i.e., the SSM.
Paper B

(B1) A compact RBF expansion and an approximate EKF gain are proposed as a way of scaling the method from Paper A to large-scale problems.

(B2) The method is shown to be more-or-less computationally invariant to the number of model parameters, making it real-time applicable, i.e., executable at ≈ 25 Hz in a naive Python implementation, even for large-scale systems with tens of thousands of parameters.

(B3) The proposed method approximately retains the same function approximation performance as Paper A while easily scaling to larger problems.

Summary

By learning parts of the model online and improving the overall model fidelity, the state estimation performance is indirectly improved over time. In particular, the developed methods provide a way of combining domain knowledge-based dynamical models with GPs in a computationally efficient manner.

5.1.2 Iterated Linearization-based Filtering

Papers C and D are concerned with iterated linearization-based filtering, a widely studied topic. The specific contributions of the two papers are as follows.

Paper C

(C1) An exact correspondence between iterated statistical linearization and QN is detailed, which allows for a QN interpretation of the IUKF, IPLF, and similar filters.

(C2) The QN interpretations open up for, e.g., efficient damping strategies to be developed that exploit this correspondence, which is exploited in Paper D.

Paper D

(D1) A unified framework of a dynamically iterated filter (DIF) is developed. The framework, as opposed to most previous iterated filters, takes nonlinearities in the dynamical evolution of the system into account.

(D2) The framework allows for the development of dynamically iterated analogs of essentially all previously proposed linearization-based iterated filters.

(D3) The DIFs are shown to have superior robustness properties and estimation performance as compared to standard linearization-based filters as well as standard iterated analogs thereof.
Summary

We study connections between existing linearization-based algorithms that lead to interpretations in terms of classical optimization algorithms, improving the understanding of the properties of these filtering algorithms. Further, new algorithms are suggested that directly improve the state estimation performance in models where the primary nonlinearities are contained in the dynamics, e.g., the models from Papers A and B, while retaining the computational efficiency inherent to linearization-based methods.

5.1.3 Computational Efficiency in Gaussian Processes

Papers E and F study approximate GPs with a particular focus on the HGP, which is an excellent approximate GP in low-dimensional applications such as geostatistics. The specific contributions of the two papers are as follows.

Paper E

(E1) The HGP is shown to have a Toeplitz/Hankel structure. The structures enable a reduction of the computational complexity of computing the posterior precision matrix from $O(NM^2)$ to $O(NM)$.

(E2) Similar structures are shown to exist in a wide variety of other sparse GP approximations which generalizes the computational complexity reduction to a wide range of previous work.

(E3) The storage complexity of aforementioned GP approximations is also reduced from $O(M^2)$ to $O(M)$. In conjunction with the reduction in computational complexity, this significantly improves the practical usability of these approximations, even with limited hardware.

Paper F

(F1) A method for prediction-point-dependent model reduction in BF expansions is proposed. The method enables computationally efficient approximate predictions in BF expansions while retaining the predictive quality.

(F2) It is shown that, due to the particular model structure under study, the model reduction can be performed without additional data, as opposed to previously proposed reduction strategies. This allows for “online” model reduction, which adapts to a user-specified input region.

Summary

Papers E and F improve the computational efficiency of the HGP in different respects, while also generalizing to other related BF expansion models. The improved efficiency allows the HGP to scale to new problem sizes. Further, the developments herein may also be used to reduce the amount of communication
necessary for distributed implementations of HGP regression, important for, e.g., multi-agent SLAM applications. Lastly, the synergy between Papers E and F is quite interesting. In a sense, Paper E focuses on improving the computational complexity of learning in the approximate GP model, whereas Paper F improves the computational complexity of inference in the model. Thus, they jointly tackle both of the major hurdles of computational complexity in approximate GPs.

5.2 Future Research Directions

In addition to future work that is pointed out in the publications in Part II, we here discuss a few general directions for the topics covered in the thesis.

Online Learning in Augmented Grey-box Models

A natural next step is to replace the EKF with more sophisticated filtering techniques. In this respect, the filtering algorithms in Papers C and D are of immediate interest. Other choices could be to move to SMC methods as detailed in Section 3.2.2, which unfortunately drastically increases the computational complexity. As the considered models are conditionally linear in the unknown dynamics, Rao-Blackwellization may help here. Regardless of the chosen estimator, the methods developed in Papers A and B are inherently unimodal, meaning that they cannot deal with possible ambiguity in the unknown dynamics, such as the “singularity” in Paper A. This can possibly be dealt with through multiple models, in a sense close to the ideas of the interacting multiple model filter [135, 136]. This will again unavoidably increase the computational complexity. However, exploiting the developments in Papers E and F may help in this regard, possibly enabling multiple-model learning at a reasonable cost. Another avenue to explore is distributed learning, which can be applied to more or less any of the applications considered in Papers A and B. Here, Paper E can help reduce the necessary communication. Lastly, incorporating some type of online hyperparameter learning, e.g., through memory [137], would greatly improve the practical usability of the models.

Iterated Linearization-based Filtering

Extending the methods to Gaussian mixture filtering could be one avenue of exploration. While iterated filtering has been developed for Gaussian mixture filtering [120], dynamically iterated analogs are still missing and should be straightforwardly available through the lens of Paper D. Exploring connections to the recently proposed Bayesian recursive update filter (BRUF) [138] may also be of interest. In [138], the BRUF is pointed out to (essentially) be a single particle implementation of a particle flow filter. In light of this connection, it should be possible to interpret other iterated filters in the same respect, which could lead to (pseudo-time) “continuous” versions of, e.g., the IEKF. Further, even though some intuition is given in Paper D as to what the developed algorithms are useful for, the algorithms are yet to be tested on a variety of applications. Lastly, while it
is not mentioned in Paper D, the algorithms were originally developed for “hyper-
local” EM-like parameter identification which remains largely unexplored (even 
though some results in this respect are lying dormant in the author’s drawer).

Computational Efficiency in Gaussian Processes

While the results in Paper E increase the practical usability for a multitude of 
GP approximations, the dominating computational complexity has essentially 
moved from the data-dependent $O(NM^2)$ term to the data-independent $O(M^3)$ 
term. Since Paper E identifies Toeplitz/Hankel structure, the next natural step 
is to exploit (iterative) Toeplitz/Hankel solvers in the inference step [139, 140], 
which could reduce the $O(M^3)$ cost drastically with little to no loss in accuracy. 
This could also yield faster hyperparameter optimization which is currently still 
at cost $O(M^3)$, limiting the methods to “small” number of BFS $M$, hindering appli-
cability to very large problems. Because of the increased efficiency, the HGP can 
now also be used as a drop-in replacement for the approximate GP in Papers A 
and B, which may alleviate some of the issues detailed in Paper B. This can then 
lead to indirect improvements in the state estimation performance by improving 
the properties of the model itself. While the ideas of Papers E and F seem di-
different at first, combining the two leads to extremely fast (approximate) predictions 
while being able to maintain a very large $M$. This is already implemented and 
exploited in Paper F, but still lacks a proper complexity comparison. Lastly, it 
is briefly mentioned in Paper F that the method can be used to perform approx-
imate model order reduction. However, this deserves proper investigation and 
comparison to previously established reduction methods, to identify the exact 
effects of the approximations in this respect.
Bibliography


Part II

Publications
Papers

The papers associated with this thesis have been removed for copyright reasons. For more details about these see:

https://doi.org/10.3384/9789180756747
Dynamic rEvolution
Adaptive state estimation via Gaussian processes and iterative filtering

Anton Kullberg