

Linköping studies in science and technology. Dissertations.  
No. 1161

# **Performance and Implementation Aspects of Nonlinear Filtering**

**Gustaf Hendeby**



Department of Electrical Engineering  
Linköping University, SE-581 83 Linköping, Sweden  
Linköping 2008

**Cover illustration:** The figures visible on the front and back of the cover was created with MATLAB® and shows the PDF of the distribution

$$0.3\mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.61 & 0.19 \\ 0.19 & 0.65 \end{pmatrix}\right) + 0.7\mathcal{N}\left(\begin{pmatrix} 2.1 \\ 2.4 \end{pmatrix}, \begin{pmatrix} 0.62 & 0.14 \\ 0.14 & 0.43 \end{pmatrix}\right)$$

from two directions. The dots below the PDF surface are 10 000 particles representing the distribution.

Linköping studies in science and technology. Dissertations.  
No. 1161

**Performance and Implementation Aspects of Nonlinear Filtering**

Gustaf Hendeby

*hendeby@isy.liu.se*  
*www.control.isy.liu.se*  
*Division of Automatic Control*  
*Department of Electrical Engineering*  
*Linköping University*  
*SE-581 83 Linköping*  
*Sweden*

ISBN 978-91-7393-979-9

ISSN 0345-7524

Copyright © 2008 Gustaf Hendeby

Printed by LiU-Tryck, Linköping, Sweden 2008

*To my secret love...*



# Abstract

Nonlinear filtering is an important standard tool for information and sensor fusion applications, e.g., localization, navigation, and tracking. It is an essential component in surveillance systems and of increasing importance for standard consumer products, such as cellular phones with localization, car navigation systems, and augmented reality. This thesis addresses several issues related to nonlinear filtering, including performance analysis of filtering and detection, algorithm analysis, and various implementation details.

The most commonly used measure of filtering performance is the *root mean square error* (RMSE), which is bounded from below by the *Cramér-Rao lower bound* (CRLB). This thesis presents a methodology to determine the effect different noise distributions have on the CRLB. This leads up to an analysis of the *intrinsic accuracy* (IA), the informativeness of a noise distribution. For linear systems the resulting expressions are direct and can be used to determine whether a problem is feasible or not, and to indicate the efficacy of nonlinear methods such as the *particle filter* (PF). A similar analysis is used for change detection performance analysis, which once again shows the importance of IA.

A problem with the RMSE evaluation is that it captures only one aspect of the resulting estimate and the distribution of the estimates can differ substantially. To solve this problem, the *Kullback divergence* has been evaluated demonstrating the shortcomings of pure RMSE evaluation.

Two estimation algorithms have been analyzed in more detail; the *Rao-Blackwellized particle filter* (RBPF), by some authors referred to as the *marginalized particle filter* (MPF), and the *unscented Kalman filter* (UKF). The RBPF analysis leads to a new way of presenting the algorithm, thereby making it easier to implement. In addition the presentation can possibly give new intuition for the RBPF as being a stochastic Kalman filter bank. In the analysis of the UKF the focus is on the *unscented transform* (UT). The results include several simulation studies and a comparison with the *Gauss approximation* of the first and second order in the limit case.

This thesis presents an implementation of a parallelized PF and outlines an object-oriented framework for filtering. The PF has been implemented on a *graphics processing unit* (GPU), i.e., a graphics card. The GPU is an inexpensive parallel computational resource available with most modern computers and is rarely used to its full potential. Being able to implement the PF in parallel makes new applications, where speed and good performance are important, possible. The object-oriented filtering framework provides the flexibility and performance needed for large scale Monte Carlo simulations using modern software design methodology. It can also be used to help to efficiently turn a prototype into a finished product.



# Populärvetenskaplig sammanfattning

I många fall är det viktigt att kunna få ut så mycket och så bra information som möjligt ur tillgängliga mätningar. Att utvinna information om till exempel position och hastighet hos ett flygplan kallas för *filtrering*. I det här fallet är positionen och hastigheten exempel på *tillstånd* hos flygplanet, som i sin tur är ett *system*. Ett typiskt exempel på problem av den här typen är olika övervakningssystem, men samma behov blir allt vanligare även i vanliga konsumentprodukter som mobiltelefoner (som talar om var telefonen är), navigationshjälpmedel i bilar och för att placera upplevelseförhöjande grafik i filmer och TV-program. Ett standardverktyg som används för att extrahera den information som behövs är olineär filtrering. Speciellt vanliga är metoderna i positionerings-, navigations- och målföljningstillämpningar. Den här avhandlingen går in på djupet på olika frågeställningar som har med olineär filtrering att göra:

- Hur utvärderar man hur bra ett filter eller en detektor fungerar?
- Vad skiljer olika metoder åt och vad betyder det för deras egenskaper?
- Hur programmerar man de datorer som används för att utvinna informationen?

Det mått som oftast används för att tala om hur effektivt ett filter fungerar är RMSE (*root mean square error*), som i princip är ett mått på hur långt ifrån det korrekta tillståndet man i medel kan förvänta sig att den skattning man får är. En fördel med att använda RMSE som mått är att det begränsas av *Cramér-Raos undre gräns* (CRLB). Avhandlingen presenterar metoder för att bestämma vilken betydelse olika brusfördelningar har för CRLB. Brus är de störningar och fel som alltid förekommer när man mäter eller försöker beskriva ett beteende, och en brusfördelning är en statistisk beskrivning av hur bruset beter sig. Studien av CRLB leder fram till en analys av *intrinsic accuracy* (IA), den inneboende noggrannheten i brus. För lineära system får man rättframma resultat som kan användas för att bestämma om de mål som satts upp kan uppnås eller inte. Samma metod kan också användas för att indikera om olineära metoder som *partikelfiltret* kan förväntas ge bättre resultat än lineära metoder som *kalmanfiltret*. Motsvarande metoder som är baserade på IA kan även användas för att utvärdera detektionsalgoritmer. Sådana algoritmer används för att upptäcka fel eller förändringar i ett system.

När man använder sig av RMSE för att utvärdera filtreringsalgoritmer fångar man upp en aspekt av filtreringsresultatet, men samtidigt finns många andra egenskaper som kan vara intressanta. Simuleringar i avhandlingen visar att även om två olika filtreringsmetoder ger samma prestanda med avseende på RMSE så kan de tillståndsfördelningar de producerar skilja sig väldigt mycket åt beroende på vilket brus det studerade systemet utsätts för. Dessa skillnader kan vara betydelsefulla i vissa fall. Som ett alternativ till RMSE används därför här *kullbackdivergensen* som tydligt visar på bristerna med att bara förlita sig på RMSE-analyser. Kullbackdivergensen är ett statistiskt mått på hur mycket två fördelningar skiljer sig åt.

Två filtreringsalgoritmer har analyserats mer i detalj: det *rao-blackwelliserade partikelfiltret* (RBPF) och den metod som kallas *unscented Kalman filter* (UKF). Analysen av RBPF leder fram till ett nytt sätt att presentera algoritmen som gör den lättare att använda i ett datorprogram. Dessutom kan den nya presentationen ge bättre förståelse för hur algoritmen fungerar. I undersökningen av UKF ligger fokus på den underliggande så

kallade *unscented transformation* som används för att beskriva vad som händer med en brusfördelning när man transformerar den, till exempel genom en mätning. Resultatet består av ett antal simuleringsstudier som visar på de olika metodernas beteenden. Ett annat resultat är en jämförelse mellan UT och *Gauss approximationsformel* av första och andra ordningen.

Den här avhandlingen beskriver även en parallell implementation av ett partikelfilter samt ett objektorienterat ramverk för filtrering i programmeringsspråket C++. Partikelfiltret har implementerats på ett grafikkort. Ett grafikkort är ett exempel på billig hårdvara som sitter i de flesta moderna datorer och mest används för datorspel. Det används därför sällan till sin fulla potential. Ett parallellt partikelfilter, det vill säga ett program som kör flera delar av partikelfiltret samtidigt, öppnar upp för nya tillämpningar där snabbhet och bra prestanda är viktigt. Det objektorienterade ramverket för filtrering uppnår den flexibilitet och prestanda som behövs för storskaliga Monte-Carlo-simuleringar med hjälp av modern mjukvarudesign. Ramverket kan också göra det enklare att gå från en prototyp av ett signalbehandlingssystem till en slutgiltig produkt.

# Acknowledgments

This book concludes five years of my life filled with blood, sweat, and tears (all those paper cuts...) and I'm proud to say that I spent it at the Automatic Control group in Linköping. Not only does the group provide excellent possibilities in research, more importantly all the people working there are simply great. I will be forever thankful for all discussions in the "fika room" (where else would I have learned how to escape from a Siberian prisoners camp?), the encouragement, well pretty much everything!

Some persons have been more important than others for my success in research than others, most directly; Prof. Fredrik Gustafsson, my supervisor and a constant source of ideas, inspiration, and support since my master's thesis; Dr. Rickard Karlsson, my co-supervisor filled with great ideas and encouragement, and with an immense patience when it comes to my swiftly changing ideas affecting him — you even *git* it all; Prof. Lennart Ljung, a truly great man, not only in research but also as a leader, the trust and confidence you have put in me in everything since I started in the group has really helped me grow as a person (though it cost me some late nights catching up on things I should have done instead); Ulla Salaneck — You are the best! — you make sure everything works smoothly, without you Automatic Control in Linköping would not be the same.

Lots of people has proofread this thesis or parts of it, their input and suggestions have been invaluable, though, let me point out all mistakes are my own, but fewer thanks to these guys: Dr. Martin Enqvist, Rickard, Sara Rassner, Dr. Thomas Schön, Henrik Tidefelt, and David Törnqvist.

Some more important names, you are all great friends and have all been part of this experience in various ways: David, Johan Sjöberg, and Daniel Axehill: we all started this together, and we end it together. Henrik Tidefelt I love our CS discussions — we **will** Shape it. Henrik Ohlsson, thanks for making me sweat at the gym and reintroducing me to Tuesdays at HG... Christian Lyzell, you'll always be my guitar hero! Dr. Erik Geijer Lundin and Dr. Markus Gerdin — why did you leave? I miss our after office-hours discussions at the office. Martin, Rickard and Dr. Mikael Norrlöf thank you for numerous futile attempts to limit my pet projects.

The funding has been supplied by: VINNOVA's Center of Excellence ISIS (Information Systems for Industrial Control and Supervision) at Linköpings universitet; SSF (Swedish Foundation for Strategic Research) Strategic Research Center MOVIII; and VR (the Swedish Research Council) project Sensor Informatics. You are gratefully acknowledged, without your support I wouldn't have had the chance to do this. And I would also like to thank my co-authors of various papers: Dr. Neil Gordon, Fredrik, Jeroen Hol, and Rickard you have been a great source of knowledge and inspiration. It has been a pleasure working with you all and I hope this doesn't have to be the end of our cooperation.

Last but not least, I'd like to thank my family and the rest of my friends for being there for me! My parents Henrik and Elisabeth, and my sisters Anna and Elvira, have always supported me in my quest for more knowledge and in life! As important all friends, providing me with a reason to have yet another beer and to engage in strange pranks to make it through tough times. (u know who u r :) Sara, you are still a biologist, but also always a good support and always willing to listen to me complain. Best of luck to you in your research!

Linköping, February 2008

*Gustaf Hendeby*



---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivating Example: Bearings-Only Tracking . . . . .	1
1.2	Problem Formulation . . . . .	4
1.3	Contributions . . . . .	5
1.4	Thesis Outline . . . . .	7
<b>2</b>	<b>Statistical Properties</b>	<b>9</b>
2.1	Stochastic Variables . . . . .	9
2.2	Information in Distributions . . . . .	11
2.2.1	Fisher Information . . . . .	11
2.2.2	Intrinsic Accuracy . . . . .	13
2.2.3	Relative Accuracy . . . . .	13
2.2.4	Accuracy Properties . . . . .	14
2.3	Kullback-Leibler Information . . . . .	16
2.4	Analysis of Selected Distributions . . . . .	18
2.4.1	Gaussian Distribution . . . . .	18
2.4.2	Gaussian Sum Distribution . . . . .	19
2.4.3	Generalized Gaussian Distribution . . . . .	24
2.4.4	Normal Inverse Gauss . . . . .	26
<b>3</b>	<b>Functions of Distributions</b>	<b>29</b>
3.1	Analytical Transformation . . . . .	29
3.2	Monte Carlo Transformation . . . . .	31
3.3	Gauss Approximation Formula . . . . .	33
3.3.1	First Order Gauss Approximation . . . . .	33
3.3.2	Second Order Gauss Approximation . . . . .	34
3.4	Unscented Transform . . . . .	35

---

3.5	Asymptotic Analysis of the Unscented Transform . . . . .	40
3.6	Comparative Example . . . . .	42
<b>4</b>	<b>Models</b>	<b>45</b>
4.1	General Model . . . . .	46
4.1.1	Hidden Markov Model . . . . .	46
4.1.2	State-Space Model . . . . .	46
4.2	Specific State-Space Model . . . . .	48
4.2.1	Linear Model . . . . .	48
4.2.2	Switched Models . . . . .	50
4.2.3	Model with Linear Gaussian Substructure . . . . .	51
4.3	Fault Model . . . . .	53
<b>5</b>	<b>Filtering Methods</b>	<b>55</b>
5.1	Estimation Preliminaries . . . . .	56
5.1.1	Parameter Estimation . . . . .	56
5.1.2	Dynamic Estimation . . . . .	58
5.2	Particle Filter . . . . .	58
5.2.1	Approximate Probability Density Function . . . . .	59
5.2.2	Resampling . . . . .	62
5.3	Kalman Filter . . . . .	65
5.4	Kalman Filters for Nonlinear Models . . . . .	67
5.4.1	Linearized Kalman Filter . . . . .	67
5.4.2	Extended Kalman Filter . . . . .	69
5.4.3	Iterated Extended Kalman Filter . . . . .	70
5.4.4	Unscented Kalman Filter . . . . .	74
5.5	Filter Banks . . . . .	77
5.5.1	Complete Filter Bank . . . . .	78
5.5.2	Filter Bank with Pruning . . . . .	79
5.5.3	Filter Bank with Merging . . . . .	81
5.6	Rao-Blackwellized Particle Filter . . . . .	83
5.6.1	Performance Gain . . . . .	83
5.6.2	Rao-Blackwellization for Filtering . . . . .	84
5.6.3	Rao-Blackwellized Filtering with Linear Gaussian Structure . . . . .	86
<b>6</b>	<b>Cramér-Rao Lower Bound</b>	<b>93</b>
6.1	Parametric Cramér-Rao Lower Bound . . . . .	94
6.2	Posterior Cramér-Rao Lower Bound . . . . .	95
6.3	Cramér-Rao Lower Bounds for Linear Systems . . . . .	96
6.4	Summary . . . . .	101
<b>7</b>	<b>Filter Performance</b>	<b>103</b>
7.1	Multimodal Posterior . . . . .	104
7.2	Altitude Based Terrain Navigation . . . . .	105
7.3	Range-Only Tracking . . . . .	108
7.4	Recursive Triangulation Using Bearings-Only Sensors . . . . .	110
7.4.1	Sensor Model . . . . .	110

7.4.2	Simulations . . . . .	111
7.4.3	Conclusions . . . . .	115
7.5	DC Motor . . . . .	116
7.5.1	Measurements with Outliers . . . . .	117
7.5.2	Load Disturbances . . . . .	118
7.5.3	Conclusion . . . . .	119
7.6	Target Tracking . . . . .	119
7.7	Observations . . . . .	122
<b>8</b>	<b>Change Detection</b>	<b>123</b>
8.1	Hypothesis Testing . . . . .	124
8.2	Test Statistics . . . . .	126
8.2.1	Likelihood Ratio Test . . . . .	126
8.2.2	Generalized Likelihood Ratio Test . . . . .	128
8.2.3	Bayes Factor Test . . . . .	128
8.3	Most Powerful Detector . . . . .	128
8.4	Asymptotic Generalized Likelihood Ratio Test . . . . .	130
8.4.1	Wald Test . . . . .	131
8.4.2	Detection Performance . . . . .	132
8.5	Uniformly Most Powerful Test for Linear Systems . . . . .	134
8.5.1	Linear System Residuals . . . . .	134
8.5.2	Prior Initial State Knowledge . . . . .	135
8.5.3	Parity Space . . . . .	136
8.6	Summary . . . . .	137
<b>9</b>	<b>Detection Performance</b>	<b>139</b>
9.1	Constant Velocity Model . . . . .	139
9.1.1	Bimodal Measurement Noise . . . . .	141
9.1.2	Tri-Gaussian Process Noise . . . . .	142
9.1.3	Conclusions . . . . .	143
9.2	DC Motor . . . . .	143
9.2.1	Measurements with Outliers . . . . .	144
9.2.2	Load Disturbances . . . . .	145
9.2.3	Conclusions . . . . .	146
9.3	Range-Only Simulation . . . . .	146
9.4	Summary . . . . .	148
<b>10</b>	<b>Implementation Details</b>	<b>149</b>
10.1	A C++ filtering framework — F++ . . . . .	149
10.1.1	Design . . . . .	150
10.1.2	Class Description . . . . .	150
10.1.3	Example: Range-Only Measurement Tracking . . . . .	155
10.2	Using Graphics Hardware for Particle Filtering . . . . .	157
10.2.1	General Purpose Graphics Programming . . . . .	158
10.2.2	GPU Based Particle Filter . . . . .	162
10.2.3	Evaluation Using Range-Only Measurement Example . . . . .	166

<b>11 Concluding Remarks</b>	<b>169</b>
11.1 Results . . . . .	170
11.2 Future Work . . . . .	171
<b>A Notational Conventions</b>	<b>173</b>
<b>B F++ Listings</b>	<b>177</b>
<b>Bibliography</b>	<b>185</b>

# 1

---

## Introduction

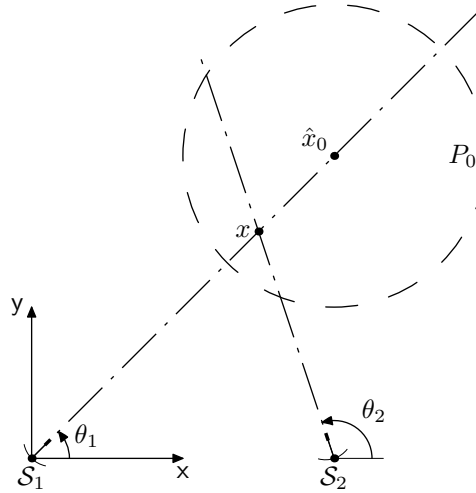
**I**NFORMATION IS BECOMING more and more important in society today, and it is possible to obtain more measurements than it is possible to make use of. With these large quantities of data collected, however usually not exactly what is needed, it is not always an easy task to extract the necessary information. Therefore, the science of how to process measurements to get interesting information of high quality is a prioritized research area. This thesis deals with different aspects of this. Given measurements and a model of how they were created, try to answer questions such as:

- How much information is available in the data? And what is a good measure for this?
- How to efficiently retrieve interesting information?
- When is it possible to determine if there has been an unexpected change?

This chapter starts with a motivational example in Section 1.1 followed by a problem description in Section 1.2. Section 1.3 then lists made contributions, and Section 1.4 gives an outline of the thesis.

### 1.1 Motivating Example: Bearings-Only Tracking

Consider the problem of tracking an airplane or a ship using sensors that only deliver bearing to the object, or bearing and very uncertain range information. This is a classic tracking problem. Many different types of sensors only give information about bearing. Traditionally, bearings-only sensors such as passive radars, for tracking aircraft without being noticed, and sonar buoys, used to follow maritime vessels, have been considered. However, sensors that only give direction appear in many other situations as well. An example of this is measurements derived from images. One place where cameras can be found is in systems for situational-awareness for automotive collision avoidance, such



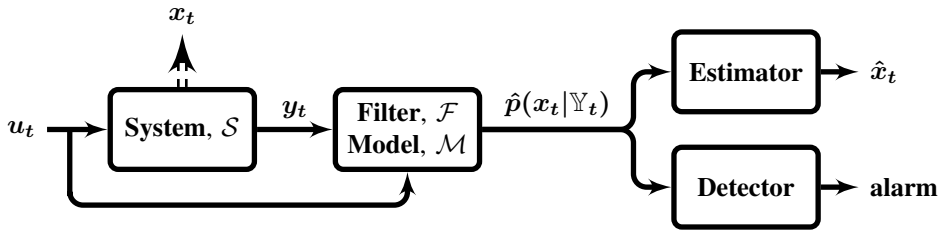
**Figure 1.1:** Bearings-only problem, with two measurements from  $S_1$  and  $S_2$ . The true target location  $x$  and initial estimate  $\hat{x}_0$ , with covariance  $P_0$ .

systems often combine information from infrared sensors and/or vision systems. These are both sensors which deliver good directional information but poor or no information about the distance to the target. Another example is using video footage to track sportmen's movements on the sports field.

A principal illustration of the triangulation problem is depicted in Figure 1.1, where two bearings-only sensors try to determine the position of a target denoted  $x$ . The sensors are located some distance apart in  $S_1$  and  $S_2$ . This setup is simplified, but can be interpreted in many ways:

- The classical triangulation problem, where two sensors simultaneously measure the direction to the same stationary target and where the target position can be derived geometrically from the measurements.
- It is also possible to assume that the measurements are taken at different times, which does not change the problem unless the target is moving. This could be the case either if the sensor is moving or the target is moving, with known velocity. An example of the former could be using a moving camera observing landmarks for positioning which is similar to traditional maritime navigation.
- Extending the problem somewhat, the target and/or the sensor could be moving in a more or less unknown way, as is the case in target tracking where the target movements are unknown.

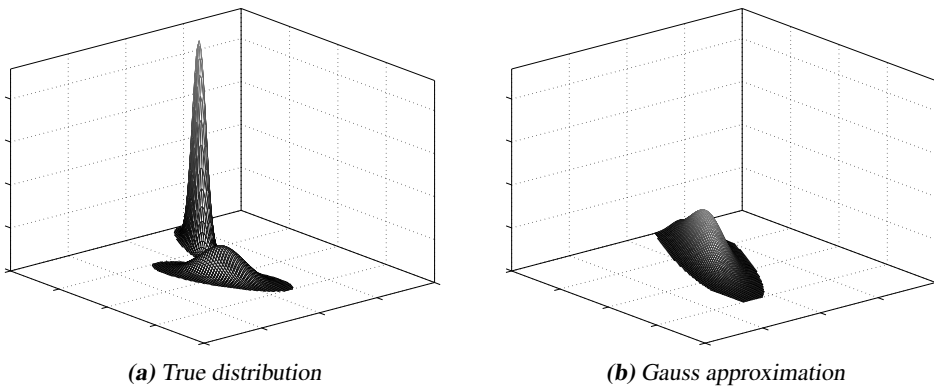
Figure 1.2 tries to illustrate the major components of a signal processing problem such as the one just presented. The system is what is described in Figure 1.1, *i.e.*, a description of how the target moves and how measurements are acquired. The model describes the system in a mathematical way that can be used by a filter to obtain an estimate of the posterior distribution  $p(x_t | \mathbb{Y}_t)$  that collects all information known about the system. The posterior distribution can then be used to get:



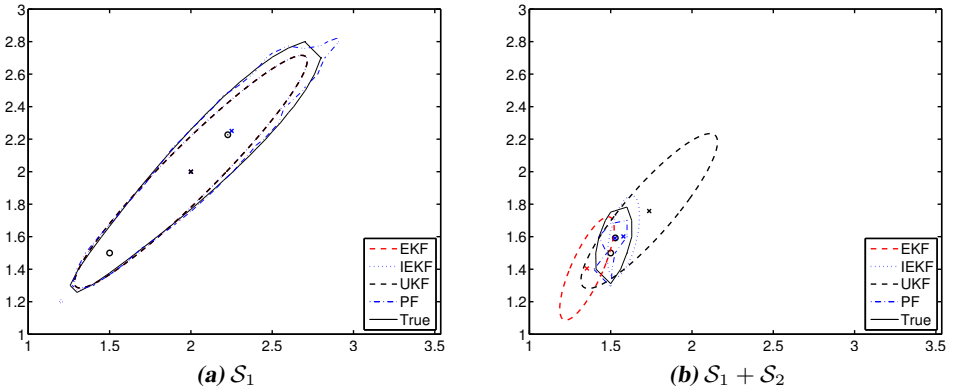
**Figure 1.2:** Illustration of the major components of a signal processing problem and how they relate to each other.

- the position of the target given in a suitable coordinate system, a filtering problem;
- an indication if there is an object present, a detection problem; or
- information about the type of the target, this however, is analysis on a level not treated in this thesis.

The estimated posterior probability distribution  $p(x_t|\mathbb{Y}_t)$ , which describes the target position, is represented in different ways with different filtering algorithms. The most common alternative is to deliver a position estimate and a covariance matrix describing the uncertainty in the estimate; these two are then usually combined into a Gaussian estimate of the position. The particle filter uses another approach and produces a cloud of particles representing the distribution. Other methods use a mixture of the two concepts. As seen in Figure 1.3 showing a distribution and its Gaussian approximation, information is lost by the approximation. An example of the different results obtained for the described bearings-only problem is given in Figure 1.4. To understand why the results differ, how they differ, and how to measure the difference is therefore important.



**Figure 1.3:** Illustration of the effect of approximating a PDF. Note how areas that are very unlikely in the true distribution have become very likely in the approximation. The approximation has the same mean and variance as the true distribution.



**Figure 1.4:** Example of estimation results from different filters trying to estimate the position of the target in Figure 1.1. The lines indicate the 47% confidence region.

## 1.2 Problem Formulation

Traditionally, estimation and change detection problems are usually treated using linear models affected by Gaussian noise. When this is not the case, linearization and noise approximations are in many cases used to create approximate linear and Gaussian systems. However, the effects on performance introduced in this way are often ignored, even though there exist methods to handle nonlinear non-Gaussian systems. There are many reasons for this, *e.g.*, the computational complexity increases, as well as the complexity of the algorithms themselves, and the design choices are different from the classical ones.

It would be ideal to have a framework to, from a complete system description, give guidelines for when classical approximations are appropriate and when other methods should be used. With such guidelines, the design effort can be put to use with methods appropriate for the specific problem at hand. Furthermore, the framework could also help in other design choices, *e.g.*, give rules of thumb to tune parameters and how to choose between different parametrizations and system descriptions.

To be able to provide this help to the user it is important to have a way to compare the performance between different alternatives. The most commonly used measure is the *mean square error* (MSE), which approximates  $E \|\hat{x} - x^0\|^2$  where  $\hat{x}$  is the estimate of the truth  $x^0$ . However, MSE is the same for both the posterior distributions in Figure 1.3, and a performance measure should indicate the difference between the two.

The thesis addresses these questions in several ways:

- Linear systems with non-Gaussian noise are analyzed with respect to the noise content to determine when nonlinear methods should be used.
- Approximations used to apply linear filters to nonlinear models are studied.
- Alternatives to the MSE are evaluated.

In addition to this, implementation issues have been studied because without being able to conduct large simulation studies it is difficult to evaluate nonlinear filtering.

## 1.3 Contributions

This section lists the contributions in this thesis and where they have been previously published.

The main focus of [53],

Gustaf Hendeby and Fredrik Gustafsson. Fundamental filtering limitations in linear non-Gaussian systems. In *Proceedings of 16th Triennial IFAC World Congress*, Prague, Czech Republic, July 2005,

is *intrinsic accuracy*, its effects on the *Cramér-Rao lower bound (CRLB)* for linear systems, and how this can be used to determine when nonlinear filters are potentially advantageous compared to the *Kalman filter*. The material in this paper makes up a large part of Chapter 2, and with extensions Chapter 6. One of the simulation studies in Chapter 7 is from this paper.

The material in [54],

Gustaf Hendeby and Fredrik Gustafsson. Fundamental fault detection limitations in linear non-Gaussian systems. In *Proceedings of 44th IEEE Conference on Decision and Control and European Control Conference*, Sevilla, Spain, December 2005,

and later in the extension [55],

Gustaf Hendeby and Fredrik Gustafsson. Detection limits for linear non-Gaussian state-space models. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Beijing, P. R. China, August–September 2006,

is an application of the intrinsic accuracy theory in [53] to fault detection. The material is found in Chapters 8 and 9, and gives guidelines for when non-Gaussian noise needs to be modeled when residuals are used for detection purposes.

The three publications [53], [54], and [55] do in an extended form make up the Licentiate's thesis [52],

Gustaf Hendeby. *Fundamental Estimation and Detection Limits in Linear Non-Gaussian Systems*. Licentiate thesis no 1199, Department of Electrical Engineering, Linköpings universitet, Sweden, November 2005.

The conference paper [57],

Gustaf Hendeby, Rickard Karlsson, Fredrik Gustafsson, and Neil Gordon. Recursive triangulation using bearings-only sensors. In *Proceedings of The IEE Seminar on Target Tracking: Algorithms and Applications*, Birmingham, UK, March 2006,

suggests using the *Kullback divergence* for comparing posterior distributions. The paper evaluates a bearings-only scenario with respect to different parameters and estimators. Results are provided that compare several filters using the mean square error in comparison with the Cramér-Rao lower bound, and using the Kullback divergence. The analysis of the *generalized Gaussian noise distribution* is presented in Chapter 2, and the simulation study is included in Chapter 7.

The work in [58],

Gustaf Hendeby, Rickard Karlsson, Fredrik Gustafsson, and Neil Gordon. Performance issues in non-Gaussian filtering problems. In *Proceedings of Nonlinear Statistical Signal Processing Workshop*, Cambridge, UK, September 2006,

continues the work in [57]. Compared to [57] it has a stronger focus on comparing posterior distributions given by different filters. It is shown, in several examples, that the MSE for two filters can be literary identical while the estimated posterior distributions differ substantially. The simulation studies appear in Chapter 7 and 9.

The C++ filtering framework presented in Chapter 10 was first presented in [56],

Gustaf Hendeby and Rickard Karlsson. Target tracking performance evaluation — a general software environment for filtering. In *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, March 2007.

The filtering framework is used to conduct Monte Carlo simulations that would otherwise have been difficult to manage in MATLAB®. The simulations once again stress the importance to use more than second order statistics to compare filter performance, and the Kullback divergence is shown to be a tool for this. The simulation studies are part of Chapter 7.

The second major contribution to Chapter 10 is [59],

Gustaf Hendeby, Jeroen D. Hol, Rickard Karlsson, and Fredrik Gustafsson. A graphics processing unit implementation of the particle filter. In *Proceedings of European Signal Processing Conference*, Poznań, Poland, September 2007.

This publication is, to the authors best knowledge, the first reported complete parallel particle filter implementation on a *graphics processing unit* (GPU). The implementation derived opens up for implementations on other types of parallel hardware.

The presentation of the *Rao-Blackwellized particle filter* (RBPF, sometimes referred to as the *marginalized particle filter*) in Chapter 5, and in part in Chapter 4, is a further refined version of the result in [60],

Gustaf Hendeby, Rickard Karlsson, and Fredrik Gustafsson. A new formulation of the Rao-Blackwellized particle filter. In *Proceedings of IEEE Workshop on Statistical Signal Processing*, Madison, WI, USA, August 2007.

The paper derives an alternative formulation, which can be interpreted as a stochastic Kalman filter bank. The new formulation is well suited for implementation in terms of standard components. The work with this was initiated by the need of a RBPF in the filtering framework described in Chapter 10.

The material in Chapter 3 is an extension of [47],

Fredrik Gustafsson and Gustaf Hendeby. On nonlinear transformations of stochastic variables and its application to nonlinear filtering. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Las Vegas, NV, USA, March 2008. Accepted for publication.

The paper studies the relationship between the first and second order *Gauss approximation* and the *unscented transform* to approximate the function of a distribution. It also shows how to formulate the Kalman filter in a way that the extended Kalman filter and the unscented Kalman filter become direct applications of the Gauss approximation and the unscented transform to nonlinear dynamics and measurement relations. This result is presented in Chapter 5.

## 1.4 Thesis Outline

This thesis is organized in the following way:

**Chapter 1 Introduction** (this chapter) Introduces the research problem and lists the contributions in the thesis.

**Chapter 2 Statistical Properties** Introduces the information measure *intrinsic accuracy* and *Kullback divergence* to compare distributions. These two are then used to characterize selected noise distributions for future reference.

**Chapter 3 Functions of Distributions** Treats functions of distributions, both analytical and approximate methods.

**Chapter 4 Models** Presents all model classes that are used in the thesis.

**Chapter 5 Filtering Methods** Surveys a number of filtering algorithms. The Rao-Blackwellized particle filter is presented in a way differing from what is usually seen in literature.

**Chapter 6 Cramér-Rao Lower Bound** Recapitulates the Cramér-Rao lower bound and derives performance measures, in terms of intrinsic accuracy, for linear systems that can be used as support to decide between different filtering methods for a given problem.

**Chapter 7 Filter Performance** Contains several simulation studies that verify the Cramér-Rao lower bound theory, and also illustrates the need for other evaluation criteria than second order statistics.

**Chapter 8 Change Detection** Focuses on a general change detection problem and derives approximate expressions in terms of intrinsic accuracy for determining whether non-Gaussian effects are important enough to be modeled.

**Chapter 9 Detection Performance** Contains several simulation studies to support the theory in Chapter 8.

**Chapter 10 Implementation Details** Presents two implementations of filtering systems. The first is a framework in C++ for efficient Monte Carlo simulations and close to application development. The second is a parallel particle filter implementation that has been implemented on a *graphic processing unit* (GPU).

**Chapter 11 Concluding Remarks** Concludes the thesis and gives ideas for future expansions

**Appendix A Notation** Lists the notation used in the thesis.

**Appendix B F++ Listings** Contains complete source-code listings in C++ for an example in Chapter 10.



# 2

---

## Statistical Properties

**T**O POSITION AN OBJECT using triangulation, as outlined in the Section 1.1, trivial application of geometry can be used given perfect measurements. However, nature is not that kind, in practice measurements are always inexact — and with noisy measurement the triangulation problem becomes nontrivial. To handle the imperfections, statistics is used. Statistics gives a mathematical way to describe and deal with randomness and phenomena difficult to explain in a structured way.

For this purpose, the available bearing measurements will be considered to be

$$y_i = \theta_i + e_i,$$

where  $y_i$  is a measurement of the angle  $\theta_i$  with some random error  $e_i$ . The noise  $e_i$  is often denoted measurement noise. Furthermore, in a similar way, if the object is not stationary but maneuvering in an unpredictable way, the maneuvers can be represented by process noise,  $w_i$ . The properties of the noise affecting a system are important for its behavior and also the ability to tell anything about system, *i.e.*, the position of the object in this case.

This chapter introduces the reader to the statistics needed for the signal processing applications in this thesis in Section 2.1. After preliminaries in statistics, Section 2.2 defines information measures used to determine how much information is available about a parameter in a distribution. This measure can then be compared to the information content in a Gaussian distribution. *Kullback divergence* are introduced in Section 2.3 to compare stochastic distributions. An important special case is to compare a distribution with its Gaussian approximation. Section 2.4 analyzes five different classes of distributions with respect to information and divergence.

### 2.1 Stochastic Variables

*Stochastic variables* are used in models of systems to describe unknown inputs, and its then often called *noise*. This section introduces noise from a statistical point of view and

serves as a foundation for the work in the rest of the thesis.

A stochastic variable is a variable without a specific value, that assumes different values with certain probabilities. To describe this behavior a *distribution function* or *cumulative distribution function* (CDF) is used. The CDF for the stochastic variable  $X$  is given by

$$P_X(x) = \Pr(X < x), \quad (2.1)$$

where  $\Pr(\mathcal{A})$  denotes the probability that the statement  $\mathcal{A}$  is true. Hence,  $P_X(x)$  denotes the probability that  $X$  is less than  $x$ . It follows from the definition of probability, that

$$\lim_{x \rightarrow -\infty} P_X(x) = 0, \quad \lim_{x \rightarrow +\infty} P_X(x) = 1,$$

and that  $P_X(x)$  is nondecreasing in  $x$ . Any function that fulfills these three conditions is a CDF and defines a statistical distribution. Another distribution description, more frequently used in this thesis, is the *probability density function* (PDF),

$$p_X(x) = \nabla_x P_X(x), \quad (2.2)$$

the gradient  $\nabla_x$  is defined in Appendix A, which describes the likelihood of certain  $X$  values, i.e.,

$$\Pr(x \in \mathcal{S}) = \int_{\mathcal{S}} p_X(x) dx.$$

Discrete stochastic variables are defined in a similar way. When the behavior of  $X$  is conditioned on another variable  $Y$  this is indicated by  $X|Y$ ,

$$p_{X|Y}(x|y) = \frac{p_{X,Y}(x,y)}{p_Y(y)}, \quad (2.3)$$

where  $p_{X|Y}(x|y)$  is the conditional PDF of  $x$  given the information  $y$ .

The CDF and the PDF both contain a complete description of the underlying stochastic variable. However, the following properties are often studied to get a better understanding of the behavior of the variable:

**Expected value** — the average value of several samples from the distribution,

$$\mu_X = \mathbb{E}(X) = \int xp_X(x) dx. \quad (2.4a)$$

When needed, an index will be used on  $\mathbb{E}$  to clarify which distribution to use in the integral, e.g.,  $\mathbb{E}_x$  or  $\mathbb{E}_{p_X}$  depending on the circumstances.

In some situations it is necessary to compute the expected value of a function of a stochastic variable. It is done using the integral

$$\mathbb{E}(f(X)) = \int f(x)p_X(x) dx.$$

**Variance or covariance matrix** for multidimensional distributions — indicates how close to the mean a sample can be expected to be,

$$\Sigma_X = \text{cov}(X) = \text{E} \left( (X - \mu_X)(X - \mu_X)^T \right), \quad (2.4b)$$

or  $\Sigma_X = \text{var}(x)$  for the scalar case. The same index system will be used as for expected values.

**Skewness** has no trivial extension to multidimensional distributions — indicates if samples are expected to have a symmetric behavior around the mean,

$$\gamma_{1,X} = \frac{\text{E}(X^3)}{\Sigma_X^{\frac{3}{2}}}. \quad (2.4c)$$

**Kurtosis** has no trivial extensions to multidimensional distributions — gives information about how heavy tails the distribution has,

$$\gamma_{2,X} = \frac{\text{E}(X^4)}{\Sigma_X^2} - 3. \quad (2.4d)$$

Traditionally, and still in Fisher based statistics, uppercase variables are used to denote stochastic variables, and their lowercase counterparts are used for realizations of them. However, this thesis will from here on use lower case variables in both cases, as in the Bayesian framework unless there is a risk of confusion.

## 2.2 Information in Distributions

Samples from a stochastic variable can be more or less informative about underlying parameters of the distribution. A measure of just how much information can be extracted about a parameter is given by the *Fisher information* (FI). As a special case of the Fisher information, the *intrinsic accuracy* (IA) measures the information available from samples from the distribution to determine the expected value. Fisher information and intrinsic accuracy are described in this section, and the relative measure *relative accuracy* (RA) is defined.

### 2.2.1 Fisher Information

The information concept used here dates back to the mid-1920's when Fisher in [34] investigated how much information is available about a parameter from a large set of samples. He elaborated further on this in [35]. Today, the result of his investigation is known as *Fisher information*.

**Definition 2.1 (Fisher Information).** The *Fisher information* (FI) with respect to the parameter  $\theta$  is defined, [89], as

$$\mathcal{I}_x(\theta) := -\text{E}_x \left( \Delta_\theta^\theta \log p(x|\theta) \right),$$

where  $p(x|\theta)$  is a PDF with the parameter  $\theta$  and the Hessian  $\Delta_\theta^\theta$  is defined in Appendix A, under the technical regularity conditions:

- (i) The set of possible  $\theta$  is an open set.
- (ii) The support of  $p(x|\theta)$  is independent of  $\theta$ .
- (iii) The partial derivative  $\nabla_{\theta} p(x|\theta)$  exists and is finite.
- (iv) The identity  $\mathbb{E}_x \nabla_{\theta} \log p(x|\theta) = 0$  holds.
- (v) The Fisher information is positive definite,  $\mathcal{I}_x(\theta) \succ 0$ .

The Fisher information for multidimensional parameters is sometimes called the *Fisher information matrix* to more clearly indicate that it is a matrix, however in this thesis Fisher information is used to denote both.

**Lemma 2.1**

*The Fisher information can be computed using the alternative expression*

$$\mathcal{I}_x(\theta) = \mathbb{E}_x \left( \nabla_{\theta} \log p(x|\theta) \right) \left( \nabla_{\theta} \log p(x|\theta) \right)^T.$$

**Proof:** From the regularity conditions in Theorem (2.1) follows:

$$0 = \mathbb{E}_x \nabla_{\theta} \log p(x|\theta) = \int \nabla_{\theta} \log p(x|\theta) p(x|\theta) dx.$$

Take the gradient with respect to  $\theta$  of this and move the gradient inside the integral,

$$\begin{aligned} 0 &= \nabla_{\theta} \int \nabla_{\theta} \log p(x|\theta) p(x|\theta) dx \\ &= \int \Delta_{\theta}^{\theta} \log p(x|\theta) p(x|\theta) + \nabla_{\theta} \log p(x|\theta) \nabla_{\theta} p(x|\theta) dx \\ &= \mathbb{E}_x \Delta_{\theta}^{\theta} \log p(x|\theta) + \mathbb{E}_x \left( \nabla_{\theta} \log p(x|\theta) \right) \left( \nabla_{\theta} p(x|\theta) \right)^T, \end{aligned}$$

which yields

$$\mathcal{I}_x(\theta) = -\mathbb{E}_x \Delta_{\theta}^{\theta} \log p(x|\theta) = \mathbb{E}_x \left( \nabla_{\theta} \log p(x|\theta) \right) \left( \nabla_{\theta} p(x|\theta) \right)^T.$$

□

The inverse of the Fisher information gives a lower bound on the variance of any unbiased estimate,  $\hat{\theta}(x)$ , of  $\theta$ , where  $\theta$  is a parameter to be estimated from the measurements  $x$ . More precisely [76, 89],

$$\text{cov}(\hat{\theta}(x)) \succeq \mathcal{I}_x^{-1}(\theta),$$

where  $A \succeq B$  denotes that the matrix  $A - B$  is positive semidefinite, i.e.,  $x^T(A - B)x \geq 0$  for all  $x$  of suitable dimension. This lower bound on the variance of an estimated parameter, introduced by Rao [113], is often referred to as the *Cramér-Rao lower bound* (CRLB) [76],

$$P_{\theta}^{\text{CRLB}} = \mathcal{I}_x^{-1}(\theta).$$

The CRLB will, when extended to handle dynamic systems in Chapter 6, play an important role in this thesis. For now, just note the connection between the Fisher information and the CRLB.

### 2.2.2 Intrinsic Accuracy

The Fisher information with respect to the location, the expected value  $\mu$ , of a stochastic variable is used frequently in this thesis, therefore, introduce the short hand notation  $\mathcal{I}_x := \mathcal{I}_x(\mu)$ . This quantity is in [27, 77, 78] referred to as the *intrinsic accuracy* (IA) of the PDF for  $x$ . This name is inspired by the terminology used in Fisher [35].

The reference [63] provides a formulation of Fisher information that relaxes the regularity conditions for intrinsic accuracy compared to Definition 2.1.

**Definition 2.2 (Intrinsic Accuracy).** Denote the Fisher information with respect to the location of the distribution *intrinsic accuracy* (IA). The intrinsic accuracy is then given by

$$\mathcal{I}_x = \int \left( \frac{\nabla_{\mu} p(x|\mu)}{p(x|\mu)} \right) \left( \frac{\nabla_{\mu} p(x|\mu)}{p(x|\mu)} \right)^T p(x|\mu) dx,$$

with the relaxed regularity conditions:

- (i) The function  $p$  is an absolutely continuous PDF.
- (ii) The integral  $\mathcal{I}_x$  is bounded.

If the regularity conditions are not fulfilled, then  $\mathcal{I}_\mu = +\infty$ .

Note the similarity with the alternative expression for the Fisher information given by Lemma 2.1 since  $\nabla_{\theta} \log p(x|\theta) = \nabla_{\theta} p(x|\theta)/p(x|\theta)$ . Observe however that the relaxed regularity conditions are valid only for information about the location.

When estimating the mean of a stochastic variable it is always possible to achieve an unbiased estimator with the same covariance as the stochastic variable, and in some cases it is possible to get an even better estimator. In fact, for non-Gaussian distributions the lower bound is always better, as stated in the following theorem.

#### Theorem 2.1

For the intrinsic accuracy and covariance of the stochastic variable  $x$ ,

$$\text{cov}(x) \succeq \mathcal{I}_x^{-1},$$

with equality if and only if  $x$  is Gaussian.

**Proof:** See [128]. □

In this respect the Gaussian distribution is a worst case distribution. Of all distributions with the same covariance the Gaussian distribution is the distribution with the least information available about its average. All other distributions have larger intrinsic accuracy.

### 2.2.3 Relative Accuracy

The relation between intrinsic accuracy and covariance is interesting in many situations. In most cases only the relative difference between the two matters, therefore introduce *relative accuracy* (RA).

**Definition 2.3 (Relative Accuracy).** A scalar constant  $\Psi_x$  such that

$$\text{cov}(x) \mathcal{I}_x = \Psi_x \cdot I$$

is denoted the *relative accuracy* (RA).

It follows from Theorem 2.1 that  $\Psi_x \geq 1$ , with equality if and only if  $x$  is Gaussian. The relative accuracy is thus a measure of how much useful information there is in the distribution compared to a Gaussian distribution with the same covariance. In this way relative accuracy can be used to determine, in this respect, how similar to Gaussian a distribution is. If the relative accuracy is close to 1 it has similar information properties as a Gaussian distribution, where as if it is larger it differs substantially.

---

**Example 2.1: Accuracy of a Gaussian variable**

---

Let  $x \sim \mathcal{N}(\mu, \Sigma)$  be a scalar Gaussian stochastic variable (the Gaussian distribution will be treated in more detail in Section 2.4.1),

$$p(x|\mu) = \frac{1}{\sqrt{2\pi\Sigma}} e^{-\frac{(x-\mu)^2}{2\Sigma}},$$

and calculate intrinsic accuracy and relative accuracy:

$$\nabla_{\mu} \log p(x|\mu) = \nabla_{\mu} \left( -\frac{\log(2\pi\Sigma)}{2} - \frac{(x-\mu)^2}{2\Sigma} \right) = 0 - \frac{x-\mu}{\Sigma}$$

and

$$\mathcal{I}_x = \mathbb{E}_x \left( -\frac{x-\mu}{\Sigma} \right)^2 = \frac{1}{\Sigma}.$$

The relative accuracy follows directly,

$$\Psi_x = \text{var}(x) \mathcal{I}_x = \Sigma/\Sigma = 1.$$


---

## 2.2.4 Accuracy Properties

The intrinsic accuracy for independent variables is separable. This is intuitive and can be used to simplify calculations considerably.

### Lemma 2.2

For a vector  $\mathbb{X}$  of independently distributed stochastic variables  $\mathbb{X} = (x_1^T, \dots, x_n^T)^T$  each with covariance  $\text{cov}(x_i) = \Sigma_{x_i}$  and intrinsic accuracy  $\mathcal{I}_{x_i}$ , for  $i = 1, \dots, n$ ,

$$\text{cov}(\mathbb{X}) = \text{diag}(\Sigma_{x_1}, \dots, \Sigma_{x_n}) \quad \text{and} \quad \mathcal{I}_{\mathbb{X}} = \text{diag}(\mathcal{I}_{x_1}, \dots, \mathcal{I}_{x_n}).$$

If  $\Sigma_{x_i} = \Sigma_x$  and  $\mathcal{I}_{x_i} = \mathcal{I}_x$  then the covariance and the intrinsic accuracy are more compactly expressed

$$\text{cov}(\mathbb{X}) = I \otimes \Sigma_x \quad \text{and} \quad \mathcal{I}_{\mathbb{X}} = I \otimes \mathcal{I}_x,$$

where  $\otimes$  denotes the Kronecker product. Furthermore, if  $\Sigma_x \mathcal{I}_x = \Psi_x \cdot I$ , with  $\Psi_x \geq 1$  a scalar, then

$$\text{cov}(\mathbb{X}) = \Psi_x \mathcal{I}_x^{-1} = \Psi_x I \otimes \mathcal{I}_x^{-1}.$$

**Proof:** For  $\mathbb{X} = (x_1^T, x_2^T, \dots, x_n^T)^T$ , with  $x_i$  independently distributed, it follows immediately that

$$\text{cov}(\mathbb{X}) = \text{diag}(\text{cov}(x_1), \dots, \text{cov}(x_n)) = \text{diag}(\Sigma_{x_1}, \dots, \Sigma_{x_n}).$$

Expand the expression:

$$\mathcal{I}_x = -\mathbb{E}(\Delta_{\mathbb{X}}^{\mathbb{X}} \log p(\mathbb{X})) = -\mathbb{E}\left(\Delta_{\mathbb{X}}^{\mathbb{X}} \log \prod_{i=1}^n p(x_i)\right) = \sum_{i=1}^n -\mathbb{E}(\Delta_{\mathbb{X}}^{\mathbb{X}} \log p(x_i)).$$

The partial derivatives of this expression become, for  $k = l = i$ ,

$$-\mathbb{E}(\nabla_{x_k} \nabla_{x_l} \log p(x_i)) = -\mathbb{E}(\Delta_{x_i}^{x_i} \log p(x_i)) = \mathcal{I}_{x_i},$$

and for  $k \neq l$  the partial derivatives vanish,  $-\mathbb{E}(\nabla_{x_k} \nabla_{x_l} \log p(x_i)) = 0$ . Combining these results using matrix notation yields

$$\mathcal{I}_x = \text{diag}(\mathcal{I}_{x_1}, \dots, \mathcal{I}_{x_n}).$$

The compact notation for  $\Sigma_{x_i} = \Sigma_x$  and  $\mathcal{I}_{x_i} = \mathcal{I}_x$  follows as in the proof of Theorem 2.1.  $\square$

Intrinsic accuracy of linear combinations of stochastic variables can be calculated from the intrinsic accuracy of the components.

### Theorem 2.2

For the linear combination of stochastic variables  $\mathbb{Z} = B\mathbb{X}$ , where  $\mathbb{X} = (x_1^T, \dots, x_n^T)^T$  is a stochastic variable with independent components with covariance  $\text{cov}(x_i) = \Sigma_{x_i}$  and intrinsic accuracy  $\mathcal{I}_{x_i}$ ,

$$\begin{aligned} \text{cov}(\mathbb{Z}) &= B \text{diag}(\Sigma_{x_1}, \dots, \Sigma_{x_n}) B^T, \\ \mathcal{I}_{\mathbb{Z}}^{-1} &= B \text{diag}(\mathcal{I}_{x_1}^{-1}, \dots, \mathcal{I}_{x_n}^{-1}) B^T. \end{aligned}$$

**Proof:** Combine the result found as Theorem 4.3 in [15] with Lemma 2.2.  $\square$

If the combined variables are *identically and independently distributed* (IID) the expressions can be further simplified using the Kronecker product formulation given in Lemma 2.2. It is also worth noticing that if  $B$  does not have full row rank the intrinsic accuracy is infinite in some direction and hence no bounded  $\mathcal{I}_{\mathbb{Z}}$  exists.

## 2.3 Kullback-Leibler Information

The *Kullback-Leibler information* [85, 86], also called the *discriminating information*, quantifies the difference between two distributions. The Kullback-Leibler information is not symmetric in its arguments, and therefore not a measure. If a symmetric quantity is needed, the *Kullback divergence*, constructed as a symmetric sum of two Kullback-Leibler information [7, 85], can be used as an alternative.

**Definition 2.4 (Kullback-Leibler Information).** The *Kullback-Leibler information* is defined, for the two proper PDFs  $p$  and  $q$ , as

$$\mathcal{I}^{\text{KL}}(p, q) = \mathbb{E}_p \log \frac{p(x)}{q(x)},$$

when  $p(x) \neq 0 \Rightarrow q(x) \neq 0$  and otherwise as  $\mathcal{I}^{\text{KL}}(p, q) = +\infty$ .

**Definition 2.5 (Kullback Divergence).** The *Kullback divergence* is defined in terms of Kullback-Leibler information as

$$\mathcal{J}^{\text{K}}(p, q) = \mathcal{I}^{\text{KL}}(p, q) + \mathcal{I}^{\text{KL}}(q, p).$$

The Kullback-Leibler information is closely related to other statistical measures, e.g., *Shannon's information* and *Akaike's information criterion* [7].

Neither Kullback-Leibler information nor Kullback divergence are a measure in the mathematical sense. Both have the property that  $\mathcal{I}^{\text{KL}}(p, q) \geq 0$  with equality if and only if  $p = q$  almost everywhere [85]. However, the Kullback-Leibler information is unsymmetric in its arguments, and the Kullback divergence does not fulfill the triangle inequality.

Both the Kullback-Leibler information and the Kullback divergence are additive for independent stochastic variables as shown in Lemma 2.3.

### Lemma 2.3

For a vector  $\mathbb{X}$  of independent stochastic variables,  $\mathbb{X} = (x_1^T, \dots, x_n^T)^T$  distributed with PDF  $p(\mathbb{X}) = \prod_{i=1}^n p_i(x_i)$  or  $q(\mathbb{X}) = \prod_{i=1}^n q_i(x_i)$  the *Kullback-Leibler information* and *Kullback divergence* are additive, i.e.,

$$\mathcal{I}^{\text{KL}}(p, q) = \sum_{i=1}^n \mathcal{I}^{\text{KL}}(p_i, q_i)$$

and

$$\mathcal{J}^{\text{K}}(p, q) = \sum_{i=1}^n \mathcal{J}^{\text{K}}(p_i, q_i).$$

**Proof:** If  $\mathbb{X}$ ,  $p$ , and  $q$  satisfy the conditions in the lemma, then

$$\begin{aligned} \mathcal{I}^{\text{KL}}(p, q) &= \mathbb{E}_p \log \frac{p(\mathbb{X})}{q(\mathbb{X})} \\ &= \int \prod_{j=1}^n p_j(x_j) \log \frac{\prod_{i=1}^n p_i(x_i)}{\prod_{i=1}^n q_i(x_i)} d\mathbb{X} = \sum_{i=1}^n \int \prod_{j=1}^n p_j(x_j) \log \frac{p_i(x_i)}{q_i(x_i)} d\mathbb{X} \\ &= \sum_{i=1}^n \int p_i(x_i) \log \frac{p_i(x_i)}{q_i(x_i)} dx_i = \sum_{i=1}^n \mathcal{I}^{\text{KL}}(p_i, q_i), \end{aligned}$$

where the second last equality is a *marginalization* utilizing that  $p_i$ , for all  $i$ , are proper PDFs and hence integrate to unity.

The Kullback divergence result follows immediately,

$$\begin{aligned} \mathcal{J}^K(p, q) &= \mathcal{I}^{\text{KL}}(p, q) + \mathcal{I}^{\text{KL}}(q, p) \\ &= \sum_{i=1}^n \mathcal{I}^{\text{KL}}(p_i, q_i) + \sum_{i=1}^n \mathcal{I}^{\text{KL}}(q_i, p_i) \\ &= \sum_{i=1}^n (\mathcal{I}^{\text{KL}}(p_i, q_i) + \mathcal{I}^{\text{KL}}(q_i, p_i)) = \sum_{i=1}^n \mathcal{J}^K(p_i, q_i). \end{aligned}$$

□

For small differences between distributions there exists an approximation of the Kullback-Leibler information and Kullback divergence in terms of Fisher information, [85]. This provides an interesting connection between the two quite conceptually different quantities. The approximation is valid up to second order terms and relates the difference between  $p(x; \theta)$  and  $p(x; \theta + \theta_\Delta)$  for small values of  $\theta_\Delta$ . Under weak conditions,

$$\mathcal{I}^{\text{KL}}(p(\cdot; \theta), p(\cdot; \theta + \theta_\Delta)) \approx \frac{1}{2} \theta_\Delta^T \mathcal{I}_{p(\cdot; \theta)}(\theta) \theta_\Delta \quad (2.5)$$

$$\mathcal{J}^K(p(\cdot; \theta), p(\cdot; \theta + \theta_\Delta)) \approx \theta_\Delta^T \mathcal{I}_{p(\cdot; \theta)}(\theta) \theta_\Delta. \quad (2.6)$$

There exist bounds on both the Kullback-Leibler information and the Kullback divergence in terms of measures that are easier to compute. The bound in Theorem 2.3 is one of the sharpest lower bounds known [93].

### Theorem 2.3

*A lower bound for the Kullback-Leibler information is defined in terms of the variational distance,  $\|\cdot\|_1$ , between the two PDFs  $p$  and  $q$ ,*

$$\|p - q\|_1 = \int |p(x) - q(x)| dx,$$

and is

$$\begin{aligned} \mathcal{I}^{\text{KL}}(p, q) &\geq \max \{L_1(\|p - q\|_1), L_2(\|p - q\|_1)\} \\ L_1(\|p - q\|_1) &= \log \frac{2 + \|p - q\|_1}{2 - \|p - q\|_1} - \frac{2\|p - q\|_1}{2 - \|p - q\|_1} \\ L_2(\|p - q\|_1) &= \frac{\|p - q\|_1^2}{2} + \frac{\|p - q\|_1^4}{36} + \frac{\|p - q\|_1^6}{288}, \end{aligned}$$

where by definition  $0 \leq \|p - q\|_1 \leq 2$ .

**Proof:** See [93].

□

Unfortunately there seems to be few useful upper bounds for the Kullback-Leibler information and the Kullback divergence.

## 2.4 Analysis of Selected Distributions

In this section, the *Gaussian distribution*, one of the most widely used distributions, is presented and properties for it are derived. Furthermore, the class of *Gaussian sum distributions*, the *generalized Gaussian distribution*, and the *inverse normal Gaussian distribution* are introduced as complements to the Gaussian distribution that can be used to explain more complex phenomena.

### 2.4.1 Gaussian Distribution

The most widely used stochastic distribution is the *Gaussian distribution*, or *Normal distribution*, denoted with  $\mathcal{N}(\mu, \Sigma)$ , where  $\mu$  and  $\Sigma \succ 0$  are parameters representing expected value and variance (covariance matrix) of the distribution, respectively. The Gaussian distribution is for a scalar stochastic variable defined in terms of its PDF,

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{\sqrt{2\pi\Sigma}} e^{-\frac{(x-\mu)^2}{2\Sigma}}, \quad (2.7)$$

which extends to a vector valued stochastic variable as in Definition 2.6.

**Definition 2.6 (Gaussian distribution).** The Gaussian distribution is defined by its PDF

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^{n_x} \det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)},$$

where  $n_x = \dim(x)$  and  $\Sigma$  is a positive definite  $n_x \times n_x$  matrix.

The Gaussian CDF,

$$\Phi(x; \mu, \Sigma) := \int_{\xi < x} \mathcal{N}(\xi; \mu, \Sigma) d\xi,$$

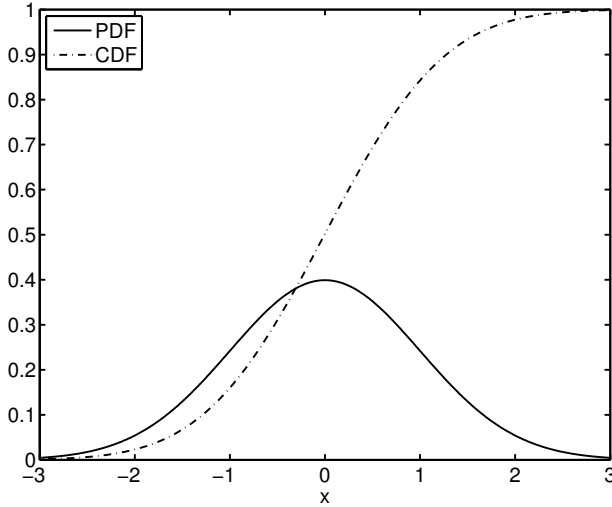
lacks an analytic expression, however, it is tabulated in most collections of statistical tables. Figure 2.1 shows the PDF and CDF of the normalized Gaussian distribution,  $\mathcal{N}(0, 1)$ .

The widespread usage of the Gaussian distribution is in part motivated by the fact that many natural phenomena exhibit Gaussian or Gaussian-like properties. A reason for this is that the sum of stochastic variables, under weak conditions by the *central limit theorem*, becomes Gaussian as the number of terms increases [108]. Hence, if a natural phenomenon is a combination of many stochastic phenomena it is often quite reasonable to assume that the combination of them is Gaussian.

The Gaussian distribution has favorable mathematical properties, and it is possible to derive many analytical results when the Gaussian distribution is used. One reason for this is that the Gaussian distribution is its own *conjugate prior*<sup>1</sup>, [117]. Another useful such property is that any linear combination of Gaussian variables is Gaussian, *i.e.*, if  $x \sim \mathcal{N}(\mu, \Sigma)$  and  $B$  is a linear transformation of full (row) rank, then

$$z := Bx \sim \mathcal{N}(B\mu, B\Sigma B^T). \quad (2.8)$$

<sup>1</sup>A family distributions  $\pi(\theta)$  is a conjugated prior to  $p(x|\theta)$  if  $\pi(\theta|x)$  belongs to the same family of distributions as  $\pi(\theta)$ .



**Figure 2.1:** PDF and CDF for the normalized Gaussian distribution,  $\mathcal{N}(0, 1)$ .

If  $B$  is rank deficient the resulting stochastic variable represents a degenerate case where one or more of the elements can be obtained as a combination of the other.

Finally, calculating the properties discussed in Section 2.1, (2.4), for  $x \sim \mathcal{N}(\mu, \Sigma)$  yields  $E(x) = \mu$ ,  $\text{cov}(x) = \Sigma$ ,  $\gamma_1 = 0$ , and  $\gamma_2 = 0$ . This makes skewness and kurtosis quantities that can be used to indicate non-Gaussianity if they are nonzero. Furthermore, the intrinsic accuracy is  $\mathcal{I}_x = \Sigma^{-1}$  and subsequently Gaussian distributions have the relative accuracy  $\Psi_x = 1$ .

### 2.4.2 Gaussian Sum Distribution

Even though the Gaussian distribution is common it is not powerful enough to capture every stochastic phenomena in a satisfactory manner. One way to extend the Gaussian distribution is to mix several Gaussian distributions. The result is a *Gaussian mixture distribution* or *Gaussian sum distribution*.

**Definition 2.7 (Gaussian sum distribution).** The *Gaussian sum distribution* is defined by its PDF

$$\mathcal{N}_n(x; (\omega_\delta, \mu_\delta, \Sigma_\delta)_{\delta=1}^n) = \sum_{\delta=1}^n \omega_\delta \mathcal{N}(x; \mu_\delta, \Sigma_\delta),$$

where  $\omega_\delta > 0$ ,  $\sum_\delta \omega_\delta = 1$ , and  $n$  indicates how many Gaussian components, *modes*, are used.

**Note:** for  $n = 1$  the Gaussian sum reduces to a Gaussian, and that the notation for the Gaussian sum is ambiguous, e.g.,  $\mathcal{N}_2((\frac{1}{2}, 0, 1), (\frac{1}{2}, 0, 1)) = \mathcal{N}(0, 1)$ .

One interpretation of the Gaussian sum is that for all possible  $\delta$  the probability is  $\omega_\delta$  that a sample comes from the mode  $\delta$ , i.e., the Gaussian distribution  $\mathcal{N}(\mu_\delta, \Sigma_\delta)$ . Using a

Gaussian sum it is possible to approximate any distribution  $p$  such that

$$\hat{p}(x) = \sum_{i=1}^N \omega_i \mathcal{N}(x; \mu_i, \Sigma_i) \rightarrow p(x), \quad N \rightarrow +\infty,$$

with uniform convergence in the norm  $\|p - \hat{p}\|_1 = \int |p(x) - \hat{p}(x)| dx$  under weak regularity conditions, [3, 4, 132].

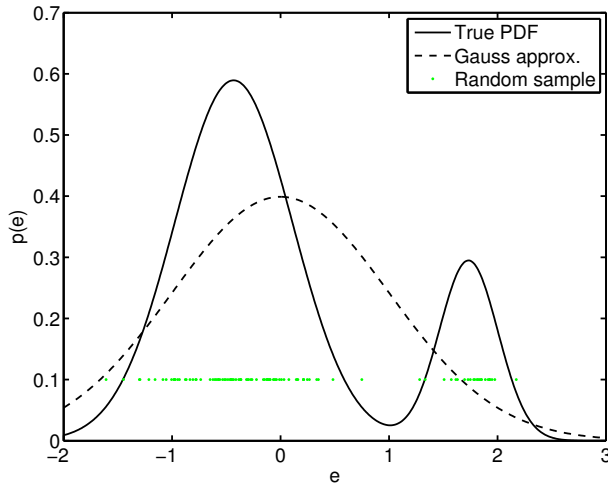
The Gaussian sum is its own conjugate prior [117]. However, generally the number of modes in the distribution increases exponentially making it difficult to use this property without some approximation to reduce the number of modes.

### Example 2.2: Bi-Gaussian noise

Radar measurements often exhibit bimodal properties due to secondary radar reflections. The Master's theses [28, 135] both study this phenomenon for radar altitude measurements from an aircraft. The collected data clearly shows that measurements over certain terrains results in bimodal measurement errors. Radar reflections in treetops is one reason for this, *e.g.*, when flying over forests. This noise,  $e$ , is well modeled using a Gaussian sum with two components, a *bi-Gaussian* distribution, similar to

$$\begin{aligned} e &\sim \mathcal{N}_2((0.8, -0.43, 0.29), (0.2, 1.73, 0.07)) \\ &= 0.8\mathcal{N}(-0.43, 0.29) + 0.2\mathcal{N}(1.73, 0.07). \end{aligned} \quad (2.9)$$

The PDF of this distribution is depicted in Figure 2.2. Note how poorly the Gaussian approximation captures the true properties of the distribution. Hence, using knowledge of non-Gaussian noise characteristics can be favorable, as shown in *e.g.*, [14–16].



**Figure 2.2:** The bimodal bi-Gaussian distribution in (2.9) and a second order equivalent Gaussian distribution. The distribution is representative of what can be found as measurement noise in radar measurements.

The mean of a Gaussian mixture is obtained by a weighted sum of the means of the combined Gaussian modes

$$\mu = \sum_{\delta} \omega_{\delta} \mu_{\delta}. \quad (2.10a)$$

The covariance matrix is a weighted sum of the covariances of the different modes and spread of the mean terms

$$\Sigma = \sum_{\delta} \omega_{\delta} (\Sigma_{\delta} + \bar{\mu}_{\delta} \bar{\mu}_{\delta}^T), \quad (2.10b)$$

where  $\bar{\mu}_{\delta} := \mu_{\delta} - \mu$ . The higher order moments, assuming a scalar distribution, are

$$\gamma_1 = \sum_{\delta} \omega_{\delta} \bar{\mu}_{\delta} (3\Sigma_{\delta} + \bar{\mu}_{\delta}^2) \Sigma^{-\frac{3}{2}} \quad (2.10c)$$

$$\gamma_2 = \sum_{\delta} \omega_{\delta} (3\Sigma_{\delta}^2 + 6\bar{\mu}_{\delta}^2 \Sigma_{\delta} + \bar{\mu}_{\delta}^4) \Sigma^{-2} - 3. \quad (2.10d)$$

Compare these values to the skewness and kurtosis,  $\gamma_1 = \gamma_2 = 0$ , obtained for Gaussian distributions.

Calculating the intrinsic accuracy for a Gaussian mixture distribution is more difficult, and in general no analytic expression exists. However, Monte Carlo integration (discussed further in Section 3.2) provides means to approximate it with sufficient accuracy.

To conclude the presentation of the Gaussian mixtures, three examples of distributions with  $E(x) = 0$  and  $\text{var}(x) = 1$  with two free parameters will be given. The distributions represent noise that can be used to describe non-Gaussian noise that can be encountered. For each distribution is the intrinsic accuracy and the Kullback divergence compared to a normalized Gaussian computed.

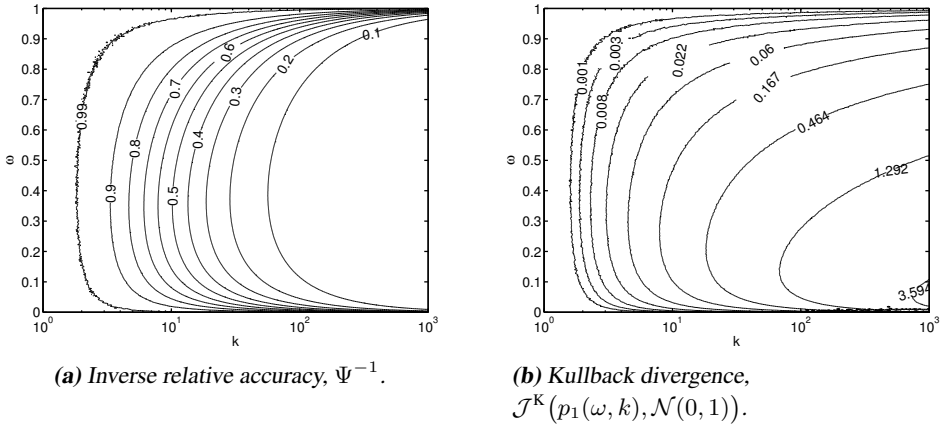
### Outliers Distribution

An example of a Gaussian sum that will be used in this thesis to illustrate theoretical results is defined in terms of the PDF

$$p_1(x; \omega, k) = (1 - \omega) \mathcal{N}(x; 0, \Sigma) + \omega \mathcal{N}(x; 0, k\Sigma), \quad (2.11)$$

where  $\Sigma = 1/(1 + (k - 1)\omega)$ ,  $0 \leq \omega \leq 1$ , and  $k > 0$  to obtain proper distributions. This distribution can be used to model outliers. The two modes of the distribution represent nominal behavior and outlier behavior, respectively. With this interpretation outliers occur with probability  $\omega$  and have  $k$  times the nominal variance. Hence, if  $x \sim p_1(0.1, 10)$  then 10% of the samples are expected to be outliers and to have 10 times the variance of the nominal samples.

The parameterization is intentionally chosen in such a way that  $\mu = 0$  and  $\Sigma = 1$  to allow for straightforward comparison with the normalized Gaussian distribution,  $\mathcal{N}(0, 1)$ . As with Gaussian distributions, moving the mean and changing the variance is a matter of changing the mean and scaling the variance of the different modes.



**Figure 2.3:** Inverse relative accuracy and Kullback divergence for the outliers description (2.11).

The relative accuracy for the parameterization (2.11) is given in Figure 2.3(a). Studying the contour plot shows that in an information perspective most information is available if about 30–40% of the samples are outliers, and the outliers have substantially larger variance, *i.e.*,  $\omega \approx 0.35$  and  $k \gg 1$ .

The Kullback divergence between the outliers distributions,  $p_1(\omega, k)$ , and the normalized Gaussian distribution  $\mathcal{N}(0, 1)$ , has been computed to illustrate the difference between the distributions. The result is found in Figure 2.3(b). The relative accuracy and the Kullback divergence behave similarly.

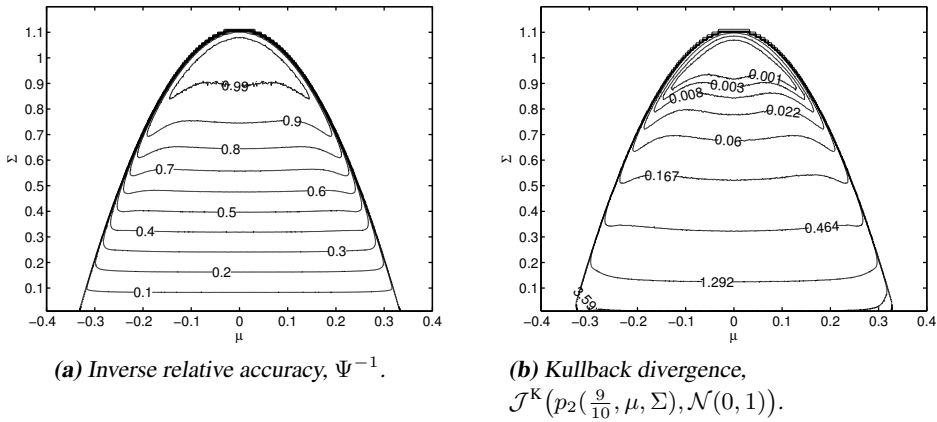
### Unsymmetric Bimodal Distribution

Another distribution that will be used throughout this thesis is the bimodal distribution given by the PDF

$$p_2(x; \omega, \mu, \Sigma) = \omega \mathcal{N}(x; \mu, \Sigma) + (1 - \omega) \mathcal{N}\left(x; \frac{-\omega\mu}{1-\omega}, \frac{1-\omega\mu^2/(1-\omega)-\omega\Sigma}{1-\omega}\right) \quad (2.12)$$

where  $0 < \Sigma < \frac{1}{\omega} - \frac{\mu^2}{1-\omega}$  and  $0 < \omega < 1$  to get a proper distribution. This is a distribution of the same type as was used in Example 2.2. As described there, this type of distribution can be used to model radar measurements, where one of the modes is the result of secondary radar reflections, *e.g.*, in treetops [15, 28, 135].

The intrinsic accuracy of the parameterized distributions and the Kullback divergence compared to a normalized Gauss distribution are found in Figure 2.4, for  $\omega = \frac{9}{10}$ . The two measures behave very similarly. Close to the boundary of the allowed parameter region, the two modes both approach point distributions since the spread of the mean will contribute substantially to the total variance of the distribution. With point distributions present the information content is very high, since once the point distribution is identified



**Figure 2.4:** Inverse relative accuracy and Kullback divergence for the bimodal distribution (2.12) with  $\omega = \frac{9}{10}$ .

the estimate will be correct. A similar argumentation can be made about the Kullback information and its interpretation as a measure of how difficult it is to distinguish between two distributions.

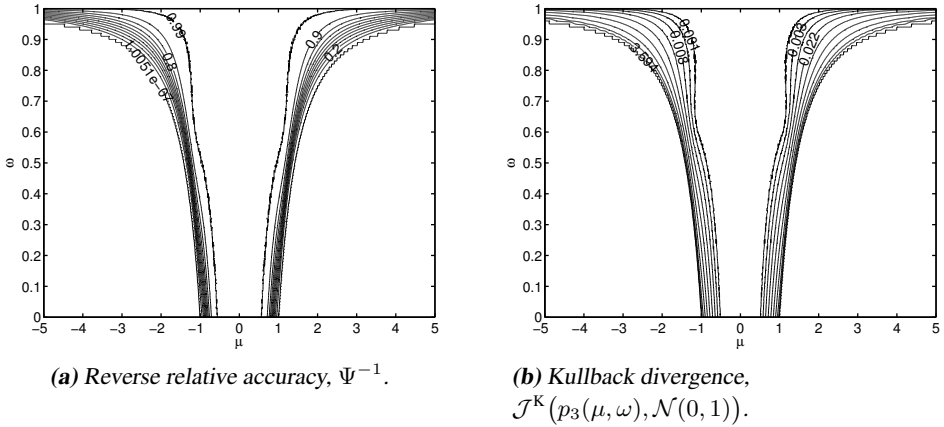
### Symmetric Trimodal Distribution

The final distribution is defined by the PDF

$$p_3(x; \mu, \omega) = \frac{1-\omega}{2} \mathcal{N}(x; -\mu, \Sigma) + \omega \mathcal{N}(x; 0, \Sigma) + \frac{1-\omega}{2} \mathcal{N}(x; +\mu, \Sigma), \quad (2.13)$$

where  $\Sigma = 1 - \mu^2(1 - \omega)$  and  $1 - \mu^{-2} < \omega < 1$  to get a proper distribution. In many respects this distribution is similar to the bimodal distribution (2.12). Generally, multimodal distributions can be used to model noise with several modes, where the different means indicate the different modes and the variance some uncertainty in them. Consider an aircraft where negative input represent turning left, positive input a right turn, and no input going straight. Then having a process noise with a negative mode, a positive mode, and a mode in zero, would indicate that the aircraft is either turning left, right, or not at all.

The relative accuracy for (2.13) is found in Figure 2.5(a) and the Kullback divergence when the distribution is compared to a normalized Gaussian is in Figure 2.5(b). Once gain, the principal behavior of the relative accuracy and the Kullback divergence is similar. As the modes are separated, the information increases up until the point where the distribution consists of three point distributions.



**Figure 2.5:** Inverse relative accuracy and Kullback divergence for the trimodal distribution (2.13).

### 2.4.3 Generalized Gaussian Distribution

The *generalized Gaussian distribution* [24] is a (scalar) unimodal distribution with the Gaussian distribution as a special case. It covers both distributions with less weight in the tails as well as distributions with heavier tails than the Gaussian distribution. The distribution is used in image and video encoding [2, 102] and to handle heavy tailed measurement noise and correlated signal sources [103]. The generalized Gaussian distribution is also shown to be the worst case distribution when it comes to CRLB-like bounds for the  $p^{\text{th}}$  moment [12] under certain conditions.

**Definition 2.8 (Generalized Gaussian distribution).** The *generalized Gaussian* PDF is given by

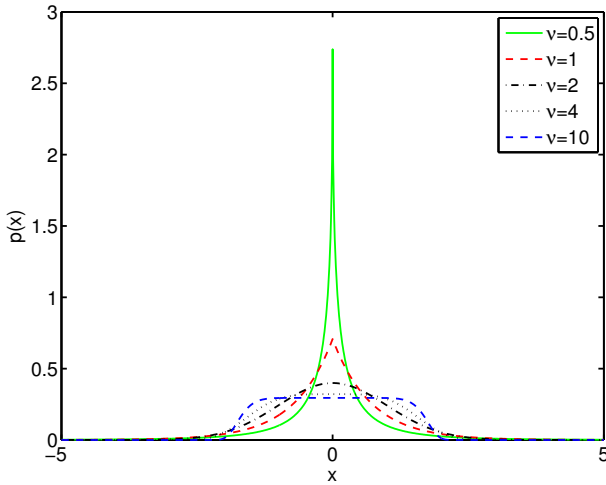
$$\mathcal{N}^\nu(x; \sigma) = \frac{\nu \eta(\nu, \sigma)}{2\Gamma(\nu^{-1})} e^{-(\eta(\nu, \sigma)|x|)^\nu}, \quad (2.14)$$

with

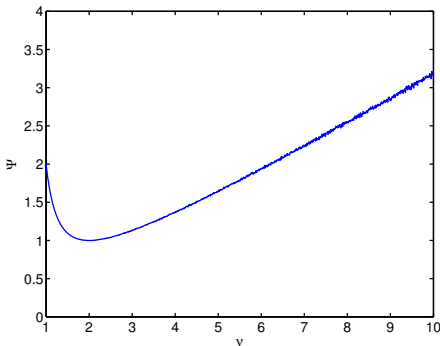
$$\eta(\nu, \sigma) = \sigma^{-1} \sqrt{\Gamma(3\nu^{-1})/\Gamma(\nu^{-1})},$$

and  $\Gamma$  is the *Gamma function* [24].

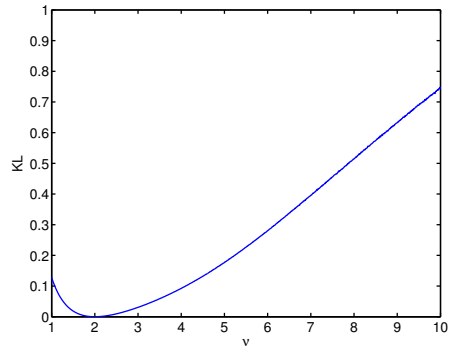
The parameter  $\nu$  in the generalized Gaussian distribution acts as a shape parameter and  $\sigma^2$  is the variance of the distribution. For  $\nu \rightarrow 0^+$  the generalized Gaussian is a *Dirac distribution* and for  $\nu \rightarrow +\infty$  a uniform distribution,  $\mathcal{U}(-\sqrt{3}\sigma, \sqrt{3}\sigma)$ , both with variance  $\sigma^2$  and infinite intrinsic accuracy. (The intrinsic accuracy is infinite because the Dirac distribution is a point distribution and the support of the uniform distribution depends on the mean of it.) Some examples of different  $\nu$  are illustrated in Figure 2.6. Two important special cases  $\nu = 1$  and  $\nu = 2$  correspond to the *Laplacian* and the *Gaussian* distribution, respectively. The mean of the distribution is  $\mu$ . The relative accuracy of the distribution and the Kullback divergence compared to a normalized Gaussian are given in Figure 2.7.



**Figure 2.6:** PDF of the generalized Gaussian distribution as a function of the shape parameter  $\nu$ . ( $\nu = 1$  Laplace distribution,  $\nu = 2$  Gaussian distribution, and  $\nu = +\infty$  uniform distribution.)



**(a)** Relative accuracy,  $\Psi$ .



**(b)** Kullback divergence,  $\mathcal{J}^K(\mathcal{N}^\nu(1), \mathcal{N}(0, 1))$ .

**Figure 2.7:** Relative accuracy and Kullback divergence, of a generalized Gaussian distribution (2.14), parameterized in  $\nu$ .

## 2.4.4 Normal Inverse Gauss

The *normal inverse Gaussian distribution* is an example of a normal variance-mean mixture distribution, [51]. Such distributions do not assume that the variance is fixed, but allows it to be related to the mean. This gives a distribution with good properties for modeling heavy-tailed distributions [51]. In [107] the distribution is used to model data from hydrophones, synthetic aperture sonars, and cluttered sea radars. To achieve this behavior the normal inverse Gaussian mixes a Gaussian distribution with a scalar *inverse Gaussian distribution* with PDF

$$p(z) = \sqrt{\frac{\chi}{2\pi z^3}} e^{\sqrt{\chi z} - (\chi z^{-1} + \psi z)/2}, \quad (2.15)$$

with  $\chi > 0$  and  $\psi > 0$ .

The parameters  $\alpha > 0$ ,  $\delta > 0$ , the vectors  $\mu$  and  $\beta$  (both  $n_x \times 1$ ), and the matrix  $\Gamma \succ 0$  ( $n_x \times n_x$ ) are then used to characterize the normal inverse Gaussian distribution [107]. A sample,  $x$ , from the distribution is obtained as

$$x = \mu + z\Gamma\beta + \sqrt{z}\Gamma^{-\frac{1}{2}}y, \quad (2.16)$$

where  $z$  is a sample from an inverse Gaussian distribution with  $\chi = \delta^2$  and  $\psi = \alpha^2 - \beta^T\Gamma\beta$  and  $y$  is a sample from a normalized Gaussian distribution. Conditioned on the value of the inverse Gaussian variable the distribution is Gaussian,  $x|z \sim \mathcal{N}(\mu + z\Gamma\beta, z\Gamma^{-1})$ . Hence, the distribution can be interpreted as a Gaussian distribution with stochastic mean and variance.

**Definition 2.9 (Inverse normal Gaussian distribution).** The normal inverse Gaussian PDF is given by

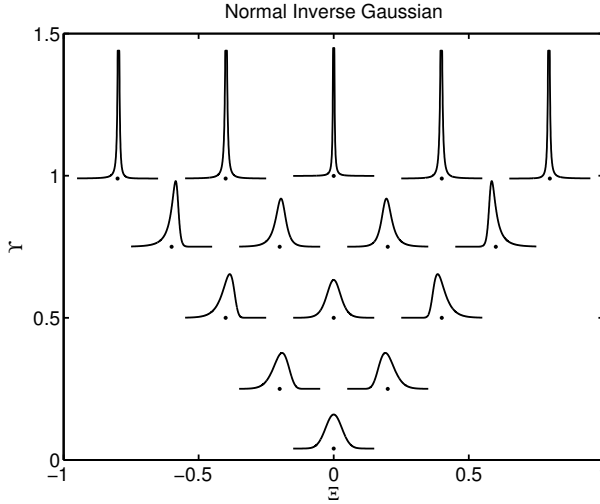
$$p(x; \alpha, \beta, \mu, \delta, \Gamma) = \frac{\delta}{2^{(n_x-1)/2}} \left( \frac{\alpha}{\pi\psi(\xi)} \right)^{(n_x+1)/2} e^{\phi(x)} K_{(n_x+1)/2}(\alpha\psi(x)),$$

where

$$\begin{aligned} \phi(x) &= \delta\sqrt{\alpha^2 - \beta^T\Gamma\beta} + \beta^T(x - \mu), \\ \psi(x) &= \sqrt{\delta^2 + (x - \mu)^T\Gamma^{-1}(x - \mu)}, \end{aligned}$$

where  $n_x = \dim(x)$  and  $K_d(x)$  is the *modified Bessel function of the second kind* with index  $d$  [107, 123]. The parameters should fulfill  $\alpha^2 > \beta^T\Gamma\beta$  and  $\det(\Gamma) = 1$ .

The pointedness of the distribution, and also indirectly how heavy the tails are, is controlled by the  $\alpha$  parameter. The greater  $\alpha$  is, the more narrow and the higher the peak becomes and the less weight is in the tails of the PDF. The parameter  $\beta$  controls the skewness of the distribution. The scalar parameter  $\delta$  can be used to scale the whole distribution. The matrix valued parameter  $\Gamma$  controls the way the components correlate. However,  $\Gamma$  diagonal is not enough to guarantee that the components are decoupled. For that  $\Gamma$  must be diagonal and  $\beta = 0$ . Furthermore, the Gaussian distribution  $\mathcal{N}(\mu + \sigma^2\Gamma\beta, \sigma^2\Gamma)$  is the limit case for  $\alpha \rightarrow \infty$  and  $\delta/\alpha \rightarrow \sigma^2$ , and in the scalar case  $p(0, 0, 1, 0)$  becomes the *Cauchy distribution*, [10].



**Figure 2.8:** Examples of the normal inverse Gaussian PDF parametrized in  $\Xi$  and  $\Upsilon$ . Each point in the plot corresponds to a parameter pair and the curve above it the resulting PDF.

The mean of the distribution is given by

$$E(x) = \mu + \frac{\beta\delta}{\sqrt{\alpha^2 - \beta^T\Gamma\beta}} \tag{2.17a}$$

and the covariance

$$\text{cov}(x) = \frac{\delta}{\sqrt{\alpha^2 - \beta^T\Gamma\beta}} \left( \Gamma + \frac{\Gamma\beta\beta^T\Gamma}{\alpha^2 - \beta^T\Gamma\beta} \right). \tag{2.17b}$$

Another parametrization of the normal inverse Gaussian is in terms of the parameters  $\Xi$ , normalized skewness, and  $\Upsilon$ , normalized pointedness, where

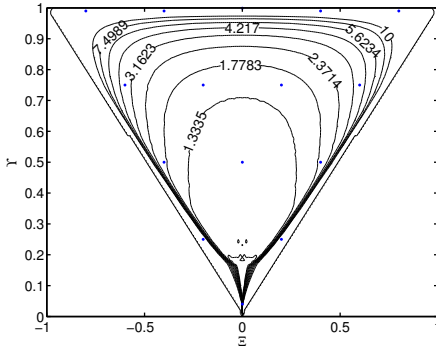
$$\Upsilon = (1 + \zeta)^{-\frac{1}{2}} \quad \text{and} \quad \Xi = \rho\Upsilon \tag{2.18}$$

with  $\rho = \alpha/\beta$  and  $\zeta = \delta\sqrt{\alpha^2 - \beta^T\Gamma\beta}$ . (See Figure 2.8 for an illustration of the effects of the two parameters.) For this parametrization, assuming given  $\Gamma$  and  $\delta$ , the allowed parameter space is  $0 \leq |\Xi| < \Upsilon < 1$ , forming a triangle as in Figure 2.8. This is an unambiguous transformation if the variance is set to 1. Towards the top, the distribution gets pointier whereas the skewness changes along the  $x$ -axis. Given  $\Xi$  and  $\Upsilon$  the original parameters are obtained as

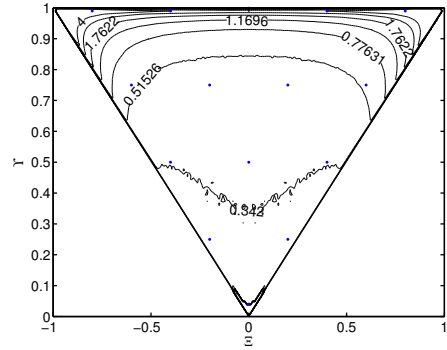
$$\alpha = \frac{\Upsilon\sqrt{1 - \Upsilon^2}}{\Upsilon^2 - \Xi^2}, \quad \beta = \frac{\Xi\sqrt{1 - \Upsilon^2}}{\Upsilon^2 - \Xi^2}, \quad \delta = \frac{(\alpha^2 - \beta^2)^{\frac{3}{2}}}{\alpha^2}. \tag{2.19}$$

For the same, alternative parametrization, the relative accuracy of the distribution and the Kullback divergence, compared to a normalized Gauss, are given in Figure 2.9. It can

be seen in the figure that the distribution gets more informative as it loses its symmetry, and also when it gets more pointy. This fits well with the experiences from the other distributions that have been studied.



(a) Inverse RA,  $\Psi^{-1}$ .



(b) Kullback divergence,  $\mathcal{J}^K(p(\Upsilon, \Xi), \mathcal{N}(0, 1))$ .

**Figure 2.9:** Relative accuracy and Kullback divergence for the normal inverse Gaussian, (2.16), with parameters (2.18). The dots indicate the parameter values of the PDFs shown in Figure 2.8.

# 3

---

## Functions of Distributions

**I**N THE INTRODUCTION to the previous chapter it was assumed that the measurements from the bearings-only sensors were given to the user as a noisy angle. Now assume that the sensor internally measures the bearing this way, and that the noise level is known, but delivers

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}_i = \begin{pmatrix} \cos(\theta_i + e_i) \\ \sin(\theta_i + e_i) \end{pmatrix}$$

to the user for some unknown reason. In order to effectively use these measurements one must determine how they are distributed. Another similar question arises if an estimate, in terms of a distribution, of the target position is known, and then the target moves and the move can be described by the function  $\varphi$ . What can be said about the distribution of the position after the move.

The purpose of this chapter is to answer this type of questions. The analytical solution is given in Section 3.1. Sections 3.2–3.4 describe three methods for approximating the analytic solution; the Monte Carlo transformation, the Gauss approximation, and the unscented transform. Section 3.5 presents an analysis of the differences between the Gauss approximation and the unscented transform, followed by an example comparing all the methods in Section 3.6.

### 3.1 Analytical Transformation

Assume that  $x$  is a stochastic variable to which the function  $\varphi : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_z}$  is applied, the result is a new stochastic variable  $z = \varphi(x)$  with a new distribution. As stated previously without motivation, if  $x$  is Gaussian and  $\varphi$  is a linear function then  $z$  is Gaussian as well, and the distribution is given by (2.8). The general way to determine the distribution of  $z$

is to use the definition of a PDF,

$$p_z(z) = \frac{d}{dz} \int_{\varphi(x) < z} p_x(x) dx, \quad (3.1)$$

which if  $\varphi$  is bijective simplifies to

$$p_z(z) = \frac{d}{dz} \int_{\varphi(x) < z} p_x(x) dx = \frac{d}{dz} \int_{-\infty}^{\varphi^{-1}(z)} p_x(x) dx = p_x(z) \frac{d\varphi^{-1}(z)}{dz}. \quad (3.2)$$

The expressions above are given for scalar variables but can be extended to vector valued variables using Theorem 3.1.

### Theorem 3.1

The density of  $z = \varphi(x)$ ,  $\varphi : \mathbb{R}^n \mapsto \mathbb{R}^n$  is

$$p_z(z) = \begin{cases} p_x(\varphi_1^{-1}(z), \varphi_2^{-1}(z), \dots, \varphi_{n_x}^{-1}(z)) |J|, & \text{for } z \in T \\ 0 & \text{otherwise} \end{cases},$$

where  $T$  is the image of the nonzero probability space of  $x$ ,  $\varphi^{-1}$  is the (unique) inverse of  $\varphi$ , and

$$J = \det(\nabla_z \varphi^{-1}(z)).$$

**Proof:** See the proof of Theorem 2.1 in Chapter I of [49]. □

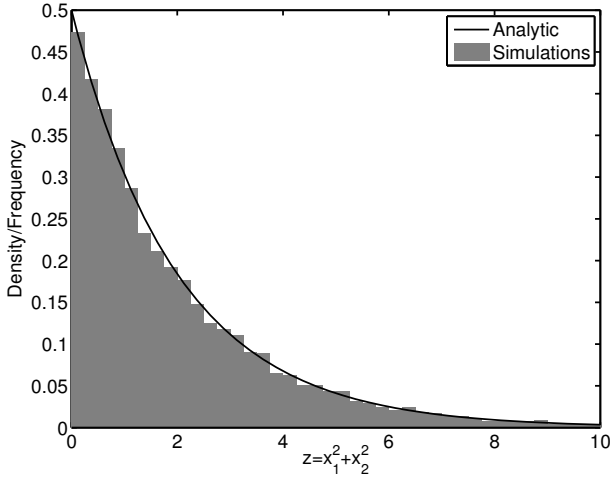
### Example 3.1: $\chi^2(2)$ distribution

Let  $x_1$  and  $x_2$  be two independent  $\mathcal{N}(0, 1)$  distributed stochastic variables, and let  $z = x_1^2 + x_2^2$ . Hence, if  $x_1$  and  $x_2$  are coordinates,  $z$  is the squared distance to origin. The distribution of  $z$  is then derived using (3.1),

$$\begin{aligned} p_z(z) &= \frac{d}{dz} \int_{x_1^2 + x_2^2 < z} \frac{1}{\sqrt{2\pi}} e^{-x_1^2/2} \cdot \frac{1}{\sqrt{2\pi}} e^{-x_2^2/2} dx_1 dx_2 = \left/ \text{Change to polar coord.} \right/ \\ &= \frac{d}{dz} \frac{1}{2\pi} \int_0^{\sqrt{z}} e^{-r^2/2} r dr \int_0^{2\pi} d\phi = \frac{d}{dz} \int_0^{\sqrt{z}} e^{-r^2/2} r dr = \frac{1}{2\sqrt{z}} e^{-z/2} \sqrt{z} = \frac{e^{-z/2}}{2}. \end{aligned}$$

The new distribution is exponential,  $\text{Exp}(\frac{1}{2})$ , which is a special case of the  $\chi^2$  distribution,  $\chi^2(2)$ . The distribution has  $E(z) = 2$  and  $\text{var}(z) = 4$ . The  $\chi^2(n)$  distribution is the result obtained when adding  $n$  independent and squared  $\mathcal{N}(0, 1)$  stochastic variables.

The analytic  $\chi^2(2)$  PDF and the result of 1 000 simulated samples are shown in Figure 3.1. Computing the sample mean and variance yields results close to the analytic values.



**Figure 3.1:** Analytical and simulated (1 000 samples) PDF of  $\chi^2(2)$ . Simulated mean: 2.0, variance: 6.0.

Even though the target distribution in Example 3.1 turned out to be analytically calculable, this is usually not the case. Hence, approximations are needed. The sequel of this chapter presents three such types of approximations: Monte Carlo transformation, Gauss approximation, and unscented transformation.

## 3.2 Monte Carlo Transformation

When an analytic solution to the distribution obtained as a function of a stochastic variable is not available, an intuitive approximative approach is to draw samples from the original distribution, apply the function, and see what happens. The resulting samples should in some way represent the new distribution. This is exactly what happens when *Monte Carlo methods* are used to approximate a function of a stochastic variable. This section will first describe the technique, and then motivate it outlining the more general *Monte Carlo integration*.

To approximate  $z = \varphi(x)$ , the distribution of  $x$  is approximated using a set of  $N \gg 1$  independent and identically distributed (IID) samples,  $\{x^{(i)}\}_{i=1}^N$ . The density of samples in a region now represents the PDF of  $x$ , often written as

$$p_x(x) \approx \sum_i \omega^{(i)} \delta(x - x^{(i)}),$$

where for now  $\omega^{(i)} = \frac{1}{N}$  for now and  $\delta(x - x^{(i)})$  is the *Dirac delta function* in  $x^{(i)}$ .

Samples from the target distribution is now obtained applying  $\varphi$  to all samples, i.e.,

$$z^{(i)} = \varphi(x^{(i)}),$$

and form the target distribution as

$$p_z(z) \approx \sum_i \omega^{(i)} \delta(z - z^{(i)}).$$

The approximation is the same as for  $p_x$ , and the density of samples in a region gives the actual PDF. The mean and covariance of  $p_z$  can be obtained as the sample mean and covariance,

$$\begin{aligned} \mu_z &= \sum_{i=1}^N \omega^{(i)} x^{(i)} \\ \Sigma_z &= \sum_{i=1}^N \omega^{(i)} (x^{(i)} - \mu_z)(x^{(i)} - \mu_z)^T. \end{aligned}$$

The larger set of samples used, the better the approximation becomes.

The theory behind Monte Carlo integration can be used to motivate this method. Monte Carlo integration [118] is a method to use statistical properties to compute integrals that are otherwise hard to handle. The idea is to reformulate difficult integrals on a form where computing an expected value renders the integral of interest. To illustrate this, consider the integral

$$I := \int \varphi(x) dx = \int g(x)p(x) dx,$$

where  $p$  should be a proper PDF and  $g(x) = \varphi(x)/p(x)$ . The value of the integral,  $I$ , can then be approximated with the sum

$$\hat{I}_N := \sum_{i=1}^N \frac{1}{N} g(x^{(i)}),$$

where  $\{x^{(i)}\}_{i=1}^N$  are  $N$  IID samples from the distribution given by  $p$ . The approximation uses that  $I = \mathbb{E}_p g(x)$  and that an expected value can be approximated with a sample mean. Furthermore, it follows from the *law of large numbers* that if  $\text{var}_p(g(x)) = \Sigma < +\infty$ , then

$$\lim_{N \rightarrow +\infty} \sqrt{N}(\hat{I}_N - I) \sim \mathcal{N}(0, \Sigma),$$

i.e.,  $\hat{I}_N \rightarrow I$  as  $N \rightarrow +\infty$  and the quality of the estimate improves with increasing  $N$  [32, 118]. Note, the convergence is in theory independent of the dimension of  $x$ , and Monte Carlo integration should hence suffer little from the *curse of dimensionality* in contrast to deterministic approximative integration methods. However, this is according to [29, 106] overly optimistic and Monte Carlo integration is claimed to suffer from the curse of dimensionality. This, on the other hand, seems too pessimistic for most applications in practice.

It may be difficult, or even impossible, to draw samples from  $p$ . This is sometimes the case with the posterior state distributions used later. If this is the problem, choose another

proper PDF  $q$  such that  $p(x) > 0$  implies  $q(x) > 0$  for  $x$  in the domain of  $p$ , i.e., the support of  $p$  is included in the support of  $q$ . Using  $q$  in the approximation yields,

$$\hat{I}_N = \sum_{i=1}^N \frac{p(x^{(i)})}{Nq(x^{(i)})} g(x^{(i)}) = \sum_{i=1}^N \omega^{(i)} g(x^{(i)}),$$

with the same limit and principal convergence as before. The distribution given by  $q$  is often called *proposal distribution* and  $\omega^{(i)}$  *importance weights*. Note that even if  $p$ , and thus  $\omega^{(i)}$ , is only known up to a normalizing constant, this is not a problem since

$$\sum_i \omega^{(i)} \rightarrow \int \frac{cp(x)}{q(x)} q(x) dx = \int cp(x) dx = c,$$

and it is hence possible to normalize the distribution and compute the integral,

$$\hat{I}_N = \frac{\sum_i \omega^{(i)} g(x^{(i)})}{\sum_i \omega^{(i)}}.$$

### 3.3 Gauss Approximation Formula

The Gauss approximation is to approximate the transform around the expected value of  $x$  using a *Taylor series expansion* around the mean  $\mu_x$ ,

$$\begin{aligned} \varphi(x) &= \varphi(\mu_x) + \nabla_x^T \varphi(\mu_x)(x - \mu_x) \\ &\quad + \frac{1}{2} [(x - \mu_x)^T \Delta_x^x \varphi(\mu_x)(x - \mu_x)]_i + \mathcal{O}(\|x - \mu_x\|^3) \end{aligned} \quad (3.3)$$

and disregard higher order terms. The notation  $[a_i]_i$  is used to denote a vector with  $i^{\text{th}}$  element  $a_i$ . Usually, terms of order two or three and higher are disregarded. Keeping only the first order term gives the classical Gauss approximation.

First and second order Gauss approximations are discussed in this section.

#### 3.3.1 First Order Gauss Approximation

The first order Gaussian approximation is sometimes denoted TT1, [44], to stress that a first order Taylor series is used in the function approximation. This approach makes no assumptions about the distribution of  $x$  other than that the mean and covariance are known. Given this, the mean of  $z$  is approximated by

$$\begin{aligned} \mu_z &= \mathbb{E}_z(z) = \mathbb{E}_x(\varphi(x)) \\ &= \mathbb{E}_x\left(\varphi(\mu_x) + \nabla_x^T \varphi(\mu_x)(x - \mu_x) + \mathcal{O}(\|x - \mu_x\|^2)\right) \\ &\approx \mathbb{E}(\varphi(\mu_x)) + \nabla_x^T \varphi(\mu_x) \mathbb{E}(x - \mu_x) = \varphi(\mu_x), \end{aligned} \quad (3.4a)$$

and the covariance matrix with

$$\begin{aligned}
\Sigma_z &= \text{cov}_z(z) = \text{cov}_x(\varphi(x)) = \mathbb{E}_x \left( (\varphi(x) - \mu_z)(\varphi(x) - \mu_z)^T \right) \\
&= \mathbb{E}_x \left( \left( \varphi(\mu_x) + \nabla_x^T \varphi(\mu_x)(x - \mu_x) + \mathcal{O}(\|x - \mu_x\|^2) - \varphi(\mu_x) \right) \left( \cdot \right)^T \right) \\
&\approx \nabla_x^T \varphi(\mu_x) \mathbb{E}_x \left( (x - \mu_x)(x - \mu_x)^T \right) \nabla_x \varphi(\mu_x) \\
&= \nabla_x^T \varphi(\mu_x) \Sigma_x \nabla_x \varphi(\mu_x), \tag{3.4b}
\end{aligned}$$

where, the notation  $(A)(\cdot)^T = AA^T$  has been used to improve the readability and save space. The method gives approximations of the first two moments of the distribution of  $z$ . Usually the distribution is then approximated with a Gauss distribution, with the computed mean and covariance. This works fairly well in many situations where  $x$  is Gaussian and the transformation is fairly linear and it is very fast once the needed derivatives are obtained.

### 3.3.2 Second Order Gauss Approximation

Keeping yet another term in the Taylor series expansion yields an approximation that also takes second order effects in  $\varphi$  into account, [4, 9, 45]. This method is denoted TT2 by some authors, [44]. This is mostly seen together with the assumption that  $x$  is Gaussian to considerably simplify the obtained expressions. The idea with another term in the Taylor series expansion is that it will give a more accurate function approximation and this way improve the approximation. A consequence of this is that it is necessary to make the assumption that  $x$  is Gaussian to obtain the covariance approximation. The mean approximation is with the second order Taylor series expansion, using  $\bar{x} = x - \mu_x$  for notational convenience,

$$\begin{aligned}
\mu_z &= \mathbb{E}_z z = \mathbb{E}_x \varphi(x) \\
&= \mathbb{E}_x \left( \varphi(\mu_x) + \nabla_x^T \varphi(\mu_x) \bar{x} + \frac{1}{2} [\bar{x}^T \Delta_x^x \varphi(\mu_x) \bar{x}]_i + \mathcal{O}(\|\bar{x}\|^3) \right) \\
&\approx \varphi(\mu_x) + \frac{1}{2} \mathbb{E}_x [\bar{x}^T \Delta_x^x \varphi(\mu_x) \bar{x}]_i \\
&= \varphi(\mu_x) + \frac{1}{2} [\text{tr}(\Delta_x^x \varphi_i(\mu_x) \Sigma_x)]_i, \tag{3.5a}
\end{aligned}$$

and the covariance

$$\begin{aligned}
\Sigma_z &= \text{cov}_z(z) = \text{cov}_x(\varphi(x)) \\
&= \mathbb{E}_x \left( \left( \varphi(\mu_x) - \varphi(\mu_x) + \nabla_x^T \varphi(\mu_x) \bar{x} \right. \right. \\
&\quad \left. \left. + \frac{1}{2} [\bar{x}^T \Delta_x^x \varphi_i(\mu_x) \bar{x}]_i - \frac{1}{2} [\text{tr}(\Delta_x^x \varphi_i(\mu_x) \Sigma_x)]_i + \mathcal{O}(\|\bar{x}\|^2) \right) \left( \cdot \right)^T \right)
\end{aligned}$$

$$\begin{aligned}
&\approx \nabla_x^T \varphi(\mu_x) \mathbf{E}_x \left( (x - \mu_x)(x - \mu_x)^T \right) \nabla_x \varphi(\mu_x) \\
&\quad + \frac{1}{2} \nabla_x^T \varphi(\mu_x) \\
&\quad \mathbf{E}_x \left( (x - \mu_x) \left[ (x - \mu_x)^T \Delta_x^x \varphi_i(\mu_x)(x - \mu_x) - \text{tr}(\Delta_x^x \varphi_i(\mu_x) \Sigma_x) \right]_i^T \right) \\
&\quad + \frac{1}{2} \mathbf{E}_x \left( \left[ (x - \mu_x)^T \Delta_x^x \varphi_i(\mu_x)(x - \mu_x) - \text{tr}(\Delta_x^x \varphi_i(\mu_x) \Sigma_x) \right]_i (x - \mu_x)^T \right) \\
&\quad \nabla_x \varphi(\mu_x) \\
&\quad + \frac{1}{4} \mathbf{E} \left( \left[ (x - \mu_x)^T \Delta_x^x \varphi_i(\mu_x)(x - \mu_x) - \text{tr}(\Delta_x^x \varphi_i(\mu_x) \Sigma_x) \right]_i (\cdot)^T \right).
\end{aligned}$$

Up until this step no assumptions has been made about  $x$  other than that is has known mean and covariance. To further simplify the expression it is now assumed that  $x$  is Gaussian. Two properties of the Gaussian distribution are used. The symmetry makes the third order term cancel out when the expected value is taken. That leaves a second order term and a fourth order term. For a Gaussian variable,  $\mathbf{E} \bar{x}^4 = 3 \mathbf{E} \bar{x}^2$ , yielding

$$\Sigma_z = \nabla_x \varphi(\mu_x) P \nabla_x^T \varphi(\mu_x) + \frac{1}{2} \left[ \text{tr}(P \Delta_x^x \varphi_i(\mu_x) P \Delta_x^x \varphi_j(\mu_x)) \right]_{ij}. \quad (3.5b)$$

The notation  $[a_{ij}]_{ij}$  is used to denote the matrix  $A$  with the  $(i, j)$  element  $a_{ij}$ .

With higher order terms in the function approximation, the result of the transformation is likely to be somewhat better. However, in practice the number of cases where this plays an important role seems to be limited. The paper [8] presents a long range radar measurement case where applying this second order compensated EKF gives a clear improvement.

### 3.4 Unscented Transform

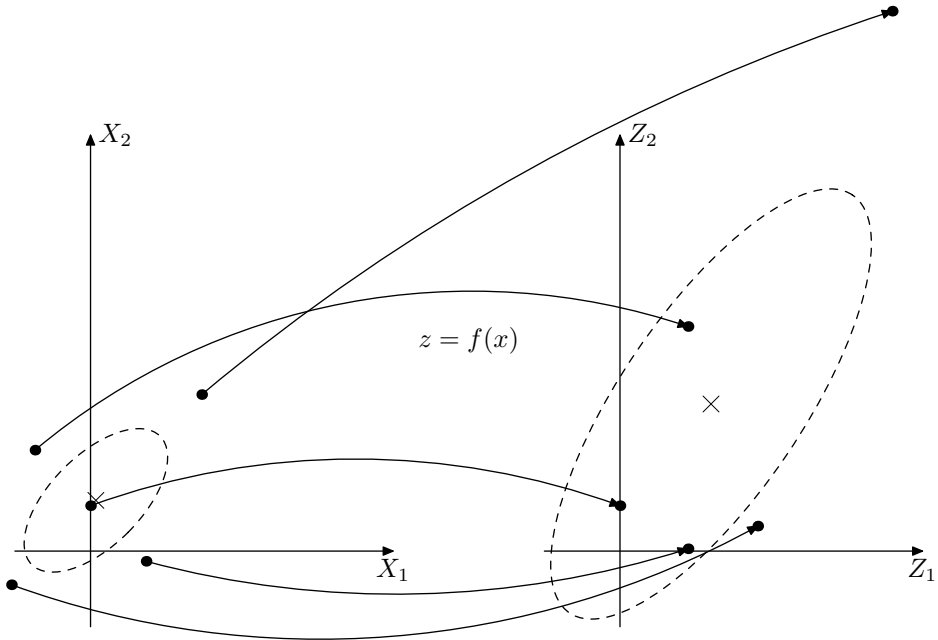
The *unscented transform* (UT), introduced by Julier [65], Julier and Uhlmann [66, 67], is a recent method used to transform stochastic variables. The unscented transform was designed as a step in the creation of the *unscented Kalman filter* (UKF). The UKF is discussed in Chapter 5 as an alternative filtering technique for nonlinear systems.

The basic idea of the unscented transform is to have a set of carefully chosen instances of the initial stochastic variable, called *sigma points*, pass through the transformation, and based on the points derive the mean and the covariance matrix of the transformed distribution. The idea is illustrated in Figure 3.2. Note that this differs considerably at a conceptual level from Monte Carlo based methods in that the unscented transform does not rely on stochastics but is completely deterministic. One noticeable difference stemming from this is that the result of the unscented transform is identical each time used on a problem, whereas the Monte Carlo simulation based method yields slightly different results each time depending on the exact realization of random numbers used.

The standard form of the unscented transform uses  $N = 2n_x + 1$  sigma points, where  $n_x$  is the dimension of  $x$ . The sigma points,  $x^{(i)}$ , and the associated weights,  $\omega^{(i)}$ , are chosen as

$$x^{(0)} = \mathbf{E}(x), \quad \omega^{(0)} \text{ is parameter}, \quad (3.6a)$$

$$x^{(\pm i)} = x^{(0)} \pm \sqrt{\frac{n_x}{1 - \omega^{(0)}}} u_i, \quad \omega^{(\pm i)} = \frac{1 - \omega^{(0)}}{2n_x}, \quad (3.6b)$$



**Figure 3.2:** Sigma points for  $x \sim \mathcal{N}((0 \ 1)^T, I)$  are passed through the nonlinear function  $z = \varphi(x) = (x_1^2, x_2^2)^T$ . (An  $\times$  denotes estimated mean and a dashed ellipse estimated covariance.)

for  $i = 1, \dots, n_x$ , where  $u_i$  is the  $i^{\text{th}}$  column in the square root of  $\text{cov}(x)$ , where  $B$  is the a square root of  $A$  if  $A = BB^T$ . The  $i^{\text{th}}$  column of the square root of the covariance matrix is used since it represents the standard deviation of the distribution in a principal direction. The set of sigma points hence spans the uncertainty of the stochastic variable. The weight on the mean,  $\omega^{(0)}$ , is used for tuning. A small value of  $\omega^{(0)}$  moves the sigma points away from the mean, whereas a large value gathers them closer to the center. This allows for tuning of the unscented transform. The weight  $\omega^{(0)} = 1 - \frac{n_x}{3}$  gives, according to [67], preferable properties for Gaussian distributions. It is also possible to use other sets of sigma points and/or parameterizations to change the behavior of the approximation and this way get more degrees of freedom for tuning [67].

Once the sigma points are chosen, the approximations of  $\mu_z$  and  $\Sigma_z$  are computed as weighted means. Denote the transformed sigma points

$$z^{(i)} = \varphi(x^{(i)}),$$

for  $i = -n_x, \dots, n_x$  and associate them with the weights  $\omega^{(i)}$ . The estimated mean follows as

$$\mu_z = \sum_{i=-n_x}^{n_x} \omega^{(i)} z^{(i)}, \quad (3.7a)$$

and the covariance matrix as

$$\Sigma_z = \sum_{i=-n_x}^{n_x} \omega^{(i)} (z^{(i)} - \mu_z)(z^{(i)} - \mu_z)^T. \quad (3.7b)$$

Once again, since only the mean and the covariance are known, a Gaussian approximation is often used to represent the result.

One problem with the unscented transform as presented above is that it sometimes produces indefinite covariance matrix estimates. As a way to solve this, and to also improve the performance further, techniques have been proposed to slightly change the weights and the placement of the sigma points. The most common variation, combining two modifications [67, 146], introduces an extra weight on the mean sigma point when computing the covariance and a different set of tuning parameters. The covariance estimate then becomes

$$\Sigma_z = \sum_{i=-n_x}^{n_x} \omega^{(i)} (z^{(i)} - \mu_z)(z^{(i)} - \mu_z)^T + (1 - \alpha^2 + \beta)(z^{(0)} - \mu_z)(z^{(0)} - \mu_z)^T, \quad (3.8)$$

with  $\omega^{(0)}$  chosen as  $\omega^{(0)} = \frac{\lambda}{n_x + \lambda}$  where  $\lambda = \alpha^2(n_x + \kappa) - n_x$ .

It is common to introduce the concept of different weights for mean and covariance approximation. Hence, use  $\omega_m^{(i)}$  for the mean and  $\omega_c^{(i)}$  for the covariance, and the only difference between the two sets of weights being  $w_c^{(0)} = w_m^{(0)} + (1 - \alpha^2 + \beta)$ .

There are three new parameters introduced with this modification:

$\alpha$  — a primary scaling factor determining how far from the center the sigma points should be placed. The recommendation is  $\alpha \approx 10^{-3}$ , *i.e.*, keep the sigma points close to the center.

$\beta$  — a distribution compensation taking care of higher order effects. In [67],  $\beta = 2$  is claimed to be optimal for Gaussian distributions.

$\kappa$  — a secondary scaling factor, usually chosen to  $\kappa = 0$ .

Some observations; the weights for the covariance estimate sums up to more than one, and  $\omega_m^{(0)} = 1 - \frac{1}{\alpha^2} \ll 0$  if  $\kappa = 0$  and  $\alpha \ll 1$ , which differs substantially from the suggestion without the covariance compensation.

According to Julier and Uhlmann [67] it is possible to get correct estimates of mean and variance to the second order, and even higher orders, using the unscented transform. However, very little is said about how and under what conditions this holds, and Example 3.2 shows how the unscented transform sometimes produce poor approximations for relatively simple transformations.

---

### Example 3.2: Problem with unscented transform

---

Assume, as in Example 3.1, two independent stochastic variables,  $x_1 \sim \mathcal{N}(0, 1)$  and  $x_2 \sim \mathcal{N}(0, 1)$ . The transformation  $z = x_1^2 + x_2^2$  is then  $\chi^2(2)$  distributed, see Example 3.1, with mean  $E(z) = 2$  and variance  $\text{var}(z) = 4$ .

Using the unscented transform with  $\omega^{(0)}$  as parameter yields and the sigma points

$$x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad x^{(\pm 1)} = \begin{pmatrix} \pm \sqrt{\frac{2}{1-\omega^{(0)}}} \\ 0 \end{pmatrix}, \quad x^{(\pm 2)} = \begin{pmatrix} 0 \\ \pm \sqrt{\frac{2}{1-\omega^{(0)}}} \end{pmatrix},$$

Applying the transform, on standard form, results in the sigma points  $z^{(0)} = 0$  and  $z^{(\pm 1)} = z^{(\pm 2)} = \frac{2}{1-\omega^{(0)}}$  and the estimated mean and variance

$$\mu_z = \sum_i \omega^{(i)} z^{(i)} = 2$$

$$\Sigma_z = \sum_i \omega^{(i)} (z^{(i)} - \mu_z)^2 = \frac{4\omega^{(0)}}{1-\omega^{(0)}},$$

where the variance differs from the variance for a  $\chi^2(2)$  distribution unless  $\omega^{(0)} = \frac{1}{2}$ . Hence, in this case the unscented transform, with  $\omega^{(0)}$  chosen as recommended in [67], will not produce a correct approximation of the distributions second moment, even for a quadratic function.

Instead using the modified unscented transform yields

$$\mu_z = \sum_i \omega_m^{(i)} z^{(i)} = 2$$

$$\Sigma_z = \sum_i \omega_c^{(i)} (z^{(i)} - \mu_z)^2 = \alpha^2 \kappa + 4\beta.$$

This, as well, differs from the correct values with the standard values  $\kappa = 0$  and  $\beta = 2$  (in this case the choice of  $\alpha$  does not influence the result).

It is however unfair to say that the unscented transform always fails to produce good estimates. Below follows an example when it works well.

### Example 3.3: Successful application of unscented transform

Let  $x \sim \mathcal{N}(1, 1)$  and  $z = x^2$ . The PDF for  $z$  is then

$$p_z(z) = \frac{d}{dz} \int_{x^2 < z} \frac{1}{\sqrt{2\pi}} e^{-(x-1)^2/2} dx = \frac{1}{\sqrt{2\pi}z} e^{-(z+1)/2} \cosh(\sqrt{z}),$$

which turns out to be a non-central  $\chi^2$  distribution. Hence,  $z \sim \chi_1^2(1)$  with expected value  $E(z) = 2$  and variance  $\text{var}(z) = 6$ .

The Gauss approximation of the target distribution is

$$\mu_z = \mu_x^2 = 1 \quad \Sigma_z = 2\mu_x \cdot 1 \cdot 2\mu_x = 4$$

for the first order approximation, and

$$\mu_z = \mu_x^2 = 2 \quad \Sigma_z = 2\mu_x \cdot 1 \cdot 2\mu_x + \frac{1}{2} \cdot 2\mu_x \cdot 1 \cdot 2\mu_x = 6$$

for the second order approximation.

Finally, consider the unscented transform. The sigma points are using the weights  $\omega^{(0)} = \omega$ :

$$\begin{aligned}
 x^{(0)} &= \mu_x = 1, & z^{(0)} &= (x^{(0)})^2 = 1, \\
 x^{(-1)} &= \mu_x - \sqrt{\frac{1 \cdot \text{var}(x)}{1 - \omega^{(0)}}}, & z^{(-1)} &= (x^{(-1)})^2 = \frac{2 - \omega}{1 - \omega} - 2\sqrt{\frac{1}{1 - \omega}}, \\
 x^{(1)} &= \mu_x + \sqrt{\frac{1 \cdot \text{var}(x)}{1 - \omega^{(0)}}}, & z^{(1)} &= (x^{(1)})^2 = \frac{2 - \omega}{1 - \omega} + 2\sqrt{\frac{1}{1 - \omega}},
 \end{aligned}$$

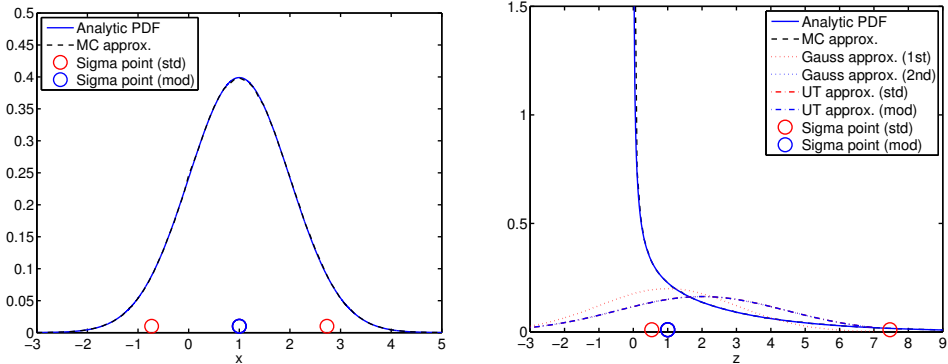
Using this, first for the standard formulation ( $\omega = \frac{2}{3}$ )

$$\mu_z = \sum_{i=-1}^1 \omega^{(i)} z^{(i)} = 2, \quad \Sigma_z = \sum_{i=-1}^1 \omega^{(i)} (z^{(i)} - \mu_z)^2 = 3 + \frac{1}{1 - \omega} = 6,$$

and for the modified version with  $\alpha = 10^{-3}$ ,  $\beta = 2$ , and  $\kappa = 0$

$$\mu_z = \sum_{i=-1}^1 \omega_m^{(i)} z^{(i)} = 2, \quad \Sigma_z = \sum_{i=-1}^1 \omega_c^{(i)} z^{(i)} (z^{(i)} - \mu_z)^2 = 4 + \beta + \alpha^2 \kappa = 6.$$

The results of applying the three approximative methods to derive the distribution of  $z$  are gathered in Figure 3.3. Note that performance is discouraging for the first order Gauss approximation, but works well with the remaining ones. The PDF estimated in all cases but the Monte Carlo transformation differs significantly from the true distribution. Hence, it is important to make sure that the result is acceptable if an approximate transformation is used.



(a) PDF before the transformation.

(b) PDF after the transformation.

**Figure 3.3:** Result of squaring a normalized Gaussian distribution: analytically, with Gauss approximation formula, with Monte Carlo simulations and with unscented transformation. (Used sigma points are included for reference.)

### 3.5 Asymptotic Analysis of the Unscented Transform

In this section the unscented transform will be analyzed and expressions for the resulting mean and covariance are given and interpreted in the limit as the sigma points approach the center point. To get the sigma points to gather close to the center point,  $\omega^{(0)}$  should be chosen such that  $n_x/(1 - \omega^{(0)}) \rightarrow 0^+$ , i.e.,  $\omega^{(0)} \rightarrow -\infty$ . This will only happen with the standard formulation of the unscented transform for very large dimensions if the recommended weights are used, since  $\omega^{(0)} = 1 - \frac{n_x}{3}$ . However, the same is not true for the modified version where

$$\frac{n_x}{1 - \omega^{(0)}} = n_x + \lambda = \alpha^2(n_x + \kappa),$$

which for the recommended  $\alpha = 10^{-3}$  and  $\kappa = 0$  gives  $n_x/(1 - \omega^{(0)}) \ll 1$ .

To see what happens in the limit, first consider the mean of the transformed variable, for simplicity  $\omega = \omega^{(0)}$  and express the remaining weights in terms of it,

$$\mu_z = \sum_{i=-n_x}^{n_x} \omega^{(i)} z^{(i)} = z^{(0)} + \frac{1 - \omega}{2n_x} \sum_{i=1}^{n_x} (z^{(i)} - 2z^{(0)} + z^{(-i)}). \quad (3.9a)$$

Calculating the covariance matrix yields the expression:

$$\begin{aligned} \Sigma_z &= (\omega + (1 - \alpha^2 + \beta))(z^{(0)} - \mu_z)(\cdot)^T \\ &\quad + \sum_{i \neq 0} \frac{1 - \omega}{2n_x} (z^{(i)} - \mu_z)(\cdot)^T \\ &= (1 - \alpha^2 + \beta) \frac{(1 - \omega)^2}{4n_x^2} \left( \sum_{i=1}^{n_x} (z^{(i)} - 2z^{(0)} + z^{(-i)}) \right) (\cdot)^T \\ &\quad + \frac{\omega(1 - \omega)^2}{4n_x^2} \left( \sum_{i=1}^{n_x} (z^{(i)} - 2z^{(0)} + z^{(-i)}) \right) (\cdot)^T \\ &\quad + \frac{(1 - \omega)(\omega^2 - 1)}{4n_x^2} \left( \sum_{i=0}^{n_x} (z^{(i)} - 2z^{(0)} + z^{(-i)}) \right) (\cdot)^T \\ &\quad + \frac{1 - \omega}{2n_x} \sum_{\substack{i=-n_x \\ i \neq 0}}^{n_x} (z^{(i)} - z^{(0)})(\cdot)^T \\ &= (1 - \alpha^2 + \beta) \frac{(1 - \omega)^2}{4n_x^2} \left( \sum_{i=1}^{n_x} (z^{(i)} - 2z^{(0)} + z^{(-i)}) \right) (\cdot)^T \\ &\quad - \frac{(1 - \omega)^2}{4n_x^2} \left( \sum_{i=1}^{n_x} (z^{(i)} - 2z^{(0)} + z^{(-i)}) \right) (\cdot)^T \\ &\quad + \frac{1 - \omega}{n_x} \sum_{i=-n_x}^{n_x} (z^{(i)} - z^{(0)})(\cdot)^T. \end{aligned} \quad (3.9b)$$

Assume that the sigma points are constructed using a singular value decomposition (SVD) of a positive definite matrix [40],

$$\Sigma_x = U \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_{n_x}^2) U^T,$$

with  $U = (u_1 \ u_2 \ \dots \ u_{n_x})$ . The purpose of using the SVD here is to obtain the orthogonal principal axis of the covariance matrix. These directions are given by  $u_i$  and the magnitude in that direction by  $\sigma_i$ . This gives the sigma points  $x^{(\pm i)} = \mu_x \pm \sqrt{n_x + \lambda} \sigma_i u_i$ .

With these sigma points, differences can be constructed that in the limit as  $n_x + \lambda \rightarrow 0^+$ , i.e.,  $\alpha \rightarrow 0^+$  with  $\kappa = 0$ , yields the derivatives:

$$\frac{z^{(i)} - z^{(0)}}{\sigma_i \sqrt{n_x + \lambda}} \rightarrow \nabla_x \varphi(\mu_x) u_i \quad (3.10)$$

$$\frac{z^{(i)} - 2z^{(0)} + z^{(-i)}}{\sigma_i^2 (n_x + \lambda)} \rightarrow [u_i^T \Delta_x^x \varphi_k(\mu_x) u_i]_k. \quad (3.11)$$

Note that  $n_x + \lambda = n_x / (1 - \omega)$ .

Using this, the limit case of (3.9) can be evaluated,

$$\mu_z \rightarrow \varphi(\mu_x) + \frac{1}{2} [\text{tr}(\Delta_x^x \varphi_i(\mu_x) \Sigma_x)]_i \quad (3.12a)$$

and

$$\begin{aligned} \Sigma_z &\rightarrow \nabla_x \varphi(\mu_x) \Sigma_x (\nabla_x \varphi(\mu_x))^T \\ &+ \frac{(\beta - \alpha^2)}{4} [\text{tr}(\Sigma_x \Delta_x^x \varphi_i(\mu_x)) \text{tr}(\Sigma_x \Delta_x^x \varphi_j(\mu_x))]_{ij}. \end{aligned} \quad (3.12b)$$

The expressions above will now be used to compare the modified unscented transform with the second order Gauss approximation, that is correct for any second order transformations of a Gaussian variable.

For a completely scalar case,  $\varphi : \mathbb{R} \mapsto \mathbb{R}$ , the second order Gauss approximation and the unscented transform yield the same result ( $\beta - \alpha^2 \approx 2$ , for the suggested parameter values)

$$\mu_z = \varphi(\mu_x) + \frac{1}{2} \Sigma_x \varphi''(\mu_x) \quad (3.13a)$$

$$\Sigma_z = (\varphi'(\mu_x))^2 \Sigma_x + \frac{1}{2} (\varphi''(\mu_x) \Sigma_x)^2. \quad (3.13b)$$

For general multidimensional case, the second order Gauss approximation (3.5) and the unscented transform (3.12) differs only in their terms compensating for the quadratic effects in the covariance. Hence, both produce the same mean estimate, and include the same first term in their covariance estimate. The correct second order compensation, given by the second order Gauss approximation, is

$$\frac{1}{2} [\text{tr}(\Sigma_x \Delta_x^x \varphi_i(\mu_x) \Sigma_x \Delta_x^x \varphi_j(\mu_x))]_{ij} \quad (3.14)$$

and the corresponding unscented transform compensation is

$$\frac{(\beta - \alpha^2)}{4} [\text{tr}(\Sigma_x \Delta_x^x \varphi_i(\mu_x)) \text{tr}(\Sigma_x \Delta_x^x \varphi_j(\mu_x))]_{ij}, \quad (3.15)$$

where  $\beta - \alpha^2 \approx 2$  is recommended. The terms are similar, but not identical. The reason for the difference is that the unscented transform cannot express the mixed second order derivatives needed for the second order Gauss approximation compensation term without increasing the number of sigma points considerably. The result of this approximation depends on the transformation and must be analyzed for the case at hand.

### 3.6 Comparative Example

This section returns to the sum of squared random Gaussian variables and extends Example 3.3. It will show on some differences between the approximations. Even though the function is purely second order,  $f(x) = x^T x$ , it manages to differentiate between the methods, and the results can be used to draw some conclusions.

First of all, it is a well-known fact that if  $x \sim \mathcal{N}(0, I)$  of dimension  $n_x$ , then  $z = \varphi(x) \sim \chi^2(n_x)$ . The  $\chi^2(n_x)$  distribution has the analytic mean  $n_x$  and the variance  $2n_x$ .

For the first order Taylor series approximation, a function evaluation in the mean and a gradient are needed

$$\varphi(\mu_x) = 0 \qquad \nabla_x \varphi(\mu_x) = 2\mu_x = 0.$$

Using these the first order Gaussian approximation becomes

$$\mu_z = 0 \qquad \Sigma_z = 0.$$

To obtain a second order Gauss approximation the Hessian must be computed as well,

$$\Delta_x^x \varphi(\mu_x) = 2I.$$

With this, the approximation yields the correct values

$$\mu_z = 0 + \frac{1}{2} \text{tr}(2I) = n_x \qquad \Sigma_z = 0 + \frac{1}{2} \text{tr}(2I \cdot 2I) = 2n_x.$$

Now resorting to the unscented transform, for which the sigma points are

$$\begin{aligned} x^{(0)} &= \mu_x = 0 & \omega^{(0)} &= \omega \\ x^{(\pm i)} &= \mu_x \pm \sqrt{\frac{n_x}{1-\omega}} & \omega^{(\pm i)} &= \frac{\omega - 1}{2n_x}. \end{aligned}$$

Squaring the sigma points yields  $z^{(0)} = 0$  and  $z^{(\pm i)} = n_x/(1-\omega)$ . The standard unscented transform then gives

$$\mu_z = \sum_i \omega^{(i)} z^{(i)} = n_x \qquad \Sigma_z = \sum_i \omega^{(i)} (z^{(i)} - \mu_z)^2 = \frac{\omega n_x}{1-\omega},$$

where the recommended choice  $\omega = 1 - \frac{n_x}{3}$  yields  $\Sigma_z = n_x(3 - n_x)$ . Note that this results in a *negative variance* approximation whenever the dimension exceeds 3.

For the modified unscented transform, a different set of parameters are used,

$$\mu_z = \sum_i \omega_m^{(i)} z^{(i)} = n_x \qquad \Sigma_z = \sum_i \omega_c^{(i)} (z^{(i)} - \mu_z)^2 = \alpha^2 \kappa n_x + \beta n_x^2.$$

Assuming the standard values for the parameters;  $\alpha = 10^{-3}$ ,  $\beta = 2$ , and  $\kappa = 0$ , the resulting covariance estimate is  $\Sigma_z = 2n_x^2$ . This is better than the other unscented estimate since it is always positive, but too large when  $n_x > 1$ .

The results obtained above are compiled in Table 3.1, the estimates are also compared for all the methods using recommended parameter values. Note how the first order Gaussian approximation is totally wrong, the variance using the standard unscented transform

becomes 0 and then negative when the dimension becomes 3 and higher. This can cause unexpected and serious problems if the results are used in other algorithms. The second order Gaussian approximation is correct, as should be expected for a second order function. The Monte Carlo approximation tends towards the correct values, but the exact result is random.

What can not be seen from the data listed in Table 3.1 is that the Monte Carlo approximation actually quite well captures the true  $\chi^2$  distribution, cf. Figure 3.1, whereas all the others merely give mean and variance. This gives an advantage to the Monte Carlo approximation, that in some situations can be useful.

**Table 3.1:** Result when trying to approximate  $x^T x$ ,  $x \sim \mathcal{N}(0, I)$ , using different approximation methods. The result is a  $\chi^2(n_x)$ , but all but the MC result are usually interpreted as Gaussian. Standard parameter values:  $\omega^{(0)} = 1 - \frac{n}{3}$  (std UT),  $\alpha = 10^{-3}$ ,  $\beta = 2$ , and  $\kappa = 0 \pmod{\text{UT}}$ , and  $10^5$  samples (MC). TT1 and TT2 are the Gaussian approximation of first and second order, respectively.

$n_x$	TT1		TT2		UT (std)		UT (mod)		MC	
	$\mu_z$	$\Sigma_z$	$\mu_z$	$\Sigma_z$	$\mu_z$	$\Sigma_z$	$\mu_z$	$\Sigma_z$	$\mu_z$	$\Sigma_z$
1	0	0	1	2	1	2	1	2	1.01	2.05
2	0	0	2	4	2	2	2	8	2.00	3.99
3	0	0	3	6	3	0	3	18	2.99	5.94
4	0	0	4	8	4	-4	4	32	4.01	8.04
5	0	0	5	10	5	-10	5	50	4.98	9.90
$n$	0	0	$n$	$2n$	$n$	$n(3 - n)$	$n$	$2n^2$	$\sim$	$\sim$



# 4

---

## Models

**T**O EXTRACT INFORMATION from measurements of a system, *i.e.*, estimate the underlying state of the system or detect a change in its behavior, it is important to have an accurate description of the system. A good system model describes how the measurable output reflects the internal state of the system and how it responds to input. For different applications, different descriptions are needed. They can be anything from textual descriptions, to measured frequency response diagrams, to strict mathematical descriptions, *e.g.*, transfer functions or state-space models.

In the case of the bearings-only setup, introduced in Section 1.1, a model can be built using only knowledge of high school mathematics. First, assume that the target is essentially not moving except for small random jumps. This is described by

$$x_{t+1} = x_t + w_t.$$

The position at time  $t + 1$  is the same as at  $t$  except for a little disturbance  $w_t$  given by a stochastic variable as in Chapter 2. The measurements are given by straightforward geometry,

$$y_t = h(x_t) + e_t = \arctan\left(\frac{y_t - y^t}{x_t - x^t}\right) + e_t, = \theta_t + e_t,$$

where  $e_t$  is measurement noise described by a stochastic variable. The mathematical description this results in is a nonlinear state-space model, which is one of the classes of models described in this chapter. Another description is the hidden Markov model, which uses two PDFs

$$\begin{aligned} p(x_{t+1}|x_t) &= p_w(x_{t+1} - x_t) \\ p(y_t|x_t) &= p_e(y_t - \theta_t). \end{aligned}$$

The outline of this chapter is: Section 4.1 studies these two general model descriptions; the *hidden Markov model* (HMM) and the *state-space model*. They are sometimes

too general to be useful and therefore special state-space forms are presented in Section 4.2. Finally it is shown in Section 4.3 how the presented models can be extended to incorporate faults.

## 4.1 General Model

This section introduces general model structures that can be used to describe almost any system encountered in a structured mathematical way. The presentation focuses on two different ways to represent the model, using the hidden Markov form and the state-space form.

### 4.1.1 Hidden Markov Model

A probabilistic model describes how probable different events or states of a model are, [117]. In Bayesian statistics, introduced by Bayes [13] in 1763, parameters are stochastic variables with distributions instead of unknown but deterministic values as in classical statistics. Prior knowledge and measurements are hence used to derive an as correct parameter distribution as possible,

$$p(\mathbb{X}_t | \mathbb{Y}_\tau).$$

In this context  $x_t$  and  $y_t$  are the state of the system and a measurement of the system, respectively, at time  $t$ . The notation  $\mathbb{X}_t$  is the ordered set of states up until and including  $x_t$ , and  $\mathbb{Y}_t$  is defined analogous for the measurements. The *Hidden Markov model* (HMM) is a statistical model where one *Markov process*, representing the underlying system, is observed through a stochastic process. That is,  $x_t$  is not directly visible in  $y_t$ .

The dynamics of the model is given by the probability distribution

$$p(x_{t+1} | \mathbb{X}_t, \mathbb{U}_t) = p(x_{t+1} | x_t, u_t), \quad (4.1a)$$

where  $u_t$  is a deterministic and known input to the system. To make the model complete, information about  $x_0$  must be supplied. This gives a stochastic relation between the state at time  $t + 1$ ,  $x_{t+1}$ , and the state  $x_t$  and the input  $u_t$  at time  $t$  (more generally between times  $t_{i+1}$  and  $t_i$ ). The Markov property of the system makes everything but  $x_t$  and  $u_t$  uninteresting for predicting  $x_{t+1}$ , hence all important information about the past is contained in  $x_t$ .

Observations of the system are made through a stochastic process, described by

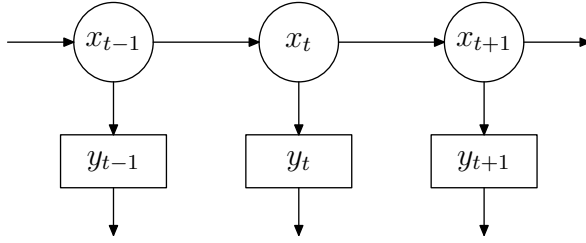
$$p(y_t | \mathbb{X}_t, \mathbb{U}_t) = p(y_t | x_t, u_t), \quad (4.1b)$$

where it is assumed that only the present state of the system affects the measurements.

The relation between the underlying dynamic model and the measurements are illustrated in Figure 4.1. A tutorial on the subject of HMM can be found in [111].

### 4.1.2 State-Space Model

In a *state-space model*, a *state vector*,  $x_t$ , holds all information about the system, at time  $t$ , needed to determine its future behavior given the input. That is a Markov chain property



**Figure 4.1:** Illustration of how the dynamic model, represented with circles, and the measurements, represented with rectangles, interact in a hidden Markov model.

when the model is stochastic. In this respect a state is very powerful since a low dimensional vector,  $x_t$ , can be used to summarize an infinite system history.

How the state evolves over time is determined by the system dynamics. That is, the state at time  $t + 1$ , relates to the previous state, and the inputs to the system  $u_t$  and  $w_t$  through the relation

$$x_{t+1} = f(x_t, u_t, w_t), \quad (4.2a)$$

where  $u_t$  and  $w_t$  are both inputs but differ in their interpretation;  $u_t$  is a known input to the system, whereas  $w_t$ , usually referred to as *process noise*, is an unknown unpredictable input modeled as a stochastic process. The only unknown component in this is the process noise.

An observation of the system at time  $t$  is denoted  $y_t$ , and is a mapping of the state  $x_t$ , and the inputs  $u_t$  and  $e_t$ , where the latter is measurement noise, an unpredictable disturbance, modeled with a stochastic process. Generally, the measurement  $y_t$  can be related to  $x_t$ ,  $u_t$ , and  $w_t$  through the relation

$$0 = h(y_t, x_t, u_t, e_t). \quad (4.2b)$$

The functions  $f$  and  $h$  in (4.2) are both implicitly assumed to depend on the time  $t$  throughout this thesis, unless otherwise stated. One way to achieve this is to include  $t$  as an element in the state vector, or by explicitly adding time as an argument to the functions.

The model defined by (4.2) is very general. For simplicity it is often assumed that  $y_t$  can be solved for in (4.2b) and that  $e_t$  is additive. The resulting model,

$$x_{t+1} = f(x_t, u_t, w_t) \quad (4.3a)$$

$$y_t = h(x_t, u_t) + e_t, \quad (4.3b)$$

is still a very general *nonlinear* model that is powerful enough to model almost any system encountered in practice. If, for example, images are used as measurements it is usually impossible to solve for  $y_t$  in (4.2b), however in this thesis, (4.3) is the most general model used in the further discussions.

The hidden Markov model has much in common with state-space models. In the HMM, the Markov process (4.1a), which cannot be observed directly, corresponds to the description of the dynamics in the state space formulation (4.2a). The observations, described with the stochastic process (4.1b), have the same purpose as the measurement

equation (4.2b). Regular systems can be represented both as a HMM and as a state-space model. This will be exemplified for some of the special cases of models in the next section.

## 4.2 Specific State-Space Model

The models presented above are very flexible and can describe basically any system encountered with good precision. However, the free mathematical representation makes it difficult to design algorithms to handle it. This section presents three important special cases of models: the linear model, the switched model, and a model with linear Gaussian substructure. The presentation is mainly based on the state-space representation of the models.

### 4.2.1 Linear Model

A very important special case of models are the *linear model*. Linearity makes model and algorithm analysis much easier, especially if combined with an assumption of Gaussian noise. Linear models are thoroughly treated in literature, e.g., [69, 121]. The linear assumption is that the dynamics and the measurement relation are both linear functions of the state,

$$x_{t+1} = F_t x_t + G_t^u u_t + G_t^w w_t \quad (4.4a)$$

$$y_t = H_t x_t + H_t^u u_t + e_t, \quad (4.4b)$$

where  $F_t$ ,  $G_t^u$ ,  $G_t^w$ ,  $H_t$ , and  $H_t^u$  are matrices of suitable dimensions. With the additional assumption that  $w_t$  and  $e_t$  are both Gaussian the model is linear Gaussian.

Many physical systems are linear systems because everyday physical phenomena can be described by linear ordinary differential equations, which assuming constant input between sampling times become a linear model. Some discussions about this, and about how to handle noise when discretizing a continuous model can be found in [44].

---

#### Example 4.1: Constant velocity model, state-space form

---

The *constant velocity* (CV) model describes linear motion with constant velocity disturbed by external forces, process noise  $w_t$ , entering the system in terms of acceleration [92]. The constant velocity model is for instance used to track airplanes, where the deterministic but unknown maneuvers by the pilot can be treated as process noise.

In one dimension, using measurements of the position and a sampling time  $T$ , the model is given by

$$\begin{aligned} x_{t+1} &= \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} \frac{1}{2}T^2 \\ T \end{pmatrix} w_t \\ y_t &= (1 \quad 0) x_t + e_t, \end{aligned}$$

where the states are position,  $x$ , and velocity,  $v$ , stacked as  $x = \begin{pmatrix} x \\ v \end{pmatrix}$ . Furthermore,  $w_t$  and  $e_t$  must be described for a complete model. A common assumption is *white* noise (independent in time), Gaussian, and mutually independent noise, e.g.,  $w_t \sim \mathcal{N}(0, Q)$  and  $e_t \sim \mathcal{N}(0, R)$ .

To extend the model to handle multi-dimensional constant velocity motion, treat each dimension separately and stack the states describing the motion in each dimension in the state vector.

In the HMM framework the linear model is described by the two distributions

$$p_{\bar{w}_t}(x_{t+1} - F_t x_t - G_t^u u_t | x_t, u_t) \quad (4.5)$$

$$p_{e_t}(y_t - H_t x_t - H_t^u u_t | x_t, u_t), \quad (4.6)$$

where the matrices  $F_t$ ,  $G_t^u$ ,  $G_t^w$ ,  $H$ , and  $H_t^u$  are the same as in (4.4) and  $\bar{W}_t = G_t^w w_t$ . If the model is linear Gaussian, both the densities in (4.5) are Gaussian. To exemplify this, the constant velocity model in Example 4.1 will be given as an HMM.

**Example 4.2: Constant velocity model, HMM form**

The constant velocity model in Example 4.1 is given as an HMM by the two densities

$$p_{G^w w_t}(x_{t+1} - F x_t | x_t) \longrightarrow \mathcal{N}(x_{t+1}; F x_t, G^w Q G^{wT})$$

$$p_{e_t}(y_t - H x_t | x_t) \longrightarrow \mathcal{N}(y_t; H x_t, R),$$

where

$$F = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \quad G^w = \begin{pmatrix} \frac{1}{2} T^2 \\ T \end{pmatrix} \quad H = (1 \quad 0),$$

and the distributions of  $w_t \sim \mathcal{N}(0, Q)$  and  $e_t \sim \mathcal{N}(0, R)$  are the same as in the previous example.

For applications where measurements are considered in batches, typically for detecting faults and similar, it is convenient to let the model reflect this. Therefore, introduce the following notation for  $L$  stacked measurements,

$$\mathbb{Y}_t^L = \begin{pmatrix} y_{t-L+1} \\ y_{t-L+2} \\ \vdots \\ y_t \end{pmatrix},$$

where the superscript  $L$  indicates that only the last  $L$  time samples are included instead of all available as in  $\mathbb{Y}_t$ . The same notation will be used throughout the thesis to represent other stacked variables, *i.e.*,  $\mathbb{W}_t^L$ ,  $\mathbb{E}_t^L$ , and  $\mathbb{U}_t^L$  will be used for  $L$  stacked  $w$ ,  $e$ , and  $u$ , respectively. Using stacked variables, it is possible to express the  $L$  measurements in a batch from the linear model (4.4) as

$$\mathbb{Y}_t = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^w \mathbb{W}_t^L + \mathbb{E}_t^L + \bar{H}_t^u \mathbb{U}_t^L, \quad (4.7)$$

where  $\mathcal{O}_t$  is the *extended observability matrix* that describes the impact of the initial state on the measurements, and the  $\bar{H}_t^*$  matrices describe how  $\star \in \{w, u, f\}$  enters the

measurements. A bar is used to separate these matrices from those used in the standard linear model. The extended observability matrix is

$$\mathcal{O} = \begin{pmatrix} H_{t-L+1} \\ H_{t-L+2} F_{t-L+1} \\ \vdots \\ H_t \prod_{i=t-1}^{t-L+1} F_i \end{pmatrix} = \text{/if time-invariant/} = \begin{pmatrix} H \\ HF \\ \vdots \\ HF^{L-1} \end{pmatrix}, \quad (4.8a)$$

where an assumption about time-invariance simplifies the expression considerably. The input matrices are defined as

$$\begin{aligned} \bar{H}^* &= \begin{pmatrix} H_{t-L+1}^* & 0 & \dots & 0 \\ H_{t-L+2} G_{t-L+1}^* & H_{t-L+2}^* & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ H_t \prod_{i=t-1}^{t-L+2} F_i G_{t-L+1}^* & H_t \prod_{i=t-1}^{t-L+3} F_i G_{t-L+2}^* & \dots & H_t^* \end{pmatrix} \\ &= \text{/if time-invariant/} = \begin{pmatrix} H^* & 0 & \dots & 0 \\ HG^* & H^* & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ HF^{L-2} G^* & HF^{L-3} G^* & \dots & H^* \end{pmatrix}, \quad (4.8b) \end{aligned}$$

for  $\star \in \{w, u, f\}$ . Note, if the system is time-invariant the  $H^*$  are *Toeplitz matrices* which allows for improved numerical algorithms to be used. For a more complete description of this way to view the system see e.g., [44, 46, 141].

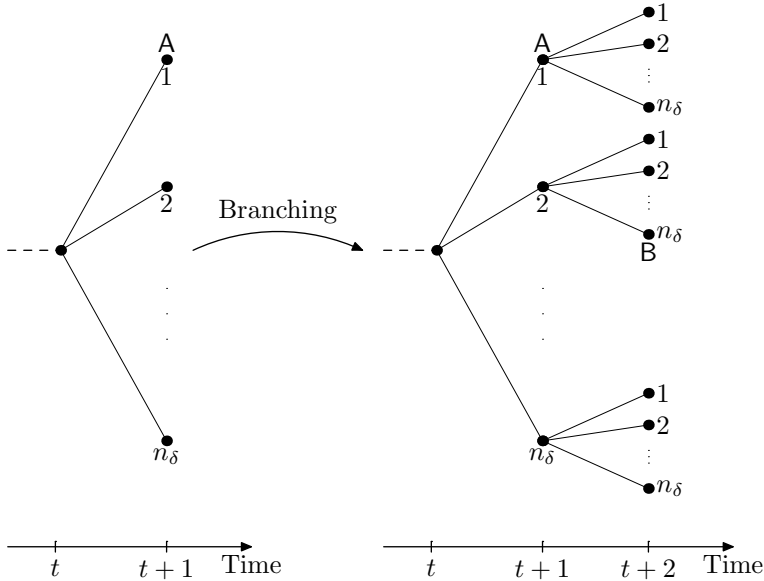
## 4.2.2 Switched Models

It is sometimes useful to have a model with different behaviors depending on the situation, the operational *mode*. As an example, an aircraft may behave in one way in one flight envelope and another way in another flight envelope. This can be described with a special case of the general model where one of the states is separated from the rest and used to indicate the mode of the system,

$$x_{t+1} = f(x_t, w_t, \delta_t) \quad (4.9a)$$

$$y_t = h(x_t, e_t, \delta_t), \quad (4.9b)$$

where  $\delta_t$  is used to indicate the mode at time  $t$ . In the HMM formulation,  $\delta$  is added as another conditional variable. If the switching between the models is indeterministic a model must be added, let  $p_{\delta|\delta'} := p(\delta|\delta')$  denote the probability that the model changes from model  $\delta'$  to  $\delta$ . This probability could depend both on the time and the state of the system. Furthermore, it should be assumed that  $w_t$  and  $e_t$  could both be affected by the active mode. How the system evolves over time is determined by which modes it has visited. Hence, introduce  $\delta_t$ , in analogy with  $\mathbb{Y}_t$ , to hold the mode history. If there are  $n_\delta$  different possible modes at each sample time this yields  $n_\delta^L$  possible different mode combinations



**Figure 4.2:** Illustration of the exponential increase of possible mode combinations. Each node represents one possible mode combination. The nodes marked A indicate  $\delta_{t+1} = (\delta_t, \delta_{t+1} = 1)$  and the B node  $\delta_{t+2} = (\delta_t, \delta_{t+1} = 2, \delta_{t+2} = n_\delta)$ .

to keep track of over a window of  $L$  samples. Thus, the mode complexity increases exponentially over time. The basic branching idea and the exponentially increasing number of possible nodes are illustrated in Figure 4.2.

For a predetermined, or exactly computable, mode sequence, this differs little from any other model with different behavior at different times. On the other hand, if the mode is stochastic the situation is slightly different in that all different modes must be considered, with associated probabilities, at all times,

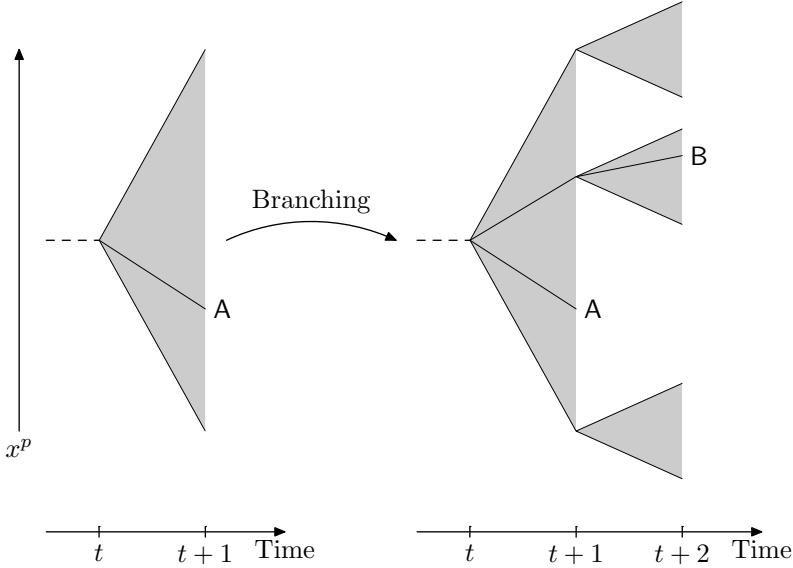
$$p(x_t | \mathbb{Y}_t) = \sum_{\delta_t} p(x_t | \mathbb{Y}_t) p(\delta_t | \mathbb{Y}_t). \quad (4.10)$$

Unless the model has such a structure that modes can be merged to avoid exponential branch growth, it is impossible to handle these models exactly other than in theory. In practice it is therefore necessary to use different reduction schemes where very similar nodes are merged and/or unlikely branches are pruned when using these models for more than a few time steps at the time.

It is an interesting observation that a general model with Gaussian sum noise can be expressed as a switched system where the only difference between the different models is the noise, and the probability of the modes is independent of the previous mode.

### 4.2.3 Model with Linear Gaussian Substructure

The model with *linear Gaussian substructure* presented in this section has much in common with the switched model presented in Section 4.2.2. Conditioned on one part of the



**Figure 4.3:** Illustration of how  $x^p$  can be seen as a continuous mode. A indicates the system in with one possible  $x_t^p$  state, and B the system with another  $x_{t+1}^p$  achieved after another  $x_t^p$ .

state space the remaining part of the model is linear and Gaussian. Viewing the part of the state space that is used to condition with as a mode, the similarity with the switched model is apparent. See Figure 4.3 for an illustration. The difference is that models must be linear Gaussian, and that the mode in this case does not make up a countable set. This class of models proves to be an important special case where filtering can be done more efficiently.

Assume that the part of the state space used to condition on is  $x^p$  and that the remaining part is  $x^k$ , organized in the state vector as  $x = \begin{pmatrix} x^p \\ x^k \end{pmatrix}$ . A model with these properties can be expressed in state space form as

$$x_{t+1}^p = f^p(x_t^p) + F^p(x_t^p)x_t^k + G^p(x_t^p)w_t^p \quad (4.11a)$$

$$x_{t+1}^k = f^k(x_t^p) + F^k(x_t^p)x_t^k + G^k(x_t^p)w_t^k \quad (4.11b)$$

$$y_t = h(x_t^p) + H^y(x_t^p)x_t^k + e_t, \quad (4.11c)$$

with  $w_t^p \sim \mathcal{N}(0, Q^p)$ ,  $w_t^k \sim \mathcal{N}(0, Q^k)$ , and  $e_t \sim \mathcal{N}(0, R)$ . It will be assumed that these are all mutually independent and independent in time. If  $w_t^p$  and  $w_t^k$  are not mutually independent this can be taken care of with a linear transformation of the system, which will preserve the structure, see [104] for details.

To simplify the notation, the following notation will be used:

$$x_{t+1} = F_t(x_t^p)x_t + f(x_t^p) + G_t(x_t^p)w_t \quad (4.12a)$$

$$y_t = H_t(x_t^p)x_t + h(x_t^p) + e_t, \quad (4.12b)$$

where

$$\begin{aligned} F_t(x_t^p) &= \begin{pmatrix} 0 & F^p(x_t^p) \\ 0 & F^k(x_t^p) \end{pmatrix} & f(x_t^p) &= \begin{pmatrix} f^p(x_t^p) \\ f^k(x_t^p) \end{pmatrix} \\ G_t(x_t^p) &= \begin{pmatrix} G^p(x_t^p) & 0 \\ 0 & G^k(x_t^p) \end{pmatrix} & Q &= \begin{pmatrix} Q^p & 0 \\ 0 & Q^k \end{pmatrix} \\ H_t(x_t^p) &= (0 \quad H^y(x_t^p)). \end{aligned}$$

The notation will be further shortened by dropping  $(x_t^p)$ , if this can be done without risking the clarity of the presentation.

For the HMM formulation it is easy to verify  $p(x_{t+1}^p | x_{t+1}^k, x_t) = p(x_{t+1}^p | x_t)$  and  $p(x_{t+1}^k | x_{t+1}^p, x_t) = p(x_{t+1}^k | x_t)$ , and that  $p(x_{t+1}^k | x_t^p)$ ,  $p(x_{t+1}^p | x_t^k)$ , and  $p(y_t | x_t)$  are Gaussian conditioned on  $x_t^p$ .

### 4.3 Fault Model

To be able to detect or diagnose changes or faults in a system it is often necessary to introduce the change into the model used to describe the system. This is usually done by introducing another input to the system in a way similar to how a known input enters it. The extra term will in this thesis be denoted  $f_t$  and is assumed to be deterministic but unknown. With a fault term, the general state-space formulation becomes

$$x_{t+1} = f(x_t, u_t, f_t, w_t) \quad (4.13a)$$

$$y_t = h(x_t, u_t, f_t) + e_t, \quad (4.13b)$$

and the linear state-space model

$$x_{t+1} = F_t x_t + G_t^u u_t + G_t^f f_t + G_t^w w_t \quad (4.14a)$$

$$y_t = H_t x_t + H_t^u u_t + H_t^f f_t + e_t. \quad (4.14b)$$

The convention is to use the same notation for changes/faults as for input signals with  $u$  exchanged for  $f$  when applicable, e.g.,  $G_t^u \rightarrow G_t^f$  as indicated in (4.14).

It will be assumed that for the nominal fault-free system  $f_t \equiv 0$ . Note that it is always possible to re-parametrize a given model in such a way that  $f_t \equiv 0$  in absence of fault.

Faults often manifest themselves in the measurements in the same way as process noise and/or measurement noise do. This introduces a difficulty because the effects of noise and faults are indistinguishable at any given time. The difference lies in the temporal behavior. Compared to noise, faults have additional structure in how they affect the system. By imposing this structure to the estimated faults it is possible to separate between noise and fault.

A fault can be split into two factors; direction and magnitude. Here the fault direction is contained in the matrices  $G_t^f$  and  $H_t^f$ , assuming a linear model. The remaining magnitude is then expressed as a linear regression,

$$f_t = \phi_t^T \theta. \quad (4.15)$$

The parameter,  $\theta$ , is a *time-invariant* fault parameter of dimension  $n_\theta$ . The basis,  $\phi_t$ , should be chosen carefully to cover all faults to be detected, and at the same time be kept as small as possible to improve detectability. Typically,  $n_\theta$  is small compared to the number of measurement used to detect a fault. Furthermore, with an orthonormal basis over a window  $\mathbb{F}_t^L = \{f_i\}_{i=t-L+1}^t$ , the energy in the fault is preserved in the fault parameter,

$$\|\mathbb{F}_t^L\|^2 = \|\Phi_t^{LT}\theta\|^2 = \|\theta\|^2,$$

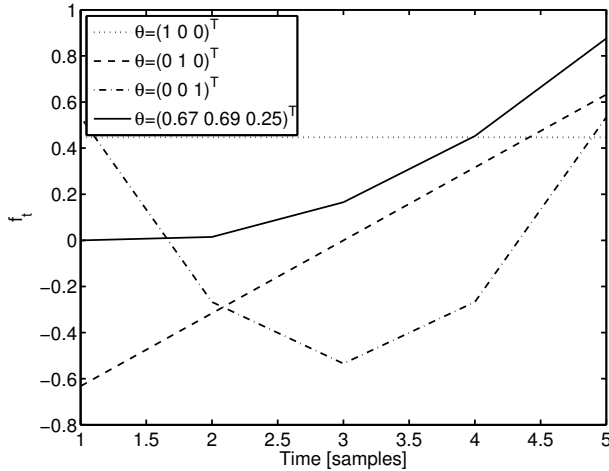
where  $\Phi_t^L := (\phi_{t-L+1} \dots \phi_t)$ . One such suitable choice for the basis is discrete *Chebyshev polynomials*, which describe orthogonal polynomials of increasing degree [1, 116]. Modeling faults in this way is an approach used in e.g., [54, 55, 141].

---

**Example 4.3: Incipient noise**

---

Assume that a window of  $L = 5$  samples is studied and that the first three discrete Chebyshev polynomials are used as basis,  $n_\theta = 3$ , for faults in the window. The resulting basis vectors and an example of an incipient fault is depicted in Figure 4.4.



**Figure 4.4:** The three first Chebyshev polynomials and an incipient fault described using them.

---

# 5

---

## Filtering Methods

**I**T IS NOW TIME to try to determine the position of the object in the bearings-only tracking problem that was introduced in Chapter 1. To do this the model derived in Chapter 3 will be used to relate the position to the measurements and Chapter 2 helps describing the noisy measurements. Based on Figure 1.1 the following initial assumptions are made about the problem:

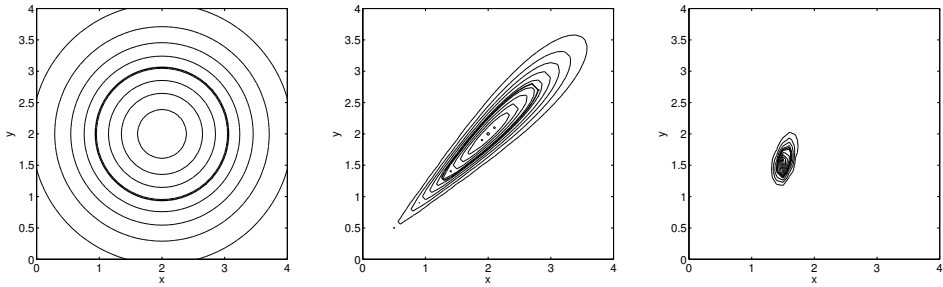
- The true object position for this realization is  $x^0 = \begin{pmatrix} 1.5 \\ 1.5 \end{pmatrix}$ .
- The initial position estimate is  $\hat{x}_0 \sim \mathcal{N}\left(\begin{pmatrix} 2 \\ 2 \end{pmatrix}, I\right)$ , as illustrated in Figure 5.1(a).
- The sensors are positioned in  $\mathcal{S}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$  and  $\mathcal{S}_2 = \begin{pmatrix} 1.5 \\ 0 \end{pmatrix}$ ,
- and the measurements are affected by the additive noise  $e_i \sim \mathcal{N}(0, 10^{-3}\pi)$ , *i.e.*, a standard deviation of approximately  $3.2^\circ$ .

Making a measurement (assuming the realization  $e_1 = 0$ ) from  $\mathcal{S}_1$  and incorporating it into the initial estimate yields  $p(x|y_1)$ , as shown in Figure 5.1(b). It is clearly visible how the measurement limits the possible object positions. A measurement from  $\mathcal{S}_2$  (again  $e_2 = 0$ ) gives the posterior distribution  $p(x|y_1, y_2)$  in Figure 5.1(c), which further limits the possible position.

This chapter presents several different methods to utilize available sensor information for estimation. First estimation in general is introduced in Section 5.1, and after that state estimation in particular. The problem is to find one or both of the posterior distributions  $p(x_t|\mathbb{Y}_t)$  and  $p(x_{t+1}|\mathbb{Y}_t)$  representing filtering and prediction, respectively. These distributions hold all information about the state that is available, based on a model assumption. The general state-space model (4.2),

$$x_{t+1} = f(x_t, w_t) \tag{5.1a}$$

$$y_t = h(x_t) + e_t, \tag{5.1b}$$



(a) Initial knowledge, no measurements.

(b) Knowledge after one measurement from  $\mathcal{S}_1$ .

(c) Knowledge after measurements from  $\mathcal{S}_1$  and  $\mathcal{S}_2$ .

**Figure 5.1:** Analytic inferred information about  $x$  as given for no measurements, one measurement from  $\mathcal{S}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ , and one from  $\mathcal{S}_2 = \begin{pmatrix} 1.5 \\ 0 \end{pmatrix}$ . The thick lines indicate the 47% confidence regions. Refer to Figure 1.1 for a schematic illustration of the setup.

where the deterministic input  $u_t$  has been removed in favor of a clearer notation, is one alternative.

In most cases, an analytic solution to the filtering problem is unavailable. The *Kalman filter* (KF) that solves the linear Gaussian problem is an important exception to this, in most other cases approximations are necessary. This chapter presents two main approximate approaches: stochastic methods represented by *sequential Monte Carlo methods*, or *particle filters* (PF), introduced in Section 5.2; and deterministic methods represented by the Kalman filter, Section 5.3, and its extensions to nonlinear systems in Section 5.4. The filter bank in Section 5.5 utilizes the structure in switched models using one filter for each mode. Section 5.6 exploits a linear Gaussian substructure for more effective filtering in the *Rao-Blackwellized particle filter* (RBPF), which is a combination of a Kalman filter bank and a particle filter.

## 5.1 Estimation Preliminaries

This section gives some estimation preliminaries before describing the filtering techniques used in this thesis.

### 5.1.1 Parameter Estimation

General parameter estimation techniques are covered thoroughly by most textbooks on estimation theory, e.g., [76, 89]. Given a set of measurements  $\mathbb{Y}_t$  related to a parameter  $x$  through

$$y_i = h(x) + e_i, \quad (5.2)$$

where  $e_i$  is noise, determine the posterior distribution  $p(x|\mathbb{Y}_t)$  and find the  $\hat{x}$  that minimizes a given *loss function*, or in a Bayesian setting a *cost function*,  $L(x, \mathbb{Y}_t)$ ,

$$\hat{x} := \arg \min_{\hat{x}} L(\hat{x}, \mathbb{Y}). \quad (5.3)$$

Two properties of an estimate are *bias* and (*co*)*variance*. The bias is defined as the difference between the expected value of the estimate and the true parameter value,

$$\mathbb{E}(\hat{x}) = x^0 + b, \quad (5.4)$$

where  $x^0$  is the true state and  $b$  the bias. An estimator such that  $b \equiv 0$  is referred to as an *unbiased* estimator. If  $b \neq 0$  then the estimate is *biased* with bias  $b$ . Unbiasedness is an important property of estimators in classical statistics, but of little interest in the Bayesian framework where the parameter distribution is the main objective and where the concept of a true parameter value does not exist, [117]. The covariance of the estimate,  $\text{cov}(\hat{x})$ , gives an indication of how much the estimate varies depending on the measurements. The *mean square error* (MSE) is connected to the bias and covariance as

$$\lim_{M \rightarrow +\infty} \frac{1}{M} \sum_{i=1}^M (\hat{x} - x^0)^2 = \text{cov}(\hat{x}) + b^2. \quad (5.5)$$

The estimate of an unbiased estimator is under weak regularity conditions bounded by the *Cramér-Rao lower bound* (CRLB),

$$\text{cov}(\hat{x}) \succeq P^{\text{CRLB}},$$

where the CRLB is given in Section 2.2.1 as the inverse Fisher information. An estimator that reaches the CRLB is said to be *efficient*. The unbiased estimator that achieves the lowest  $\text{cov}(\hat{x})$  is called the *minimum variance estimator* (MVE). The *best linear unbiased estimator* (BLUE) is the linear estimator that has the lowest error variance.

There are many estimation methods available, four important methods are listed below:

- *Least square* (LS) estimation, where the squared errors between the estimated and the actual measurements are minimized,

$$\hat{x}^{\text{LS}} := \arg \min_{\hat{x}} \sum_{i=1}^t \|y_i - h(\hat{x})\|_2^2. \quad (5.6a)$$

- *Maximum likelihood* (ML) estimation, where the estimate is chosen to be the parameter most likely to produce the measurements obtained,

$$\hat{x}^{\text{ML}} := \arg \max_{\hat{x}} p(\mathbb{Y}_t | \hat{x}). \quad (5.6b)$$

The maximum likelihood estimate is optimal in the sense that it is unbiased and reaches the CRLB as the information in the measurements goes to infinity. However, the maximum likelihood estimate is generally not unbiased for finite information.

- *Minimum mean square error* (MMSE) estimation, which basically is the Bayesian version of least square estimation,

$$\hat{x}^{\text{MMSE}} := \arg \min_{\hat{x}} \mathbf{E}_{x, \mathbb{Y}_t} (x - \hat{x})^2 = \mathbf{E}_{x | \mathbb{Y}_t} (x), \quad (5.6c)$$

where the last equality follows from performing the optimization.

- *Maximum a posteriori* (MAP) estimation, the Bayesian equivalent of maximum likelihood estimation, is defined as the most likely  $x$  given the measurements based on the *a priori* distribution and the measurements,

$$\hat{x}^{\text{MAP}} := \arg \max_{\hat{x}} p(\hat{x} | \mathbb{Y}_t). \quad (5.6d)$$

### 5.1.2 Dynamic Estimation

When working with a time-varying state the distribution of interest is

$$p(x_t | \mathbb{Y}_\tau). \quad (5.7)$$

The point estimate  $\hat{x}_{t|\tau}$  is used to indicate the state estimate of  $x_t$  is based on the measurements  $\mathbb{Y}_\tau$ . If  $t > \tau$  this is a prediction problem, if  $t = \tau$  a filtering problem, and if  $t < \tau$  a smoothing problem. Statistical inference gives the Bayesian filtering equations [64]. The prediction distribution follows from marginalization,

$$p(x_{t+1} | \mathbb{Y}_t) = \int p(x_{t+1} | x_t) p(x_t | \mathbb{Y}_t) dx_t, \quad (5.8a)$$

where the integration is performed over the entire state space of  $x_t$ . Information acquired from new measurements is incorporated into knowledge about the state using Bayes' rule,

$$p(x_t | \mathbb{Y}_t) = \frac{p(y_t | x_t) p(x_t | \mathbb{Y}_{t-1})}{\int p(y_t | x_t) p(x_t | \mathbb{Y}_{t-1}) dx_t} = \frac{p(y_t | x_t) p(x_t | \mathbb{Y}_{t-1})}{p(y_t | \mathbb{Y}_{t-1})}. \quad (5.8b)$$

Given  $p(x_t | \mathbb{Y}_\tau)$  the same principal estimators as in the static case (5.6) can be used to obtain the point estimate  $\hat{x}_{t|\tau}$  and the same estimator properties apply as for parameter estimation.

## 5.2 Particle Filter

When an analytic solution is unavailable numeric approximations of (5.8) are necessary. The main problem with (5.8) is to evaluate the integrals. One way to handle this is to deterministically grid the state space and approximate the PDFs with piecewise constant functions so that the integrals can be treated as sums that are easier to handle. This grid based approach, also called the *point-mass filter* (PMF), is described in depth in [25, 64, 84] and evaluated in [15, 83]. Another method, which has much in common with filter banks (discussed in Section 5.5), is to use a Gaussian sum approximation [3, 132]. However, these methods suffer heavily from the curse of dimensionality. In practice this

drastically limits the usability of the methods mentioned above. An alternative is to resort to stochastic methods, such as *Monte Carlo integration*, leading to *sequential Monte Carlo* filters, also known as *particle filters* (PF).

The foundation for the particle filter was laid in the 1950s [50], but the technique never came into use at that time. Reasons for this may include the lack of computing power needed to fully take advantage of the method, and that the fundamental resampling step, introduced later by Gordon, Salmond, and Smith [42], was not yet used. Since the seminal paper [42] much has been written about the particle filter and its favorable properties, as well as applications using it. The particle filtering theory is described in [32, 33, 115]. Interesting variations of the theory that tries to improve the proposal densities are: the *auxiliary particle filter*, [110] and the *unscented particle filter*, [122, 143], which try to improve the proposal distribution using an auxiliary variable and a UKF step, respectively; and the *Gaussian particle filter*, [81], which approximates the posterior distributions with a Gaussian, and the *Gaussian sum particle filter*, [82], which is a filter bank of Gaussian particle filters. Different applications of the particle filter can be found in [16, 48, 72].

### 5.2.1 Approximate Probability Density Function

Based on the Monte Carlo integration technique and Monte Carlo transformation, described in Section 3.2, a reasonable approximation of the interesting PDF is to use a set of  $N \gg 1$  particles  $\{x_t^{(i)}\}_{i=1}^N$ , with associated weights  $\{\omega_{t|t-1}^{(i)}\}_{i=1}^N$ , such that

$$p(x_t | \mathbb{Y}_{t-1}) \approx \sum_{i=1}^N \omega_{t|t-1}^{(i)} \delta(x_t - x_t^{(i)}),$$

where the particles are IID samples from a proposal distribution and the weights are matching importance weights. Properties such as the mean of  $x_t$ , the MMSE estimate, is then easily computed using Monte Carlo integration,

$$\hat{x}_{t|t-1} = \mathbb{E}_{x_t | \mathbb{Y}_{t-1}}(x_t) \approx \sum_{i=1}^N \omega_{t|t-1}^{(i)} x_t^{(i)}.$$

Now, assume that the measurement  $y_t$  is obtained, the updated PDF is then according to (5.8b)

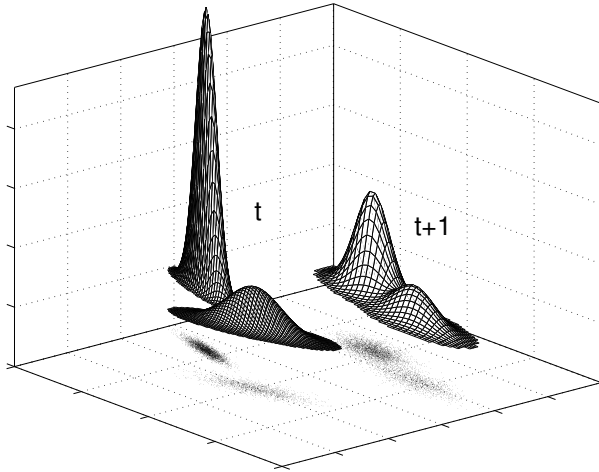
$$p(x_t | \mathbb{Y}_t) \propto p(y_t | x_t) p(x_t | \mathbb{Y}_{t-1}),$$

leading to the importance weight update

$$\omega_{t|t}^{(i)} = \frac{p(y_t | x_t^{(i)}) \omega_{t|t-1}^{(i)}}{\sum_j p(y_t | x_t^{(j)}) \omega_{t|t-1}^{(j)}}.$$

Using these updated weights the measurement updated PDF becomes,

$$p(x_t | \mathbb{Y}_t) \approx \sum_{i=1}^N \omega_{t|t}^{(i)} \delta(x_t - x_t^{(i)}).$$



**Figure 5.2:** PDFs for, and particles describing, the state distribution before (left) and after (right) the prediction step in a particle filter.

The following PDF is constructed for the prediction, (5.8a),

$$p(x_{t+1}|\mathbb{Y}_t) = \int p(x_{t+1}|x_t)p(x_t|\mathbb{Y}_t) dx_t.$$

To do this, draw a new set of IID particles to represent the updated PDF, sample  $x_{t+1}^{(i)}$  from an proposal distribution  $q(x_{t+1}|x_t^{(i)}, \mathbb{Y}_t)$  and update the importance weights accordingly,

$$\omega_{t+1|t}^{(i)} = \frac{p(x_{t+1}^{(i)}|x_t^{(i)})}{q(x_{t+1}^{(i)}|x_t^{(i)}, \mathbb{Y}_t)} \omega_{t|t}^{(i)},$$

yielding the approximate prediction PDF

$$p(x_{t+1}|\mathbb{Y}_t) \approx \sum_{i=1}^N \omega_{t+1|t}^{(i)} \delta(x_{t+1} - x_{t+1}^{(i)}).$$

Figure 5.2 illustrates a particle cloud before and after a prediction step. Note how the cloud is translated by the transfer function  $f$  and how the process noise spreads the particles. Compare this illustration to Figure 1.3(b) in the introduction where the effects of a Gaussian approximation are illustrated.

Note that if the proposal distribution is chosen to be  $q(x_{t+1}|x_t, \mathbb{Y}_t) = p(x_{t+1}|x_t)$  this simplifies the update of the importance weights to  $\omega_{t+1|t}^{(i)} = \omega_{t|t}^{(i)}$ . However, this is not optimal with respect the statistical properties of the filter. Nevertheless, due to its simplicity the simple proposal is often used.

The above is a sequential Monte Carlo method and represents what was available in the 1950s [50]. It can be shown that using this method the approximated distribution will degenerate so that only a few particles actually contribute to the description of the

**Algorithm 5.1** Particle Filter

1. Initiate the filter:  $\{x_0^{(i)}\}_{i=1}^N \sim p_{x_0}$  and  $\{\omega_{0|-1}^{(i)}\}_{i=1}^N = \frac{1}{N}$  for  $i = 1, \dots, N$ . Let  $t := 0$ .

2. Measurement update phase:

$$\omega_{t|t}^{(i)} = \frac{p(y_t|x_t^{(i)})\omega_t^{(i)}}{\sum_j p(y_t|x_t^{(j)})\omega_t^{(j)}}, \quad i = 1, \dots, N.$$

3. Resample! See Algorithms 5.2 and 5.3 in Section 5.2.2, for the resampling steps used in the SIR and SIS particle filter.

4. Time update phase: Generate  $N$  IID samples from an importance distribution  $q(\cdot)$ ,  $\{x_{t+1|t}^{(i)}\}_{i=1}^N \sim q(x_{t+1}|x_t^{(i)}, \mathbb{Y}_t)$  and update the importance weights accordingly,

$$\omega_{t+1|t}^{(i)} = \omega_{t|t}^{(i)} \frac{p(x_{t+1|t}^{(i)}|x_t^{(i)})}{q(x_{t+1|t}^{(i)}|x_t^{(i)}, \mathbb{Y}_t)}, \quad i = 1, \dots, N.$$

If  $q(x_{t+1|t}^{(i)}|x_t^{(i)}, \mathbb{Y}_t) = p(x_{t+1|t}^{(i)})$  this simplifies to  $\omega_{t+1|t}^{(i)} = \omega_{t|t}^{(i)}$ .

5. Let  $t := t + 1$  and repeat from 2.

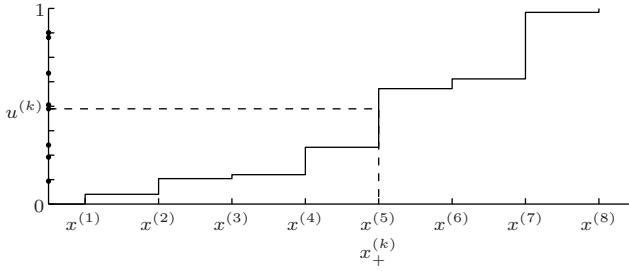
PDF [32, 80]. Using  $q(x_{t+1}|x_t, \mathbb{Y}_t) = p(x_{t+1}|x_t)$  this can happen quite quickly, whereas a more careful choice of importance distribution may at best slow down the process. A solution to the problem is to introduce a *resampling* step as suggested in [42]. The resampling step in combination with the increased computational power was what was needed to turn the particle filter into an interesting method. Algorithm 5.1 represents a generic particle filter with resampling.

At any time in the algorithm a minimum variance estimate of the state can be obtained as a weighted sample mean

$$\hat{x}_{t|\tau} = \sum_i \omega_{t|\tau}^{(i)} x_t^{(i)}, \quad (5.10a)$$

and the covariance of the estimate is given by the sample covariance

$$P_{t|\tau} = \sum_i \omega_{t|\tau}^{(i)} (x_t^{(i)} - \hat{x}_{t|\tau})(x_t^{(i)} - \hat{x}_{t|\tau})^T. \quad (5.10b)$$



**Figure 5.3:** Illustration of resampling using the generalised inverse CDF for the particles.

## 5.2.2 Resampling

The resampling step rejuvenates the particles used to represent the PDF in the particle filter so that the stochastic support is maintained. That is, given

$$p(x) \approx \sum_{i=1} \omega^{(i)} \delta(x - x^{(i)})$$

find an approximation

$$p_+(x) \approx \sum_{i=1} \omega_+^{(i)} \delta(x - x_+^{(i)})$$

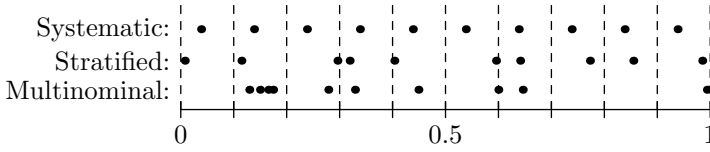
with better support. Usually,  $\omega_+^{(i)} \equiv \frac{1}{N}$  and  $x_+^{(i)}$  are selected from the original set of particles  $x^{(i)}$ . Several methods have been devised to do this. Four different resampling algorithms are compared and contrasted in [61, 62]. The four methods are given in this section. Resampling complexity from a more hardware near point of view is studied in [22] and in a setting with a network of distributed computational nodes in [23].

The original particle filter, [42], used the *sampling importance resampling* (SIR) strategy, which is a special case of the more general *sequential importance sampling* (SIS), [32]. The difference between the two strategies is how often the actual resampling is performed.

Resampling is performed because the stochastic support of the particle set is lost since the particles lose importance over time. The resampling avoids this by randomly selecting new particles from the old set paying attention to how likely they are, *i.e.*, drawing new particles from the old ones with replacement and  $\Pr(x_+^{(j)} = x^{(i)}) = \omega^{(i)}$ . More mathematically, let  $x_+^{(j)} = x^{(i)}$  with

$$i = P^{-1}(u^{(j)}),$$

where  $u^{(j)}$  is a random number  $u^{(j)} \in [0, 1)$ , and  $P^{-1}$  is the generalized inverse CDF for the particles,  $P(i) = \sum_{j=1}^i \omega^{(j)}$ , as illustrated in Figure 5.3. Different sampling methods differ in how the  $u^{(j)}$  are generated.



**Figure 5.4:** Illustration of how  $u^{(j)}$  are chosen with multinomial, stratified, and systematic resampling, in this case with 10 particles.

**Multinomial resampling** The random numbers  $u^{(j)}$  are selected in a completely random fashion from a uniform distribution,

$$u^{(j)} \sim \mathcal{U}[0, 1]. \tag{5.11a}$$

This is the method used in the first papers about particle filtering. To perform the resampling efficiently  $u^{(j)}$  should be ordered, for which the look-up in  $P^{-1}$  can be efficiently implemented. An efficient algorithm to generate ordered uniformly distributed samples in linear time is given in [114], and a compact MATLAB® implementation of the algorithm in [15].

**Stratified resampling** The random numbers  $u^{(j)}$  are drawn as

$$u^{(j)} = \frac{(k - 1) + \tilde{u}^{(j)}}{N}, \quad \tilde{u}^{(j)} \sim \mathcal{U}[0, 1]. \tag{5.11b}$$

Generated in this way,  $u^{(j)}$  are always ordered and no extra measures need to be taken in order to ensure that. Furthermore, it guarantees that if the inverse CDF is split in  $N$  equally large parts, there will be one sample from each. This somewhat restricts the randomness as there is some structure in the  $u^{(j)}$  values, for instance there cannot be two  $u^{(j)} < 1/N$ . In [62] it is argued that this improves the result of the resampling.

**Systematic resampling** Where  $u^{(j)}$  are evenly space according to,

$$u^{(i)} = \frac{(k - 1) + \tilde{u}}{N}, \quad \tilde{u} \sim \mathcal{U}[0, 1]. \tag{5.11c}$$

That is, as with the stratified sampling, there is one particle selected from each segment of length  $1/N$ .

**Residual resampling** With this resampling, particle  $i$  should be chosen  $n^{(i)} = \lfloor N\omega^{(i)} \rfloor$  times, giving  $\underline{N} = \sum_{i=1}^N n^{(i)}$  new particles, the remaining  $N - \underline{N}$  particles are then chosen using another resampling algorithm. Choosing the first  $\underline{N}$  particles is very fast, and using this method could hence improve the computational complexity.

Figure 5.4 illustrates the different strategies that multinomial, stratified, and systematic resampling use to select the  $u^{(i)}$  that determine the new set of particles.

The difference between the SIR and SIS particle filter is how often resampling is conducted, the resampling itself can be done using any resampling algorithm but the original

**Algorithm 5.2** Sampling Importance Resampling (SIR) Filter

Use Algorithm 5.1 with the following resampling step:

3. Resample: Use e.g., multinomial, stratified, systematic, or residual resampling to get a new set of particles  $\{x_t^{(i)}\}_{i=1}^N$  and let  $\omega_{t|t}^{(i)} = 1/N$ .

**Algorithm 5.3** Sequential Importance Sampling (SIS) Filter

Use Algorithm 5.1 with the following resampling step:

3. Resample: Compute the effective sample size,

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^N (\omega_{t|t}^{(i)})^2}.$$

**If**  $\hat{N}_{\text{eff}} < N_{\text{th}}$

**then** construct a new set  $\{x_t^{(i)}\}_{i=1}^N$  using a resampling algorithm of choice and set  $\omega_{t|t}^{(i)} = 1/N$

**else** keep the current particles and weights.

formulations use multinomial resampling. When using SIR resampling (Algorithm 5.2) the particles are resampled every time the estimation loop is performed, whereas in the generalization, SIS (Algorithm 5.3), the particles are only resampled when it is necessary according to some measure. An often used measure of the particle quality is the *effective sample size* [15, 80],

$$N_{\text{eff}} = \frac{N}{1 + \text{var}(\omega^{(i)})},$$

but other measures have been suggested e.g., [133]. The effective sample size indicates how many of the particles actually contribute to the support of the studied PDF. If  $N_{\text{eff}} \ll N$  this indicates that the support is poor and that resampling is needed to avoid degeneration of the filter. Unfortunately,  $N_{\text{eff}}$  is difficult to calculate analytically, but can be approximated [80] with,

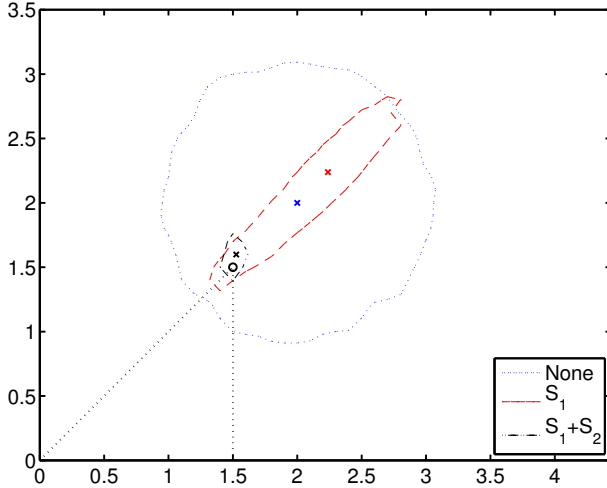
$$\hat{N}_{\text{eff}} = \frac{1}{\sum_i (\omega^{(i)})^2}.$$

Hence,  $\hat{N}_{\text{eff}}$  can be used to determine when to resample, i.e., given some threshold  $N_{\text{th}}$  resample when  $N_{\text{eff}} \approx \hat{N}_{\text{eff}} < N_{\text{th}}$ . In [15]  $N_{\text{th}} = \frac{2}{3}N$  is used as a threshold for resampling. The SIS particle filter is given in Algorithm 5.3.

— **Example 5.1: Bearings-only tracking — applying the particle filter** —

The bearings-only tracking problem that is described in the introduction of this chapter can be solved using a particle filter. Apply a standard SIR PF yields the result in Figure 5.5. The figure shows the MMSE estimates and matching 47% confidence regions

after smoothing the particle cloud with a small Gaussian kernel. The particle filter produces proper PDFs very similar to what is found in Figure 5.1. Note the unevenness that is a result of the approximate PDF representation.



**Figure 5.5:** Estimate and 47% confidence region for the particle filter applied to the bearings-only example with no, one, and two measurements. Compare this with Figure 5.1. (Estimates are denoted with  $\times$ , the true target position is denoted with  $\circ$ .)

### 5.3 Kalman Filter

Probably the most well-known estimation algorithm for linear systems,

$$x_{t+1} = F_t x_t + G_t^u u_t + G_t^w w_t, \tag{5.12a}$$

$$y_t = H_t x_t + H_t^u u_t + e_t \tag{5.12b}$$

(cf. Section 4.2.1), is the *Kalman filter*. The filter is named after Rudolph E. Kalman, who in 1960 published a famous paper introducing the method [70]. At that time others were independently working with similar methods; amongst other Swerling [136] and researchers from the USSR [131]. Nevertheless Kalman has over time received most of the credit for developing the filter. In a few decades the Kalman filter became widely used. A collection of important work from the first decades of Kalman filtering, both theoretical and application oriented, can be found in [131]. The original Kalman filter assumes a discrete-time linear model (4.4), but just a few years later the theory was extended to also include continuous-time systems [71].

The main idea of the Kalman filter is to use a linear filter to update the mean and the covariance of the estimate so that the covariance of the estimation error is minimized. If

---

**Algorithm 5.4** Kalman Filter
 

---

1. Initiate the filter with the initial information:

$$\hat{x}_{0|-1} = x_0 \quad \text{and} \quad P_{0|-1} = \Pi_0,$$

where  $x_0$  is the initial state estimate and  $\Pi_0 = \text{cov}(x_0)$ . Let  $t := 0$ .

2. Measurement update phase:

$$\begin{aligned} K_t &= P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1} \\ \hat{x}_{t|t} &= \hat{x}_{t|t-1} + K_t (y_t - H_t \hat{x}_{t|t-1} - H_t^u u_t) \\ P_{t|t} &= (I - K_t H_t) P_{t|t-1} \end{aligned}$$

3. Time update phase:

$$\begin{aligned} \hat{x}_{t+1|t} &= F_t \hat{x}_{t|t} + G_t^u u_t \\ P_{t+1|t} &= F_t P_{t|t} F_t^T + G_t^w Q_t G_t^{wT} \end{aligned}$$

4. Let  $t := t + 1$  and repeat from 2.
- 

all noises are Gaussian, and hence stays Gaussian even after linear transformations, the Kalman filter is efficient and provides the minimum variance solution to (5.8) [69]. If the noise is non-Gaussian, the Kalman filter is the BLUE, i.e., it has the smallest error covariance of all linear filters, but there may exist nonlinear estimators that are better.

The Kalman filter is often derived for Gaussian noise where the calculations are straightforward to perform and the optimality follows immediately. There are many good books on the subject how to derive the Kalman filter, e.g., the work by Anderson and Moore [4] and more recently the book by Kailath, Sayed, and Hassibi [69], therefore the filter is just given in Algorithm 5.4 without any derivation. The Kalman filter equations can be divided into a time update phase, where the dynamics of the system is handled, and a measurement update phase, where the measurements are incorporated in the estimate.

Algorithm 5.4 is a straightforward way to implement a Kalman filter. However, if the estimation problem is poorly stated it may result in numerical problems. One such issue is that the covariance matrices lose symmetry. Another problem is that the covariances become indefinite. Symmetry is easily checked for, but indefiniteness is more difficult. One solution to this problem is to use a *square-root implementation*, also called an *array implementation*, of the Kalman filter algorithm where the square-root of the covariance matrix is propagated. This guarantees both symmetry and positive definiteness. Square-root based Kalman filter algorithms are described in e.g., [69]. The same technique is also used for other methods derived from the Kalman filter, e.g., the unscented Kalman filter discussed in Section 5.4.4.

An alternative view of the Kalman filter is to focus more on the recursive equations for the filtering PDFs. Doing this, the filter can be described as propagating a stochastic

variables through the model equations and a compensation step to include information from measurements.

With this approach the time update step is to compute the distribution of

$$x_{t+1|t} = f(x_{t|t}, w_t) \sim \mathcal{N}(\hat{x}_{t+1|t}, P_{t+1|t}),$$

which for a linear Gaussian model is exact, and for a linear non-Gaussian model the Kalman filter time update if a first order Gaussian approximation is used to approximate the new stochastic variable.

The time update in the Kalman filter can be reformulated as

$$\begin{aligned} \hat{x}_{t|t} &= \hat{x}_{t|t-1} + P_{t|t-1}^{xy} P_{t|t-1}^{-yy} (y_t - \hat{y}_t) \\ P_{t|t-1} &= (I - P_{t|t-1}^{xy} P_{t|t-1}^{-yy} P_{t|t-1}^{yx}) P_{t|t-1}, \end{aligned}$$

where unknown quantities comes from the Gaussian approximation of the distribution

$$\begin{pmatrix} x_{t|t-1} \\ y_t \end{pmatrix} = \begin{pmatrix} x_{t|t-1} \\ h(x_{t|t-1}) + e_t \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \hat{x}_{t|t-1} \\ \hat{y}_t \end{pmatrix}, \begin{pmatrix} P_{t|t-1} & P_{t|t-1}^{xy} \\ P_{t|t-1}^{yx} & P_{t|t-1}^{yy} \end{pmatrix} \right).$$

The distributions should be derived using the first order Gaussian approximation, which for the linear Gaussian case is exactly the same as the analytic solution according to Section 3.3.

The resulting algorithm is presented in Algorithm 5.5. As will be seen, using this alternative form of the Kalman filter leads to natural extensions to more general systems.

## 5.4 Kalman Filters for Nonlinear Models

The Kalman filter, even though quite general, is in many cases not applicable due to its limitation to *linear models*. Many real life situations have either nonlinear dynamics or nonlinear measurement equation. Over time an approximative extension to the Kalman filter was developed, the *extended Kalman filter* (EKF) that uses linearized models to handle nonlinear models. Early examples of this are the works of Schmidt [124], Smith, Schmidt, and McGee [130]. An early reference where the EKF is thoroughly treated is Jazwinski [64]. The EKF is also treated in other standard references on filtering, e.g., [4, 69].

### 5.4.1 Linearized Kalman Filter

A first step towards a Kalman filter for nonlinear models is to assume that a nominal trajectory is known,  $x^{\text{nom}}$ , linearize the system around this trajectory (*cf.* Gaussian transformation of stochastic variables in Section 3.3) and then apply the Kalman filter to the linearized system. This is often called the *linearized Kalman filter* [69] and works in situations where a good nominal trajectory is known in advance, e.g., the trajectory of an orbiting satellite.

**Algorithm 5.5** General Kalman Filter

1. Initiate the filter with the initial information:

$$\hat{x}_{0|-1} = x_0 \quad \text{and} \quad P_{0|-1} = \Pi_0,$$

where  $x_0$  is the until state estimate and  $\Pi_0 = \text{cov}(x_0)$ . Let  $t := 0$ .

2. Measurement update phase:

$$\begin{aligned} \hat{x}_{t|t} &= \hat{x}_{t|t-1} + P_{t|t-1}^{xy} P_{t|t-1}^{-yy} (y_t - \hat{y}_t) \\ P_{t|t} &= P_{t|t-1} - P_{t|t-1}^{xy} P_{t|t-1}^{-yy} P_{t|t-1}^{yx}, \end{aligned}$$

where needed quantities comes from the transformed stochastic variable (cf. Chapter 3 for a discussion on different available methods)

$$\begin{pmatrix} x_{t|t-1} \\ y_t \end{pmatrix} = \begin{pmatrix} x_{t|t-1} \\ h(x_{t|t-1}, e_t) \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} \hat{x}_{t|t-1} \\ \hat{y}_t \end{pmatrix}, \begin{pmatrix} P_{t|t-1} & P_{t|t-1}^{xy} \\ P_{t|t-1}^{yx} & P_{t|t-1}^{yy} \end{pmatrix} \right).$$

If the model is not linear Gaussian, the resulting distribution is only approximately Gaussian.

3. Time update phase: Follows from the transformation (cf. Chapter 3 for a discussion on different available methods)

$$x_{t+1|t} = f(x_{t|t}, w_t) \sim \mathcal{N}(\hat{x}_{t+1|t}, P_{t+1|t}).$$

If the model is not linear Gaussian, the resulting distribution is only approximately Gaussian.

4. Let  $t := t + 1$  and repeat from 2.

The linear system resulting from linearization of (4.3) has the dynamics

$$\begin{aligned} x_{t+1} &= f(x_t, 0) \approx f(x_t^{\text{nom}}, 0) + F_t(x_t - x_t^{\text{nom}}) + G_t^w w_t \\ &= F_t x_t + \underbrace{f(x_t^{\text{nom}}, 0) - F_t x_t^{\text{nom}}}_{=: u_t^f} + G_t^w w_t = F_t x_t + u_t^f + G_t^w w_t, \end{aligned} \quad (5.15a)$$

without deterministic input, and assuming  $E(w_t) = 0$  and  $E(e_t) = 0$  from now on for notational clarity. The gradient

$$F_t := \left( \nabla_x f(x, 0) \Big|_{x=x_t^{\text{nom}}} \right)^T, \quad G_t := \left( \nabla_w f(x_t^{\text{nom}}, w) \Big|_{w=0} \right)^T,$$

and the artificial input  $u_t^f$  is known at design time, since the nominal trajectory  $x^{\text{nom}}$  is assumed known. Using the same technique, the measurement relation can be expressed

as

$$\begin{aligned} y_t &= h(x_t) + e_t \approx h(x_t^{\text{nom}}) + H_t(x_t - x_t^{\text{nom}}) + e_t \\ &= H_t x_t + \underbrace{h(x_t^{\text{nom}}) - H_t x_t^{\text{nom}}}_{=: u_t^h} + e_t = H_t x_t + u_t^h + e_t, \end{aligned} \quad (5.15b)$$

where the gradient

$$H_t := \left( \nabla_x h(x) \Big|_{x=x_t^{\text{nom}}} \right)^T,$$

and the artificial input  $u_t^h$  are both known. It is now possible to use the linearized system (5.15) in the Kalman filter. Note that since the nominal trajectory is assumed known, it is possible to precompute  $P$  and  $K$ . However, if the nominal trajectory is not known, or not accurate enough to allow for the higher order terms in the linearization to be discarded, other methods are needed.

## 5.4.2 Extended Kalman Filter

In the *extended Kalman filter* (EKF) the problem with the lack of a nominal trajectory is solved using the information available from estimates by the filter itself. The best state estimate available, the latest estimate, can be used to construct a new linearization, *i.e.*, substitute  $\hat{x}_{t|t}$  for  $x_t^{\text{nom}}$  in (5.15a) and  $\hat{x}_{t|t-1}$  for  $x_t^{\text{nom}}$  in (5.15b). For this linearization,  $K$  and  $P$  cannot be computed at design time, because neither  $\hat{x}_{t|t}$  nor  $\hat{x}_{t|t-1}$  are available beforehand, both become available in the time step before they are needed in the recursion. Furthermore, when  $\hat{x}_{t|t}$  is used for  $x_t$  in the time update and  $\hat{x}_{t|t-1}$  in the measurement update, and  $E(w_t) = 0$  and  $E(e_t) = 0$

$$\begin{aligned} x_{t+1} &\approx f(\hat{x}_{t|t}, 0) \\ y_t &\approx h(\hat{x}_{t|t-1}), \end{aligned}$$

which can be used in the time update and measurement update phase, respectively.

The result, found in Algorithm 5.6, is the EKF. This filter is harder to analyze than the linearized Kalman filter, but in practice it has proved to work well in many applications. The filter can also be obtained by from the general Kalman filter, Algorithm 5.5, using a first order Gaussian approximation when transforming the stochastic variables.

Using a second order Gaussian approximation in Algorithm 5.5 yields another filter [9, 44], by some authors called EKF2. This EKF is given in Algorithm 5.7. Note that assuming additive process noise considerably simplifies the expressions, and that assumptions is made in most presentations. The presentation in Algorithm 5.7, however, does not make that assumption.

---

**Algorithm 5.6** Extended Kalman Filter
 

---

1. Initiate the filter with the initial information:

$$\hat{x}_{0|-1} = x_0 \quad \text{and} \quad P_{0|-1} = \Pi_0,$$

where  $x_0$  is the initial state estimate and  $\Pi_0 = \text{cov}(x_0)$ . Let  $t = 0$ .

2. Measurement update phase:

$$\begin{aligned} \hat{x}_{t|t} &= \hat{x}_{t|t-1} + K_t(y_t - h(\hat{x}_{t|t-1})) \\ P_{t|t} &= (I - K_t H_t) P_{t|t-1} \\ K_t &= P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1}, \end{aligned}$$

with

$$H_t := \left( \nabla_x h(x) \Big|_{x=\hat{x}_{t|t-1}} \right)^T.$$

3. Time update phase:

$$\begin{aligned} \hat{x}_{t+1|t} &= f(\hat{x}_{t|t}, 0) \\ P_{t+1|t} &= F_t P_{t|t} F_t^T + G_t Q_t G_t^T, \end{aligned}$$

where

$$F_t := \left( \nabla_x f(x, 0) \Big|_{x=\hat{x}_{t|t}} \right)^T \quad \text{and} \quad G_t := \left( \nabla_w f(\hat{x}_{t|t}, w) \Big|_{w=0} \right)^T.$$

4. Let  $t := t + 1$  and repeat from 2.
- 

### 5.4.3 Iterated Extended Kalman Filter

One problem that can sometimes occur with the EKF is that with a poor estimate the filter gain matrix  $K$  is affected due to linearizing around a point far from the true state. This is most serious when the functions involved are highly nonlinear. The *iterated extended Kalman filter* (IEKF) is a slight modification of the EKF intended to improve this behavior [64, 69].

The main idea in the IEKF is that the measurement update most likely improves the state estimate, and that if the new estimate is used for the linearization this should further improve performance. To utilize this idea, the measurement update of the EKF is performed repeatedly using what is likely to be better and better estimates of the state as they become available. This is obtained by replacing the measurement update in Algorithm 5.6 with

$$\hat{x}_{t|t} = \hat{x}_{t|t}^{(m)} \tag{5.18a}$$

$$P_{t|t} = (I - K_t^{(m)} H_t^{(m)}) P_{t|t-1}, \tag{5.18b}$$

---

**Algorithm 5.7** Second Order Extended Kalman Filter
 

---

1. Initiate the filter with the initial information:

$$\hat{x}_{0|-1} = x_0 \quad \text{and} \quad P_{0|-1} = \Pi_0,$$

where  $x_0$  is the initial state estimate and  $\Pi_0 = \text{cov}(x_0)$ . Let  $t := 0$ .

2. Measurement update phase:

$$\begin{aligned} \hat{x}_{t|t} &= \hat{x}_{t|t-1} + K_t (y_t - h(\hat{x}_{t|t-1}) - \frac{1}{2} [\text{tr} \Delta_x^x h_i(x) \Big|_{x=\hat{x}_{t|t-1}} P_{t|t-1}]_i) \\ P_{t|t} &= (I - K_t H_t) P_{t|t-1} \\ K_t &= P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T \\ &\quad + \frac{1}{2} [\text{tr} P_{t|t-1} \Delta_x^x h_i(x) \Big|_{x=\hat{x}_{t|t-1}} P_{t|t-1} \Delta_x^x h_j(x) \Big|_{x=\hat{x}_{t|t-1}}]_{ij} + R_t)^{-1} \end{aligned}$$

with

$$H_t := \left( \nabla_x h(x) \Big|_{x=\hat{x}_{t|t-1}} \right)^T.$$

3. Time update phase:

$$\begin{aligned} \hat{x}_{t+1|t} &= f(\hat{x}_{t|t}, 0) + \frac{1}{2} [\text{tr} \Delta_x^x f_i(x, 0) \Big|_{x=\hat{x}_{t|t}} P_{t|t}]_i \\ &\quad + \frac{1}{2} [\text{tr} \Delta_w^w f_i(\hat{x}_{t|t}, w) \Big|_{w=0} Q_t]_i \\ P_{t+1|t} &= F_t P_{t|t} F_t^T + G_t Q_t G_t^T \\ &\quad + \frac{1}{2} [\text{tr} P_{t|t} \Delta_x^x f_i(x, 0) \Big|_{x=\hat{x}_{t|t}} P_{t|t} \Delta_x^x f_j(x, 0) \Big|_{x=\hat{x}_{t|t}}]_{ij} \\ &\quad + \frac{1}{2} [\text{tr} Q_t \Delta_w^w f_i(\hat{x}_{t|t}, w) \Big|_{w=0} Q_t \Delta_w^w f_j(\hat{x}_{t|t}, w) \Big|_{w=0}]_{ij} \\ &\quad + \frac{1}{2} [\text{tr} P_{t|t} \Delta_x^x f_i(x, w) \Big|_{x=\hat{x}_{t|t}} Q_t \Delta_w^w f_j(x, w) \Big|_{w=0}]_{ij} \\ &\quad + \frac{1}{2} [\text{tr} Q_t \Delta_w^w f_i(x, w) \Big|_{w=0} P_{t|t} \Delta_x^x f_j(x, w) \Big|_{x=\hat{x}_{t|t}}]_{ij}, \end{aligned}$$

where

$$F_t := \left( \nabla_x f(x, 0) \Big|_{x=\hat{x}_{t|t}} \right)^T \quad \text{and} \quad G_t := \left( \nabla_w f(\hat{x}_{t|t}, w) \Big|_{w=0} \right)^T.$$

4. Let  $t := t + 1$  and repeat from 2.
-

where  $\hat{x}_{t|t}^{(m)}$ ,  $K_t^{(m)}$ , and  $H_t^{(m)}$  are given by the recursion

$$\begin{aligned}\hat{x}_{t|t}^{(0)} &= \hat{x}_{t|t-1} \\ \hat{x}_{t|t}^{(i)} &= \hat{x}_{t|t-1} + K_t^{(i)}(y_t - h(\hat{x}_{t|t-1})) \\ K_t^{(i)} &= P_{t|t-1} H_t^{(i)T} (H_t^{(i)} P_{t|t-1} H_t^{(i)T} + R_t)^{-1} \\ H_t^{(i)} &:= \left( \nabla_x h(x) \Big|_{x=\hat{x}_{t|t-1}^{(i)}} \right)^T.\end{aligned}$$

The number of iterative improvements performed is determined by the parameter  $m$ . Choosing  $m$  to be 3–5 is usually enough to obtain the desired effect. Note that this is an *ad hoc* method, and that in some situations it improves the result but also that sometimes the result is worse than the result of a standard EKF.

— **Example 5.2: Bearings-only tracking — applying the EKF** —

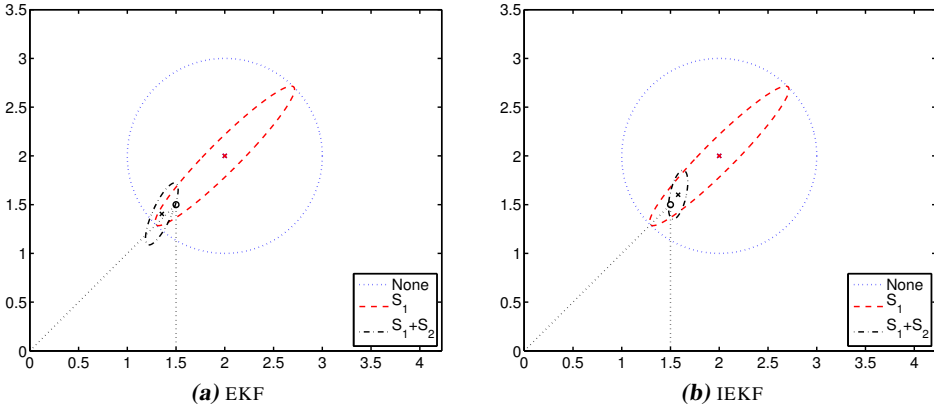
One common way to solve the bearings-only tracking problem in the introduction of this chapter is to apply an EKF or an IEKF. To apply the EKF, as described in Algorithm 5.6, the following gradients are needed:

$$\begin{aligned}F^T &= \nabla_x x = I \\ H^T &= \nabla_x \arctan\left(\frac{y - y^0}{x - x^0}\right) = \frac{1}{(x - x^0)^2 + (y - y^0)^2} \begin{pmatrix} -(y - y^0) \\ x - x^0 \end{pmatrix}.\end{aligned}$$

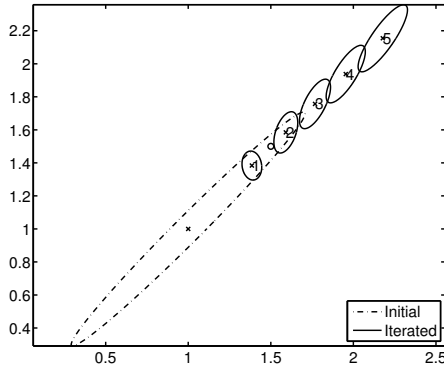
Note, the first order Taylor approximation of  $\arctan(x)$  around  $x$  is best for  $|x| \gg 1$  or  $|x| \ll 1$ , and that in practice moving between different quadrants is a problem that must be addressed.

Applying the EKF is now trivial and the resulting PDFs are shown in Figure 5.6(a), and the result if iterating five times in an IEKF is presented in Figure 5.6(b). For this specific instance the IEKF improves the situation.

However, the IEKF is not guaranteed to improve the situation, as shown in Figure 5.7. What happens here is similar to positive feedback. The first filter gain obtained moves the new linearization point further away from the value, and as the improved estimate is moved away the new linearization results in an even larger filter gain, that result in an even worse linearization point. Hence, if the IEKF is used it is important to monitor the progress so that this does not happen.



**Figure 5.6:** Estimate and covariance for the bearings-only example with zero, one, and two measurements, using an EKF and an IEKF. Compare this to the true distributions in Figure 5.1 and the particle estimate in Figure 5.5. (Estimates are denoted with  $\times$  in the center of each covariance ellipse, and the true target position is denoted with  $\circ$ .)



**Figure 5.7:** Illustration of the effect of the iteration in the IEKF when the second measurement arrives. (Estimates are denoted with  $\times$  in the center of each covariance ellipse, and the true target position is denoted with  $\circ$ .)

### 5.4.4 Unscented Kalman Filter

The EKF is sufficient for many applications. The popularity and wide spread usage of the EKF is a proof of this. However, there are situations when the EKF performs poorly. Furthermore, to use an EKF, gradients must be computed and computing the gradients can be computationally expensive and numerically ill conditioned. Julier [65] and Julier and Uhlmann [66, 67] therefore suggested an alternative approach that avoids numerical problems such as indefinite and unsymmetric covariance matrices and gradients by using the *unscented transform*, discussed in Section 3.4, in the generalized Kalman filter formulation, Algorithm 5.5. This way the Riccati recursion is performed only implicitly. This *unscented Kalman filter* (UKF) and aspects of it has also been discussed in for instance [142, 146, 147]. The UKF, sometimes also called the *sigma point filter*, and other similar approaches can be gathered in a common framework of *linear regression Kalman filters*, [87, 88].

The basic idea of the UKF, and other linear regression Kalman filters, is to use a set of carefully chosen points in the state space to capture the effect of model nonlinearities on means and covariances during filtration. The UKF uses the *unscented transform* [66] to select points from an augmented state space that includes the state,  $x_t$ , the process noise,  $w_t$ , and the measurement noise,  $e_t$ , in one augmented state vector,

$$\mathcal{X}_t = \begin{pmatrix} x_t \\ w_t \\ e_t \end{pmatrix},$$

with the dimension  $n_{\mathcal{X}}$ . Using the augmented state vector  $\mathcal{X}_t$  allows for a straightforward utilization of the unscented transform discussed in Section 3.4, especially when it comes to finding suitable sigma points. However, once the sigma points have been acquired the augmented state can be split up again to give a more familiar notation. By letting the sigma points pass through the model dynamics the following time update phase is obtained,

$$x_{t+1|t}^{(i)} = f(x_{t|t}^{(i)}, w_t^{(i)}),$$

and the result can then be combined to obtain the estimate

$$\hat{x}_{t+1|t} = \sum_{i=-n_{\mathcal{X}}}^{n_{\mathcal{X}}} \omega_t^{(i)} x_{t+1|t}^{(i)}$$

$$P_{t+1|t} = \sum_{i=-n_{\mathcal{X}}}^{n_{\mathcal{X}}} \omega_t^{(i)} (x_{t+1|t}^{(i)} - \hat{x}_{t+1|t})(x_{t+1|t}^{(i)} - \hat{x}_{t+1|t})^T.$$

The information in the measurements is introduced in a similar way by first obtaining the predicted measurements based on the sigma points

$$y_t^{(i)} = h(x_t^{(i)}, e_t^{(i)}),$$

yielding

$$\hat{y}_t = \sum_{i=-n_x}^{n_x} \omega_t^{(i)} y_t^{(i)}$$

$$P_{t|t-1}^{yy} = \sum_{i=-n_x}^{n_x} \omega_t^{(i)} (y_t^{(i)} - \hat{y}_t)(y_t^{(i)} - \hat{y}_t)^T.$$

The filter gain is then computed as the cross-covariance between state and measurement,

$$P_{t|t-1}^{xy} = \sum_{i=-n_x}^{n_x} \omega_t^{(i)} (x_{t|t-1}^{(i)} - \hat{x}_{t|t-1})(y_t^{(i)} - \hat{y}_t)^T,$$

divided by the covariance of the state, as this will project the state onto the new innovation, and the estimate follows as

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + P_{t|t-1}^{xy} P_{t|t-1}^{-yy} (y_t - \hat{y}_t)$$

$$P_{t|t} = P_{t|t-1} - P_{t|t-1}^{xy} P_{t|t-1}^{-yy} P_{t|t-1}^{xyT}.$$

The UKF is given in detail in Algorithm 5.8. It is of course possible to use the modified form of the unscented transform with different weights for mean and covariance computations. The author recommends, from personal experience, that this opportunity should be used as it often improves the result.

A question, not dealt with in detail here is how often new sigma points should be generated. In Algorithm 5.8 it is done once every time through the filtering loop, but it is also possible to do it before both measurement update and time update. However, if the transformation in the time update results in an asymmetric distribution, the sigma points will represent this fact, while the Gaussian approximation does not. Hence, generating new sigma points from the Gaussian approximation before the measurement update will lose information compared to keeping the ones from the time update. Note that the same is not true for the measurement update since new sigma points must be generated to represent the distributions after the filtering.

Furthermore, if the noise enters the system linearly it is possible to use this structure to improve performance since then the contributions from the noise can be handled analytically at low cost. More about this can be found in [147].

---

**Algorithm 5.8** Unscented Kalman Filter
 

---

1. Initiate the filter with the initial information:

$$\hat{x}_{0|-1} = x_0 \quad \text{and} \quad P_{0|-1} = \Pi_0,$$

where  $x_0$  is the initial estimate and  $\Pi_0 = \text{cov}(x_0)$ , and choose sigma points as described in step 3. Let  $t := 0$ .

2. Measurement update phase:

$$\begin{aligned} \hat{x}_{t|t} &= \hat{x}_{t|t-1} + P_{t|t-1}^{xy} P_{t|t-1}^{-yy} (y_t - \hat{y}_t) \\ P_{t|t} &= P_{t|t-1} - P_{t|t-1}^{xy} P_{t|t-1}^{-yy} P_{t|t-1}^{xyT}, \end{aligned}$$

where

$$\begin{aligned} y_t^{(i)} &= h(x_{t|t-1}^{(i)}, e_t^{(i)}) \\ \hat{y}_t &= \sum_{i=0}^N \omega_{m,t}^{(i)} y_t^{(i)} \\ P_{t|t-1}^{yy} &= \sum_{i=0}^N \omega_{c,t}^{(i)} (y_t^{(i)} - \hat{y}_t)(y_t^{(i)} - \hat{y}_t)^T \\ P_{t|t-1}^{xy} &= \sum_{i=0}^N \omega_{c,t}^{(i)} (x_{t|t-1}^{(i)} - \hat{x}_{t|t-1})(y_t^{(i)} - \hat{y}_t)^T. \end{aligned}$$

3. Choose  $N$  sigma points,  $\mathcal{X}_t^{(i)}$  partitioned as  $\mathcal{X}_t = \begin{pmatrix} x_t \\ w_t \\ e_t \end{pmatrix}$ , and weights,  $\omega_{m,t}^{(i)}$  and  $\omega_{c,t}^{(i)}$ , e.g., as suggested by (3.6).

4. Time update phase:

$$\begin{aligned} \hat{x}_{t+1|t} &= \sum_{i=0}^N \omega_{m,t}^{(i)} x_{t+1|t}^{(i)} \\ P_{t+1|t} &= \sum_{i=0}^N \omega_{c,t}^{(i)} (x_{t+1|t}^{(i)} - \hat{x}_{t+1|t})(x_{t+1|t}^{(i)} - \hat{x}_{t+1|t})^T, \end{aligned}$$

where

$$x_{t+1|t}^{(i)} = f(x_{t|t-1}^{(i)}, w_t^{(i)}).$$

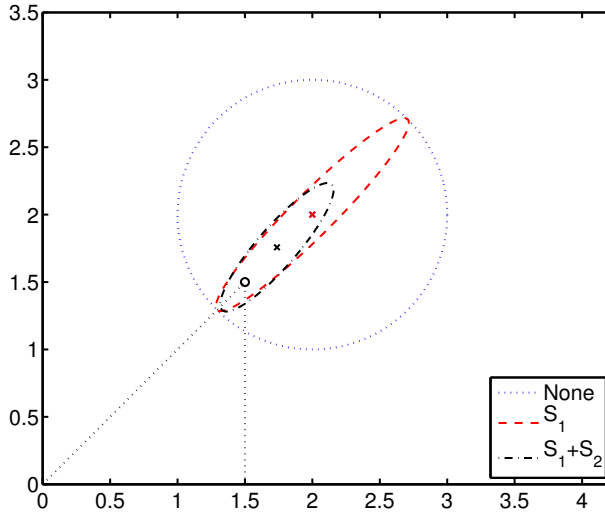
5. Let  $t := t + 1$  and repeat from 2.
-

---

**Example 5.3: Bearings-only tracking — applying the UKF**


---

The UKF is well suited for application to the tracking problem described in the introduction to this chapter. Using the modified form, and the recommended standard values for the parameters, the result in Figure 5.8



**Figure 5.8:** Estimate and covariance for the bearings-only example with no, one, and two measurements, using a UKF. Compare with Figures 5.1, 5.5, and 5.6. (Estimates are denoted with  $\times$  in the center of each covariance ellipse, and the true target position is denoted with  $\circ$ .)

---

## 5.5 Filter Banks

The structure of a switched model, as described in Section 4.2.2,

$$x_{t+1} = f(x_t, w_t, \delta_t) \quad (5.20a)$$

$$y_t = h(x_t, e_t, \delta_t), \quad (5.20b)$$

can be used when constructing a filter. The idea is to treat each mode of the model independently, design filters as if the mode was known, and combine the independent results based on the likelihood of the obtained measurements. The number of parallel filters must in practice be kept from increasing exponentially and different approaches have been developed to achieve this. Two major and closely related methods are the *generalized pseudo-Bayesian* (GPB) filter [20] and the *interacting multiple models* (IMM) filter [20]. Especially the latter is popular in the tracking community. The *adaptive forgetting through multiple models* (AFMM) by Andersson [5] is another solution to the problem.

The three different filters assume that the different modes represent linear models and differ mainly in how and when the exponential complexity is avoided, and in how the estimate is computed. A similar idea is used in the *Gaussian sum filter*, [3, 132], however it approximates the posterior distribution with a Gaussian sum without necessarily having an underlying switched model. This section covers how to maintain a complete multiple model filter bank, and then a *pruning* algorithm and the GPB algorithm are presented as examples of ways to handle the growing filter complexity.

### 5.5.1 Complete Filter Bank

To get the inferred PDF for (4.9) one filter for each possible mode is maintained and associated with a probability based on the model and the available measurements. Given that there are filters to correctly handle the different modes of the model, the result is optimal in the sense that it produces the correct *posterior* distribution for  $x_t$ .

To do this a *filter bank*,  $\mathcal{F}$ , is maintained to hold the filters and the matching probabilities. If Kalman filters, or EKFs, are used, the filter bank  $\mathcal{F}_{t|t}$  reduces to a set of quadruples  $(\delta_t, \hat{x}_{t|t}^{(\delta_t)}, P_{t|t}^{(\delta_t)}, \omega_{t|t}^{(\delta_t)})$  representing mode, estimate, covariance matrix, and probability of the filter operating in that mode. The mode history up to time  $t$  is denoted  $\delta_t = \{\delta_i\}_{i=1}^t$ .

Now, assume that a filter bank,  $\mathcal{F}_{t|t}$ , is available that incorporates all information available from the model and measurements at time  $t$ . That is, there is one entry in it for every node at time  $t$  in the tree in Figure 4.2.

In order for the filter bank to evolve in time and correctly represent the *posterior* state distribution it must *branch*, i.e., for each filter in  $\mathcal{F}_{t|t}$  in total  $n_\delta$  new filters should be created. One filter for each possible mode at the next time step. These new filters obtain their initial state from the filter they are derived from and are then time updated. The probability of these new filters are

$$\omega_{t+1|t}^{(\delta_{t+1})} = p(\delta_{t+1}|\mathbb{Y}_t) = p(\delta_{t+1}|\delta_t)p(\delta_t|\mathbb{Y}_t) = p_{\delta_{t+1}|\delta_t}\omega_{t|t}^{(\delta_t)}.$$

The new filters together with the associated probabilities make up the filter bank  $\mathcal{F}_{t+1|t}$ .

The next step is to update the filter bank when a new measurement arrives. This is done in two steps. First, each individual filter in  $\mathcal{F}_{t|t-1}$  is updated using standard measurement update methods, e.g., a Kalman filter, and then the probability is updated according to how probable that mode is given the measurement,

$$\omega_{t|t}^{(\delta_t)} = p(\delta_t|\mathbb{Y}_t) = \frac{p(y_t|\delta_t, \mathbb{Y}_{t-1})p(\delta_t|\mathbb{Y}_{t|t-1})}{p(y_t|\mathbb{Y}_t)} = \frac{p(y_t|\delta_t)\omega_{t|t-1}^{(\delta_t)}}{\sum_{\delta_t} p(y_t|\delta_t)\omega_{t|t-1}^{(\delta_t)}},$$

where the last step requires that the measurements are mutually independent and that only the present state affects the measurement. The filters and the associated weights can now be gathered in the filter bank  $\mathcal{F}_{t|t}$ .

An algorithm to create a complete filter bank is given in Algorithm 5.9.

**Algorithm 5.9** Filter Bank Algorithm

1. Let  $\mathcal{F}_{0|-1}$  representing the initial state knowledge, e.g., with one single entry  $(\delta_0, x_0, \Pi_0, 1)$ . Let  $t := 0$ .
2. Measurement update all filters in  $\mathcal{F}_{t|t-1}$  and put the result in  $\mathcal{F}_{t|t}$ . Update the mode probabilities in  $\mathcal{F}_{t|t}$  using

$$\omega_{t|t}^{(\delta_t)} = \frac{p(y_t|\delta_t)\omega_{t|t-1}^{(\delta_t)}}{\sum_{\delta_t} p(y_t|\delta_t)\omega_{t|t-1}^{(\delta_t)}}.$$

The minimum variance estimate based on this is:

$$\hat{x}_{t|t} = \sum_{\delta_t} \omega_{t|t}^{(\delta_t)} \hat{x}_{t|t}^{(\delta_t)}$$

$$P_{t|t} = \sum_{\delta_t} \omega_{t|t}^{(\delta_t)} \left( P_{t|t}^{(\delta_t)} + (\hat{x}_{t|t}^{(\delta_t)} - \hat{x}_{t|t}) (\hat{x}_{t|t}^{(\delta_t)} - \hat{x}_{t|t})^T \right).$$

3. Optional filter bank reduction. (Ignore to get a complete filter bank.)
4. Branch the filter bank, i.e., from  $\mathcal{F}_{t|t}$  construct  $\mathcal{F}_{t+1|t}$  by copying each entry in  $\mathcal{F}_{t|t}$   $n_\delta$  times to the new filter bank only updating the current mode, and the probability of that mode. The probabilities of the modes are updated using

$$\omega_{t+1|t}^{(\delta_{t+1})} = p_{\delta_{t+1}|\delta_t} \omega_{t|t}^{(\delta_t)}.$$

The minimum variance estimate based on this is:

$$\hat{x}_{t+1|t} = \sum_{\delta_{t+1}} \omega_{t+1|t}^{(\delta_{t+1})} \hat{x}_{t+1|t}^{(\delta_{t+1})}$$

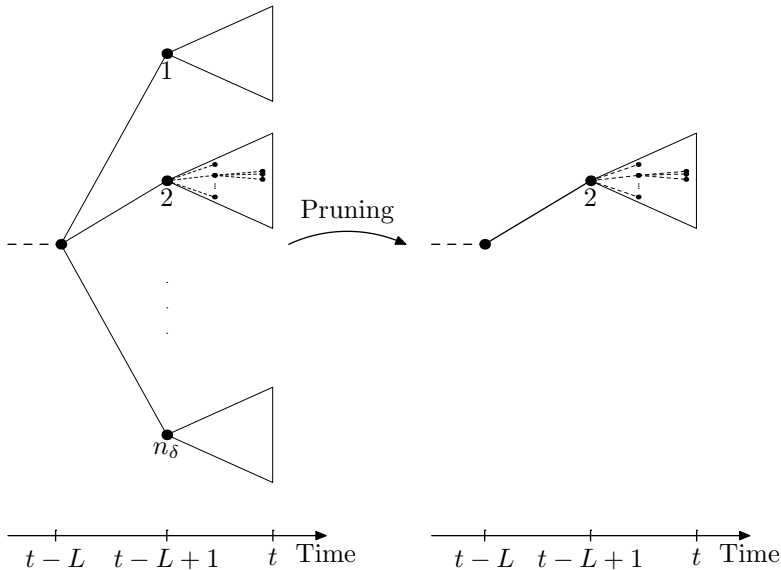
$$P_{t+1|t} = \sum_{\delta_{t+1}} \omega_{t+1|t}^{(\delta_{t+1})} \left( P_{t+1|t}^{(\delta_{t+1})} + (\hat{x}_{t+1|t}^{(\delta_{t+1})} - \hat{x}_{t+1|t}) (\hat{x}_{t+1|t}^{(\delta_{t+1})} - \hat{x}_{t+1|t})^T \right).$$

5. Time update all filters in  $\mathcal{F}_{t+1|t}$ .
6. Let  $t := t + 1$  and repeat from 2.

**5.5.2 Filter Bank with Pruning**

The method used in this section demonstrates how to reduce the size of a filter bank by simply throwing away, *pruning*, unlikely mode combinations to allow for more focus on the more probable ones. To achieve this only minor modifications to Algorithm 5.9 are needed.

To keep a complete filter bank over a window of  $L$  samples the filter bank must hold at least  $(n_\delta)^L$  filters with associated probabilities. This way the window length  $L$  becomes



**Figure 5.9:** Illustration of pruning of a filter bank represented as a tree. In the left tree, before pruning, the subtree (represented by a triangle) originating at  $\delta_{t-L+1} = 2$  is the most probable and the other modes are removed. The result is displayed to the right.

a parameter for tuning; if  $L$  is large the inferred PDF is accurate at the cost of a large filter bank, if  $L$  is small the filter bank is small but the approximation less accurate. A rule of thumb is to choose  $L \geq n_x + 1$  so that the modes get a chance to manifest themselves before being removed [44].

The reduction of the filter bank, the pruning, is performed in the following way. Since a complete mode history over a window of  $L$  samples is asked for, the different modes at time  $T - L + 1$  are examined. This should be the first time instance where multiple modes are present. Determine which mode at this level that is most probable and remove all filters in the filter bank not stemming from this mode. This reduces the filter bank to one  $n_\delta$ th of the size. Hopefully the removed filters carry little weight, hence the effect on the estimate will be small. As a last step, renormalize the weights. The pruning process is illustrated in Figure 5.9.

The pruning can be performed in connection with any of the steps in Algorithm 5.9. However, the best place to do it is between steps 2 and 4. At this point new information from measurements has just been incorporated into the filter, and the model is just about to branch and make a deterministic time update. Hence, branching at this time is the only sensible alternative. The filter bank algorithm with pruning is given in Algorithm 5.10.

**Algorithm 5.10** Filter Bank Reduction using Pruning

This filter bank reduction step is to be used as step 3 in Algorithm 5.9.

3. Prune the filter bank. Compute which mode,  $\delta$ , that at time  $t - L + 1$  is the most probable, *i.e.*,

$$\delta = \arg \max_i \sum_{\{\delta_t: \delta_{t-L+1}=i\}} \omega_{t|t}^{\delta_t}.$$

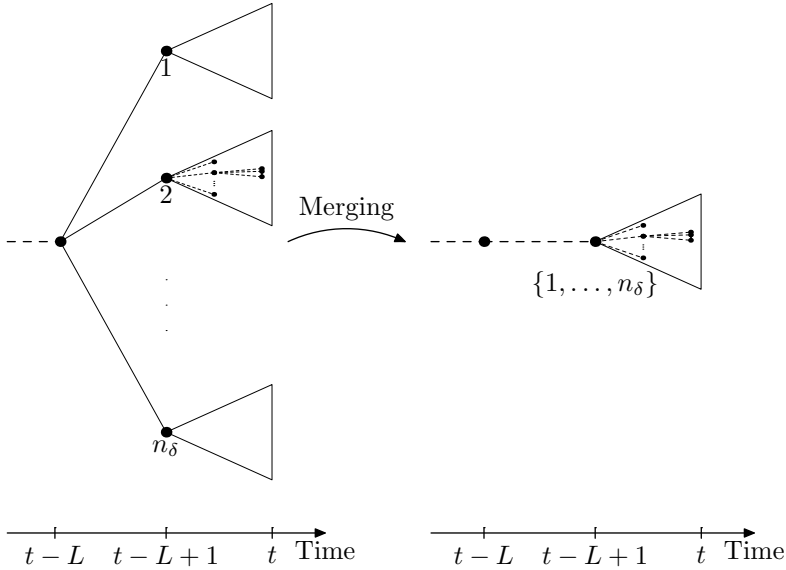
Remove all filters from  $\mathcal{F}_{t|t}$  not having  $\delta_{t-L+1} = \delta$ , and renormalize the remaining weights,

$$\omega_{t|t}^{(i)} := \omega_{t|t}^{(i)} / \sum_j \omega_{t|t}^{(j)}.$$

**5.5.3 Filter Bank with Merging**

An alternative approach to throwing away unlikely branches, is to combine or merge branches that share the same history in some window of length  $L$ . This way all branches contribute to the final result. The idea is that the effect of the exact mode  $L$  time steps ago should have died out, and it therefore makes sense to combine branches that only differs in that mode. This approach is used in filtering methods such as GPB and IMM [18, 20]. This section will describe the GPB approach, which fits better into the framework set by Algorithm 5.9 than the IMM. The IMM is then a clever implementation of the GPB with window size  $L = 2$  samples, where the number of modes actually propagated is reduced. For details on the IMM the reader is referred to [18, 20].

The GPB is usually presented with a window of size one or two, but the generalization to any window length is straightforward. Instead of pruning some branches, branches with the same  $\delta_{t-L+1}$  are combined to a new branch. The merging assumes all modes to be associated with a weight and a Gaussian distribution, resulting in a Gaussian sum. This sum is then reduced to one mode, represented by a Gauss with the same mean and covariance as the Gaussian sum it approximates and the combined weight. This is illustrated in Figure 5.10 and the details are given in Algorithm 5.11.



**Figure 5.10:** Illustration of merging of a filter bank represented as a tree. The resulting tree on the right hand side is the combination of all the subtrees on the left hand side.

---

**Algorithm 5.11** Filter Bank Reduction using Merging

---

This filter bank reduction step is to be used as step 3 in Algorithm 5.9.

3. Merge the filter bank. The result is a set of filters with common history  $L - 1$  steps back in history.

$$\omega_{t|t}^{(\delta_t^{L-1})} = \sum_{\delta_{t-L+1}} \omega_{t|t}^{(\delta_t^L)}$$

$$\hat{x}_{t|t}^{(\delta_t^{L-1})} = \sum_{\delta_{t-L+1}} \frac{\omega_{t|t}^{(\delta_t^L)}}{\omega_{t|t}^{(\delta_t^{L-1})}} \hat{x}_{t|t}^{(\delta_t^L)}$$

$$P_{t|t}^{(\delta_t^{L-1})} = \sum_{\delta_{t-L+1}} \frac{\omega_{t|t}^{(\delta_t^L)}}{\omega_{t|t}^{(\delta_t^{L-1})}} \left( P_{t|t}^{(\delta_t^L)} + (\hat{x}_{t|t}^{(\delta_t^L)} - \hat{x}_{t|t}^{(\delta_t^{L-1})}) (\hat{x}_{t|t}^{(\delta_t^L)} - \hat{x}_{t|t}^{(\delta_t^{L-1})})^T \right),$$

where  $\delta_t^L = \{\delta_i\}_{i=t-L+1}^t$ .

---

## 5.6 Rao-Blackwellized Particle Filter

Back in the 1940's Rao, [112], and Blackwell, [19], showed that an estimator can be improved by using information about conditional probabilities. Furthermore, they showed how the estimator based on this knowledge should be constructed as a conditioned expected value of an estimator not taking the extra information into consideration. The technique is now called *Rao-Blackwellization*.

The *Rao-Blackwell theorem* is presented in this section and the potential gain for the special case of the covariance of the estimate is given. Knowing how much can be gained, the expressions needed to use Rao-Blackwellization in recursive Bayesian filtering are derived. Finally, for the important special case of a nonlinear state-space model with a linear Gaussian substructure, a mixture of a Kalman filter bank and a particle filter is given. The filter is often referred to as the *Rao-Blackwellized particle filter* (RBPF) [6, 26, 33] or by some authors the *marginalized particle filter* (MPF) [72, 125, 127].

### 5.6.1 Performance Gain

The Rao-Blackwell theorem can be used to show how the knowledge of a conditional probability can improve estimation performance, but this is only a special case. The theorem specifies that any convex loss function improves if a conditional probability is utilized. The Rao-Blackwell theorem is given in Theorem 5.1, [89, Theorem 6.4].

#### Theorem 5.1 (The Rao-Blackwell theorem [89])

Let  $x$  be a random observable with distribution  $P_\theta \in \mathcal{P} = \{P_{\theta'} \in \Omega\}$ , and let  $t$  be sufficient statistics for  $\mathcal{P}$ . Let  $\hat{\theta}$  be an estimator of an estimand  $g(\theta)$ , and let the loss function  $L(\theta, d)$  be a strictly convex function of  $d$ . Then if  $\hat{\theta}$  has finite expectation and risk,

$$R(\theta, \hat{\theta}) = \mathbb{E} L(\theta, \hat{\theta}(x))$$

and if

$$\hat{\theta}^{\text{RB}}(t) = \mathbb{E}_{X|t} \hat{\theta}(x)$$

the risk of the estimator  $\hat{\theta}^{\text{RB}}(t)$  satisfies

$$R(\theta, \hat{\theta}^{\text{RB}}) \prec R(\theta, \hat{\theta})$$

unless  $\hat{\theta}(x) = \hat{\theta}^{\text{RB}}(t)$  with probability 1.

**Proof:** See [89]. □

An important special case of Theorem 5.1 describes variance properties of a Rao-Blackwell estimator.

#### Corollary 5.1

Let  $\hat{\theta}$  be an unbiased estimator that fulfills the requirements in Theorem 5.1 and  $\hat{\theta}^{\text{RB}}$  the Rao-Blackwell estimator, then

$$\text{cov}(\hat{\theta}) \succeq \text{cov}(\hat{\theta}^{\text{RB}}).$$

**Proof:** Follows directly by choosing  $L(\theta, \hat{\theta}) = (\theta - \hat{\theta})(\theta - \hat{\theta})^T$ , and applying Theorem 5.1.  $\square$

How large the improvement is in terms of covariance can be derived from the separation of covariance, [49], which states

$$\text{cov}(x) = \text{cov } E_{x|y} x + E \text{cov}_{x|y} x.$$

Assume that  $x$  is the estimate, then  $E_{x|y} x$  is the improved estimate from the Rao-Blackwell theorem. Hence, compared to the Rao-Blackwellized estimate, the regular estimate has an extra covariance term,  $E \text{cov}_{x|y} x$ , that by definition is positive definite. The covariance gain obtained with Rao-Blackwell estimator comes from eliminating this term in the covariance expression.

The next section will show how to utilize structure in a filtering problem to be able to get the gain promised by the Rao-Blackwell theorem. Based on this, a state-space model with linear Gaussian substructure will be presented where the Rao-Blackwell theorem applies and an algorithm will be presented that combines a Kalman filter bank with a particle filter into a Rao-Blackwellized particle filter.

## 5.6.2 Rao-Blackwellization for Filtering

It is possible to utilize the Rao-Blackwell theorem in recursive filtering given some properties of the involved distributions. Let the state vector be split into  $x = \begin{pmatrix} x^p \\ x^k \end{pmatrix}$  then it is possible to write the posterior distribution as

$$p(x_t^k, \mathbb{X}_t^p | \mathbb{Y}_t) = p(x_t^k | \mathbb{X}_t^p, \mathbb{Y}_t) p(\mathbb{X}_t^p | \mathbb{Y}_t), \quad (5.22)$$

where  $x^p$  should take the place of the sufficient statistics  $t$  in Theorem 5.1. Preferably,  $p(x_t^k | \mathbb{X}_t^p, \mathbb{Y}_t)$  should be available in closed form, and allow for efficient estimation of  $x^k$ . Furthermore, assumptions are made on the underlying model to simplify things:

$$p(x_{t+1} | \mathbb{X}_t^p, x_t^k, \mathbb{Y}_t) = p(x_{t+1} | x_t) \quad (5.23a)$$

$$p(y_{t+1} | \mathbb{X}_t^p, x_t^k, \mathbb{Y}_t) = p(y_{t+1} | x_t). \quad (5.23b)$$

This implies a hidden Markov process. It will later be shown how an efficient filter-bank-based particle filter can be derived with the additional assumption about a linear Gaussian substructure.

In this section recursive filtering equations will be derived that utilize Rao-Blackwellization. The time and measurement update will be treated separately, starting with the time update. This will also show how the conditional separation is maintained. The presentation is slightly different from the presentation usually given in literature, but this formulation provides a more efficient presentation for the linear Gaussian substructure case. The result is an algorithm and a discussion about how this algorithm fits very well into the filter bank framework.

### Initialization

To initialize the filtering recursion, the following distribution is assumed known:

$$p(x_t^k, \mathbb{X}_t^p | \mathbb{Y}_t) = p(x_t^k | \mathbb{X}_t^p, \mathbb{Y}_t) p(\mathbb{X}_t^p | \mathbb{Y}_t), \quad (5.24)$$

where  $p(x_t^k | \mathbb{X}_t^p, \mathbb{Y}_t)$  should be analytically tractable for best result.

### Time Update

The time update can be performed in different ways, depending on what is most suitable in the application at hand. The difference lies in how the first factor of the right hand side of (5.22) is handled.

One way to perform the time update is to first add  $x_{t+1}^p$  as another condition, and then proceed to increase the time index of  $x_t^k$ . This is the approach chosen in e.g., [126]. Doing the time update this way gives the following two steps:

$$p(x_t^k | \mathbb{X}_{t+1}^p, \mathbb{Y}_t) = \frac{p(x_{t+1}^p | \mathbb{X}_t^p, x_t^k, \mathbb{Y}_t) p(x_t^k | \mathbb{X}_t^p, \mathbb{Y}_t)}{p(x_{t+1}^p | \mathbb{X}_t^p, \mathbb{Y}_t)} \quad (5.25a)$$

$$p(x_{t+1}^k | \mathbb{X}_{t+1}^p, \mathbb{Y}_t) = \int p(x_{t+1}^k | \mathbb{X}_{t+1}^p, x_t^k, \mathbb{Y}_t) p(x_t^k | \mathbb{X}_{t+1}^p, \mathbb{Y}_t) dx_t^k. \quad (5.25b)$$

The expressions can be interpreted as a virtual measurement update, and a regular time update, cf. (5.8b) and (5.8a), respectively. This is the way the time update is often presented, see e.g., [125].

An alternative way to reach the same final result, more suitable for the filter bank interpretation of the RBPF, is to first derive  $p(x_{t+1}^p, x_{t+1}^k | \mathbb{X}_t^p, \mathbb{Y}_t)$  and then introduce  $x_{t+1}^p$  as a condition. This turns (5.8a) into the following two steps:

$$\begin{aligned} p(x_{t+1}^p, x_{t+1}^k | \mathbb{X}_t^p, \mathbb{Y}_t) &= \int p(x_{t+1}^p, x_{t+1}^k | \mathbb{X}_t^p, x_t^k, \mathbb{Y}_t) p(x_t^k | \mathbb{X}_t^p, \mathbb{Y}_t) dx_t^k \\ &= \int p(x_{t+1}^p, x_{t+1}^k | x_t^p, x_t^k) p(x_t^k | \mathbb{X}_t^p, \mathbb{Y}_t) dx_t^k, \end{aligned} \quad (5.26a)$$

where (5.23a) has been used in the last step. Introducing  $x_{t+1}^p$  as a condition follows immediately as

$$p(x_{t+1}^k | \mathbb{X}_{t+1}^p, \mathbb{Y}_t) = \frac{p(x_{t+1}^p, x_{t+1}^k | \mathbb{X}_t^p, \mathbb{Y}_t)}{p(x_{t+1}^p | \mathbb{X}_t^p, \mathbb{Y}_t)}. \quad (5.26b)$$

At a first look, (5.26a) may look a bit more involved than (5.25), due to the need for  $p(x_{t+1}^p, x_{t+1}^k | \mathbb{X}_t^p, x_t^k, \mathbb{Y}_t)$ . However, in practice this is rarely a problem, and the latter formulation has other preferable properties when it comes to designing a filter.

For both approaches, the second factor of (5.22) can then be handled using marginalization,

$$\begin{aligned} p(\mathbb{X}_{t+1}^p | \mathbb{Y}_t) &= p(x_{t+1}^p | \mathbb{Y}_t, \mathbb{X}_t^p) p(\mathbb{X}_t^p | \mathbb{Y}_t) \\ &= \int p(x_{t+1}^p | x_t^k, \mathbb{X}_t^p, \mathbb{Y}_t) p(x_t^k | \mathbb{Y}_t, \mathbb{X}_t^p) dx_t^k p(\mathbb{X}_t^p | \mathbb{Y}_t) \\ &= \int p(x_{t+1}^p | x_t^k, x_t^p) p(x_t^k | \mathbb{Y}_t, \mathbb{X}_t^p) dx_t^k p(\mathbb{X}_t^p | \mathbb{Y}_t), \end{aligned} \quad (5.26c)$$

where the introduction of the marginalization in the middle step is the trick needed for this to be useful. The last step uses (5.23a).

Note that the conditional separation still holds so that

$$p(x_{t+1}^k, \mathbb{X}_{t+1}^p | \mathbb{Y}_t) = p(x_{t+1}^k | \mathbb{X}_{t+1}^p, \mathbb{Y}_t) p(\mathbb{X}_{t+1}^p | \mathbb{Y}_t), \quad (5.27)$$

which is still suitable for a Rao-Blackwellized measurement update.

### Measurement Update

The next step is to introduce information from the measurement  $y_{t+1}$  into the posterior distributions in (5.27). First for the left factor in (5.27),

$$\begin{aligned} p(x_{t+1}^k | \mathbb{Y}_{t+1}, \mathbb{X}_{t+1}^p) &= \frac{p(y_{t+1} | \mathbb{Y}_t, \mathbb{X}_{t+1}^p, x_{t+1}^k) p(x_{t+1}^k | \mathbb{Y}_t, \mathbb{X}_{t+1}^p)}{p(y_{t+1} | \mathbb{Y}_t, \mathbb{X}_{t+1}^p)} \\ &= \frac{p(y_{t+1} | x_{t+1}^k, \mathbb{X}_{t+1}^p) p(x_{t+1}^k | \mathbb{Y}_t, \mathbb{X}_{t+1}^p)}{p(y_{t+1} | \mathbb{Y}_t, \mathbb{X}_{t+1}^p)}. \end{aligned} \quad (5.28a)$$

The last equality follows from (5.23b).

The second factor of (5.27) follows in a similar way,

$$\begin{aligned} p(\mathbb{X}_{t+1}^p | \mathbb{Y}_{t+1}) &= \frac{p(y_{t+1} | \mathbb{Y}_t, \mathbb{X}_{t+1}^p) p(\mathbb{X}_{t+1}^p | \mathbb{Y}_t)}{p(y_{t+1} | \mathbb{Y}_t)} \\ &= \frac{\int p(y_{t+1} | \mathbb{Y}_t, \mathbb{X}_{t+1}^p, x_{t+1}^k) p(x_{t+1}^k | \mathbb{Y}_t, \mathbb{X}_{t+1}^p) dx_{t+1}^k p(\mathbb{X}_{t+1}^p | \mathbb{Y}_t)}{p(y_{t+1} | \mathbb{Y}_t)} \\ &= \frac{\int p(y_{t+1} | x_{t+1}^k, \mathbb{X}_{t+1}^p) p(x_{t+1}^k | \mathbb{Y}_t, \mathbb{X}_{t+1}^p) dx_{t+1}^k p(\mathbb{X}_{t+1}^p | \mathbb{Y}_t)}{p(y_{t+1} | \mathbb{Y}_t)}, \end{aligned} \quad (5.28b)$$

where once again the marginalization in the middle step is a trick and the last equality uses (5.23b).

This completes the recursion since this gives

$$p(x_{t+1}^k, \mathbb{X}_{t+1}^p | \mathbb{Y}_{t+1}) = p(x_{t+1}^k | \mathbb{X}_{t+1}^p, \mathbb{Y}_{t+1}) p(\mathbb{X}_{t+1}^p | \mathbb{Y}_{t+1}). \quad (5.29)$$

### 5.6.3 Rao-Blackwellized Filtering with Linear Gaussian Structure

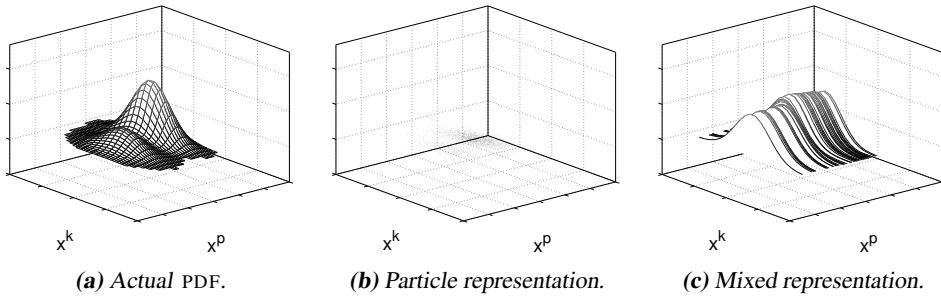
The filter that will be derived borrows ideas both from Kalman filter banks and from the particle filter. A problem with the particle filter is that it needs many particles to work well in high-dimensional state spaces. With many particles the particle filter is quite computationally demanding, sometimes making it infeasible to apply in practise. On the other hand, the Kalman filter, and variations thereof, are computationally cheap but does not handle nonlinear systems and systems with non-Gaussian noise as well as particle filters.

The idea is therefore to, for the system in (4.11),

$$x_{t+1}^p = f^p(x_t^p) + F^p(x_t^p)x_t^k + G^p(x_t^p)w_t^p \quad (5.30a)$$

$$x_{t+1}^k = f^k(x_t^p) + F^k(x_t^p)x_t^k + G^k(x_t^p)w_t^k \quad (5.30b)$$

$$y_t = h(x_t^p) + H^y(x_t^p)x_t^k + e_t, \quad (5.30c)$$



**Figure 5.11:** Illustration of the state representation in the RBPF (c), to be compared with the particle representation in (b), and the actual distribution depict in (a). Note that 5 000 particles are used for the particle representation, whereas only 50 was needed for an acceptable RBPF representation.

which is simplified to

$$x_{t+1} = F_t(x_t^p)x_t + f(x_t^p) + G_t(x_t^p)w_t \quad (5.31a)$$

$$y_t = H_t(x_t^p)x_t + h(x_t^p) + e_t, \quad (5.31b)$$

try to combine the two to obtain a filter that handles the nonlinearities, at a low computational complexity, at the same time as the Rao-Blackwell gain is obtained. (Refer to Section 4.2.3 for more details about the model and the notation used.) This is achieved by using Kalman filters to handle the linear Gaussian states, and particle filtering techniques to handle the remaining states.

To do this, the state space of (5.30) is divided into two parts,  $x^p$  that is nonlinear and non-Gaussian and must be handled with particle filtering techniques, and  $x^k$  that conditioned on  $x^p$  is linear and Gaussian, and hence suited for the Kalman filter. Doing this separation the dimension of the state space that must be covered with particles is reduced, reducing the need for many particles. This improves performance.

The state is represented by a set of particles with matching covariance matrices and weights,

$$x^{(i)} = \begin{pmatrix} x^{p(i)} \\ x^{k(i)} \end{pmatrix} \quad P^{(i)} = \begin{pmatrix} 0 & 0 \\ 0 & P^{k(i)} \end{pmatrix} \quad \omega^{(i)},$$

where the particles are chosen to follow the distribution for  $x^p$ , and  $\omega^{(i)}$  represents the particle weight. Here,  $x^p$  is point distributed, hence the singular covariance matrix. Furthermore, the value of  $x^k$  depends on the specific  $x^p$ . Figure 5.11 illustrates this representation, and compares it to a pure particle representation.

Based on this, the filtering problem can be solved with the time update and the measurement update steps outlined below.

### Initialize

To initialize the filter, an initial distribution for the state is needed. Usually, it is chosen so that the nonlinear part and the linear Gaussian part of the state space is independent. If

this is the case, draw  $N$  IID  $x_{0|-1}^{p(i)} \sim p(x_0^p)$ , set  $\omega_{0|-1} = \frac{1}{N}$ , and compose the combined state vectors as

$$x_{0|-1}^{(i)} = \begin{pmatrix} x_{0|-1}^{p(i)} \\ x_{0|-1}^k \end{pmatrix} \quad P^{(i)} = \begin{pmatrix} 0 & 0 \\ 0 & \Pi_{0|-1}^k \end{pmatrix}.$$

This now gives an initial state estimate with a representation similar to Figure 5.11(c).

### Time Update

The time update step in the filter will be derived analogous to the derivation of the filtering distributions in Section 5.6.2. The first task is to introduce the new information from the model (4.11) into (5.26b). The result is

$$\begin{aligned} p(x_{t+1}|\mathbb{X}_t^p, \mathbb{Y}_t) &= \int p(x_{t+1}|x_t)p(x_t^k|\mathbb{X}_t^p, \mathbb{Y}_t) dx_t^k \\ &= \int \mathcal{N}(x_{t+1}; F_t x_{t|t} + f(x_t^p), F_t P_{t|t} F_t^T + G_t Q_t G_t^T) \cdot \mathcal{N}(x_t^k; x_{t|t}^k, P_{t|t}^k) dx_t^k \\ &= \mathcal{N}(x_{t+1}; F_t x_{t|t} + f(x_t^p), F_t P_{t|t} F_t^T + G_t Q_t G_t^T), \end{aligned} \quad (5.32)$$

where the notation in Section 4.2.3 has been used. The result uses both the initial Gaussian assumption, as well as the Markov property in (5.23a). The last step follows immediately with only Gaussian distributions involved. It can either be recognized as a Kalman filter time update step, or be derived through straightforward but lengthy calculations. In terms of the representation in the filter, this can be expressed as

$$x'_{t+1|t} = F_{t|t}^{(i)} x_{t|t}^{(i)} + f(x_t^{p(i)}) \quad (5.33a)$$

$$P'_{t+1|t} = F_{t|t}^{(i)} P_{t|t}^{(i)} F_{t|t}^{(i)T} + G_{t|t}^{(i)} Q_t G_{t|t}^{(i)T}. \quad (5.33b)$$

Note that this actually time updates the  $x^p$  part of the state vector as if it was a regular part of the state. As a result,  $x$  does no longer have a point distribution in the  $x^p$  dimension, instead the distribution is now a Gaussian mixture. To continue, it is suitable to update the particle part of the state representation using (5.26c). Conveniently enough, the distribution  $p(x_{t+1}^p|\mathbb{X}_t^p, \mathbb{Y}_t)$  is available as a marginalization, yielding

$$p(x_{t+1}^k|\mathbb{X}_{t+1}^p, \mathbb{Y}_t) = \frac{\mathcal{N}(x_{t+1}^p, x_{t+1}^k; x'_{t+1|t}, P'_{t+1|t})}{p(x_{t+1}^p|\mathbb{X}_{t+1}^p, \mathbb{Y}_t)}.$$

This can be identified as a measurement update in a Kalman filter, where the newly selected particles become virtual measurements without measurement noise. Once again, this can be verified with straightforward, but quite lengthy, calculations.

The combination of the particle filter step and the conditioning on  $x_{t+1}^p$  can now be combined into the following virtual measurement update

$$x_{t+1|t}^{(i)} = x'_{t+1|t} + P'_{t+1|t} H'^T (H' P'_{t+1|t} H'^T)^{-1} (\xi_{t+1}^{(i)} - H' x'_{t+1|t}) \quad (5.34a)$$

$$P_{t+1|t}^{(i)} = P'_{t+1|t} - P'_{t+1|t} H'^T (H' P'_{t+1|t} H'^T)^{-1} H' P'_{t+1|t}, \quad (5.34b)$$

where  $H' = (I \ 0)$ . The virtual measurements are chosen from the Gaussian distribution given by (5.33),

$$\xi_{t+1}^{(i)} \sim \mathcal{N}(H' x_{t+1|t}^{(i)}, H' P_{t+1|t}^{(i)} H'^T). \quad (5.35)$$

After this step  $x^p$  is once again a point distribution  $x_{t+1|t}^{p(i)} = \xi_{t+1}^{(i)}$  and all the elements of  $P_{t+1|t}^{(i)}$  but  $P_{t+1|t}^{k(i)}$  are zeros, and the particle filter update and the compensation for the selected particle has been done in one step. Taking this structure into account it is possible to obtain a more efficient implementation, just computing  $x_{t+1|t}^k$  and  $P_{t+1|t}^k$ .

If another different proposal density for the particle filter is more suitable, this is easily incorporated by simply changing the distribution of  $\xi_{t+1}$  and then appropriately compensate the weights for this.

### Measurement Update

The next step in deriving the Rao-Blackwellized filter is to incorporate new measurements into the estimate. For the  $x^k$  part of the state this is done using (5.28a), which with the model (4.11) and results above becomes

$$\begin{aligned} p(x_{t+1}^k | \mathbb{Y}_{t+1}, \mathbb{X}_{t+1}^p) \\ = \frac{\mathcal{N}(y_{t+1}; h(x_{t+1}^p), R_t) + H_{t+1|t} x_{t+1}^k, R_t) \cdot \mathcal{N}(x_{t+1}^k; x_{t+1|t}^k, P_{t+1|t}^k)}{p(y_{t+1} | \mathbb{Y}_t, \mathbb{X}_{t+1}^p)}. \end{aligned} \quad (5.36)$$

The expression can be identified as a measurement update in a Kalman filter,

$$x_{t+1|t+1}^{(i)} = x_{t+1|t}^{(i)} + K_{t+1}^{(i)} (y_{t+1} - H_{t+1|t}^{(i)} x_{t+1|t}^{(i)} - h(x_{t+1|t}^{p(i)})) \quad (5.37a)$$

$$P_{t+1|t+1}^{(i)} = P_{t+1|t}^{(i)} - K_{t+1}^{(i)} S_{t+1}^{(i)} K_{t+1}^{(i)T} \quad (5.37b)$$

$$K_{t+1}^{(i)} = P_{t+1|t}^{(i)} \bar{H}_{t+1}^{(i)T} (S_{t+1}^{(i)})^{-1}$$

$$S_{t+1}^{(i)} = \bar{H}_{t+1}^{(i)} P_{t+1|t}^{(i)} \bar{H}_{t+1}^{(i)T} + R.$$

The particle filter part of the state space is handled using (5.28b),

$$\begin{aligned} p(\mathbb{X}_{t+1}^p | \mathbb{Y}_{t+1}) &= \frac{\int p(y_{t+1} | x_{t+1}^k, x_{t+1}^p) p(x_{t+1}^k | \mathbb{Y}_t, \mathbb{X}_{t+1}^p) dx_{t+1}^k p(\mathbb{X}_{t+1}^p | \mathbb{Y}_t)}{p(y_{t+1} | \mathbb{Y}_t)} \\ &= \frac{\int \mathcal{N}(y_{t+1}; \bar{H}_{t+1} x_{t+1} + h_{t+1}, R) \mathcal{N}(x_{t+1}^k | x_{t+1|t}^k, P_{t+1|t}^k) dx_{t+1}^k p(\mathbb{X}_{t+1}^p | \mathbb{Y}_t)}{p(y_{t+1} | \mathbb{Y}_t)} \\ &= \frac{\mathcal{N}(y_{t+1}; \bar{H}_{t+1} x_{t+1} + h_{t+1}, \bar{H}_{t+1} P_{t+1|t} \bar{H}_{t+1}^T + R) p(\mathbb{X}_{t+1}^p | \mathbb{Y}_t)}{p(y_{t+1} | \mathbb{Y}_t)}, \end{aligned} \quad (5.38)$$

which is used to update the particle weights

$$\omega_{t+1|t+1}^{(i)} \propto \mathcal{N}(y_{t+1}; \bar{H}_{t+1} x_{t+1} + h_{t+1}, \bar{H}_{t+1} P_{t+1|t} \bar{H}_{t+1}^T + R) \omega_{t+1|t}^{(i)}. \quad (5.39)$$

This step completes the recursion, however, as with the particle filter a resampling step is needed in order to get the method to work.

**Algorithm 5.12** Rao-Blackwellized PF (Filter bank formulation)

For the system

$$\begin{aligned}x_{t+1} &= F(x_t^p)x_t + f(x_t^p) + G(x_t^p)w_t \\y_t &= \bar{H}(x_t^p)x_t + h(x_t^p) + e_t,\end{aligned}$$

cf. (4.11) in Section 4.2.3.

1: Initialization: For  $i = 1, \dots, N$ , let  $x_{0|-1}^{(i)} = \begin{pmatrix} x_{0|-1}^{p(i)} \\ x_{0|-1}^{k(i)} \end{pmatrix}$ , where  $x_{0|-1}^{p(i)} \sim p_{x_0^p}(x_0^p)$  and

$$x_{0|-1}^{k(i)} = x_0^k, P_{0|-1}^{(i)} = \begin{pmatrix} 0 & 0 \\ 0 & P_0 \end{pmatrix}. \text{ Let } t := 0.$$

2: Measurement update

$$\begin{aligned}\omega_{t|t}^{(i)} &\propto \mathcal{N}(y_t; \hat{y}_t^{(i)}, S_t^{(i)}) \cdot \omega_{t|t-1}^{(i)} \\x_{t|t}^{(i)} &= x_{t|t-1}^{(i)} + K_t^{(i)}(y_t - \hat{y}_t^{(i)}) \\P_{t|t}^{(i)} &= P_{t|t-1}^{(i)} - K_t^{(i)}S_t^{(i)}K_t^{(i)T},\end{aligned}$$

with  $\hat{y}_t^{(i)} = h(x_{t|t-1}^{p(i)}) + \bar{H}_{t|t-1}^{(i)}x_{t|t-1}^{(i)}$ ,  $S_t^{(i)} = \bar{H}_{t|t-1}^{(i)}P_{t|t-1}^{(i)}\bar{H}_{t|t-1}^{(i)T} + R$ , and  $K_t^{(i)} = P_{t|t-1}^{(i)}\bar{H}_{t|t-1}^{(i)T}(S_t^{(i)})^{-1}$ .

3: Prune/resample the filter bank according to the techniques in Section 5.2.2.

4: Time update

$$\begin{aligned}x'_{t+1|t} &= F_t^{(i)}x_{t|t}^{(i)} + f(x_t^{p(i)}) \\P'_{t+1|t} &= P_{t+1|t}^{(i)} + F_t^{(i)}P_{t|t}^{(i)}F_t^{(i)T} + G_t^{(i)}Q_tG_t^{(i)T}.\end{aligned}$$

For  $\xi_{t+1}^{(i)} \sim \mathcal{N}(H'x'_{t+1|t}, H'P'_{t+1|t}H'^T)$ , where  $H' = \begin{pmatrix} I & 0 \end{pmatrix}$ , do:

$$\begin{aligned}x_{t+1|t}^{(i)} &= x'_{t+1|t} + P'_{t+1|t}H'^T(H'P'_{t+1|t}H'^T)^{-1}(\xi_{t+1}^{(i)} - H'x'_{t+1|t}) \\P_{t+1|t}^{(i)} &= P'_{t+1|t} - P'_{t+1|t}H'^T(H'P'_{t+1|t}H'^T)^{-1}H'P'_{t+1|t}.\end{aligned}$$

5: Increase time and repeat from step 2.

## Resampling

As with the particle filter, if the described RBPF is run with exactly the steps described above it will end up with all particle weight in one single particle. This degrades estimation performance. The solution is to resample, randomly get rid of unimportant particles and replace them with more likely ones. In the RBPF this is done in exactly the same way as described for the particle filter, with the difference that when a particle is selected so is the full state matching that particle, as well as the covariance matrix describing the Kalman filter part of the state. See Section 5.2.2 for details about resampling.

The complete Rao-Blackwellized particle filter algorithm is given in Algorithm 5.12.

### Filter Bank Interpretation

Recall the main ideas of the Kalman filter bank presented in Section 5.5. The model is such that it is possible to, conditioned on a mode parameter  $\delta_t$ , enumerate a set of different behaviors of the system. An optimal filter can then be obtained by running a filter for each mode, and then combine the different filters to a global estimate. A problem is the exponential growth of modes. This is solved with approximations that reduce the number of modes. Generic algorithms are given in Algorithms 5.9 and 5.10. This is in many aspects very similar to how the RBPF works, as seen in the following comparison.

In the state representation used in the RBPF setup, the  $x^p$  part of the state vector conditions the system to be linear and Gaussian, and does in that way act just as the mode  $\delta$  in the filter bank. One difference is that  $x^p$  is kept in the state vector, whereas  $\delta$  is kept separate, and also that  $\delta$  can be enumerated in contrast to the real valued vector  $x^p$ . The fact that  $x^p$  cannot be enumerated, makes it impossible to use classical filter bank techniques. However, introducing some ideas from particle filtering, the RBPF can be seen as a Kalman filter bank with randomly chosen modes.

Given this background, the RBPF algorithm matches the Kalman filter bank algorithm, making the following connections:

- The  $x^p$  part of the state space should be interpreted as the system mode,  $\delta$ , kept in the state vector.
- The filter weight computation in the filter bank corresponds and the RBPF is basically the same operation.
- The time update in the RBPF is the same as the filter time update in the Kalman filter bank. Note that  $x^p$  becomes real valued here, and hence not completely corresponds to  $\delta$  at this point.
- $x^p$  is returned to a mode indicator again in the second step in the time update as it is sampled from the distribution given by the previous step. This can be viewed as a branching step since the infinite set of  $x^p$  values is reduced to a finite set again. In the process,  $x^k$  is compensated to reflect the specific choices of  $x^p$ .

Viewed this way Algorithm 5.12 describes a Kalman filter bank with stochastic branching and pruning. Gaining this understanding of the RBPF can be very useful. One benefit is that it gives a different view of what happens in the algorithm, another benefit is that it enables for efficient implementations of the RBPF in general filtering frameworks without having to introduce new concepts which would increase the code complexity and at the same time introduces redundant code. The initial idea for this formulation of the RBPF was derived when trying to incorporate the filter into F++, the filtering framework presented in Chapter 10.

There are mainly two reasons to use a RBPF instead of a regular particle filter when it is possible. One reason is the performance gain obtained from the Rao-Blackwellization itself, however often even more important is that by reducing the dimension of the state space where particles are used to represent the PDF, it is possible to use less particles and still maintaining the same performance. This can be used either to improve filter performance, or to speed up the filter using less particles still maintaining the same filter

performance. In [75] the authors compare the number of particles needed to obtain equivalent performance using different divisions of the state space in particle filter states and Kalman filter states.

With further restrictions on the model used it is possible to make simplifications to drastically reduce the computational complexity. One such example is if the model is such that the linear Gaussian part is independent and it is enough to compute only one  $P$  for the all the Kalman filters. This reduces the computational complexity considerably.

# 6

---

## Cramér-Rao Lower Bound

IT IS OFTEN OF interest to know how well a parameter or a state can be estimated. Consider the bearings-only example described in Chapter 1 again. This time assume that the measured object is an obstacle, and that the position of it is used to guide a vehicle past it without crashing into it. The shortest, and hence cheapest, path is to go as close to the obstacle as possible. Possible questions are then:

- How close is it safe to go to the estimated obstacle?
- Is it possible to improve the estimate of the obstacle position in some way so that it is safe to go closer to it?
- Is it theoretically possible to fulfill the accuracy constraints without adding new or better sensors?

This chapter studies the *Cramér-Rao Lower Bound* (CRLB) to try and answer these questions.

As mentioned in Section 2.2.1, the CRLB is a lower bound on the variance of any unbiased estimator  $\hat{\theta}$  of a parameter, given by the inverse Fisher information,

$$\text{cov}(\hat{\theta}) \succeq \mathcal{I}^{-1}(\theta) = P_{\theta}^{\text{CRLB}}.$$

This chapter extends the CRLB to describe estimation of the state in dynamic systems; first for deterministic trajectories, where the *parametric Cramér-Rao lower bound*, Section 6.1, can be used, and then for stochastic trajectories using the theory for the *posterior Cramér-Rao lower bound*, Section 6.2. It is then, in Section 6.3, shown that for linear systems, the parametric and the posterior CRLB bounds yield identical limits in terms of the intrinsic accuracy of the involved noise distributions. Comparing the CRLB for the information in the true noise PDF and a Gaussian approximation makes it possible to see how much can be gained from using nonlinear estimation methods.

## 6.1 Parametric Cramér-Rao Lower Bound

The *parametric Cramér-Rao lower bound* is a natural extension to the CRLB for parameter estimation. Basically, treat the state at each time as a parameter with a deterministic but unknown value (more precisely treat each process noise realization,  $w_t$ , as a parameter from which the state can then be found) and derive a bound for how well these parameters can be estimated. The subject is thoroughly treated in e.g., [15, 138], therefore only the resulting theorem is given here.

### Theorem 6.1 (Parametric Cramér-Rao lower bound)

The parametric Cramér-Rao lower bound for a one-step-ahead prediction

$$P_{t+1|t} \preceq \mathbb{E}_{\hat{x}_{t+1|t}} \left( (\hat{x}_{t+1|t} - x_{t+1}^0)(\hat{x}_{t+1|t} - x_{t+1}^0)^T \right)$$

and filtering,

$$P_{t|t} \preceq \mathbb{E}_{\hat{x}_{t|t}} \left( (\hat{x}_{t|t} - x_t^0)(\hat{x}_{t|t} - x_t^0)^T \right),$$

for the system (4.3) is given by the recursion:

$$\begin{aligned} P_{t|t} &= P_{t|t-1} - P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1} H_t P_{t|t-1} \\ P_{t+1|t} &= F_t P_{t|t} F_t^T + G_t Q_t G_t^T, \end{aligned}$$

initialized with  $P_{0|-1}^{-1} = \mathcal{I}_{x_0}$  and with

$$\begin{aligned} F_t^T &= \nabla_{x_t} f(x_t, w_t^0) \Big|_{x_t=x_t^0}, & G_t^T &= \nabla_{w_t} f(x_t^0, w_t) \Big|_{w_t=w_t^0}, \\ H_t^T R_t^{-1} H_t &= -\mathbb{E}_{y_t} \left( \Delta_{x_t}^{x_t} p(y_t | x_t) \right), & Q_t^{-1} &= -\mathbb{E}_{x_t} \left( \Delta_{w_t}^{w_t} p(x_t, w_t) \Big|_{w_t=w_t^0} \right), \end{aligned}$$

where superscript  $0$  denotes the true value, and the factorization between  $H_t$  and  $R_t$  is chosen such that  $R_t \succ 0$ .

The bounds are valid if all involved gradients, expectations, and matrix inverses exist. Explicitly this means that  $p(x_{t+1}|x_t)$  must be well defined for all  $t$ .

**Proof:** For the prediction bound, see Theorem 4.3 in [15]. The proof of the filtering bound follows with only minor modifications.  $\square$

The recursions in Theorem 6.1 should be recognized as the *Riccati* recursions used in the EKF to update the error covariance, (5.16). The difference lies not in the mathematical expressions themselves, but in the exact matrices involved. The matrices are however related, as the naming scheme suggests. If all involved distributions are Gaussian, the expressions are exactly the same as for the EKF if it is fed with the correct state trajectory. This connection is shown in [138], and is useful when it comes to evaluating a filter compared to the parametric CRLB. Simply run the EKF and feed it with the correct states to get the parametric CRLB. With the CRLB available Monte Carlo simulations is a common technique to derive the performance of the filter. The idea is to use Monte Carlo integration over enough different measurement noise instances to use the *mean square error* (MSE) instead of the variance of the error. One way to do this is given in Algorithm 6.1.

The results in Theorem 6.1 are in [15] extended to cover multi-step prediction and smoothing, too. The results are similar to those presented above in their resemblance to the covariance propagation in the EKF.

**Algorithm 6.1** Parametric Mean Square Error

1. Generate *one* true state trajectory,  $\mathbb{X}$ .
2. Based on the trajectory  $\mathbb{X}$ , generate  $M \gg 1$  independent measurement sequences,  $\mathbb{Y}^{(i)}$  for  $i = 1, \dots, M$ .
3. Apply the filter to the measurements to get  $M$  estimated trajectories,  $\hat{\mathbb{X}}^{(i)}$ .
4. How well the filter performs compared to the parametric CRLB,  $P_t^{\text{CRLB}}$ , is now given by,

$$\frac{1}{M} \sum_{i=1}^M (\hat{x}_t^{(i)} - x_t)(\hat{x}_t^{(i)} - x_t)^T \underset{\sim}{\succ} P_t^{\text{CRLB}},$$

where  $\underset{\sim}{\succ}$  indicates that for finite  $M$  the MSE is only approximately bounded by the CRLB.

## 6.2 Posterior Cramér-Rao Lower Bound

The parametric CRLB is insufficient in a Bayesian estimation setting since it assumes a true state trajectory, which is inconsistent with the Bayesian philosophy. The introduction of the *posterior Cramér-Rao lower bound* solves this philosophical dilemma. The posterior CRLB differs from the parametric CRLB in that it does not assume a true trajectory, hence it is acceptable from a Bayesian perspective. The posterior CRLB is more frequently treated in the literature, some references are [21, 31, 36, 37, 140]. As with the parametric CRLB, [15] provides a thorough description and derivation of the posterior CRLB as well. Therefore, Theorem 6.2 is given here without further derivation.

### Theorem 6.2 (Posterior Cramér-Rao lower bound)

*The one-step-ahead posterior Cramér-Rao lower bound for the model (4.3) is given by the recursion*

$$P_{t+1|t}^{-1} = Q_t^{-1} - S_t^T (P_{t|t-1}^{-1} + R_t^{-1} + V_t)^{-1} S_t$$

*initiated with  $P_{0|-1}^{-1} = \mathcal{I}_{x_0}^{-1}$ , and the filtering bound by*

$$P_{t+1|t+1}^{-1} = Q_t^{-1} + R_{t+1}^{-1} - S_t^T (P_{t|t}^{-1} + V_t)^{-1} S_t$$

*initiated with  $P_{0|0}^{-1} = (P_{0|-1}^{-1} + R_0^{-1})^{-1}$ , both recursions use the system dependent matrices:*

$$\begin{aligned} V_t &= -\mathbb{E}_{x_t, w_t} \left( \Delta_{x_t}^{x_t} \log p(x_{t+1}|x_t) \right), & R_t^{-1} &= -\mathbb{E}_{x_t, y_t} \left( \Delta_{x_t}^{x_t} \log p(y_t|x_t) \right), \\ S_t &= -\mathbb{E}_{x_t, w_t} \left( \Delta_{x_t}^{x_{t+1}} \log p(x_{t+1}|x_t) \right), & Q_t^{-1} &= -\mathbb{E}_{x_t, w_t} \left( \Delta_{x_{t+1}}^{x_{t+1}} \log p(x_{t+1}|x_t) \right). \end{aligned}$$

*For the bounds to be valid the involved differentiations and expectations must exist, however there is no need for  $R_t$  and  $Q_t$  to exist as long as the matrices  $R_t^{-1}$  and  $Q_t^{-1}$  do. Explicitly this means that  $p(x_{t+1}|x_t)$  must be well defined for all  $t$ .*

**Proof:** See the proof of Theorem 4.5 in [15]. □

As with Theorem 6.1, [15] elaborates further on the posterior CRLB and gives expressions for multi-step prediction and smoothing.

One observation that should be made is that the posterior CRLB is often harder to evaluate than the parametric one because more stochastic dimensions are introduced. However, explicit expressions can be found in some special cases. One such case is systems with linear dynamics. For situations when no explicit posterior CRLB expressions exist, [139] suggests a method based on Monte Carlo simulations, similar to the particle filter, to compute the CRLB numerically.

To evaluate a filter against the posterior CRLB Monte Carlo simulations can be used in a way similar to what is described for the parametric CRLB. However, since the trajectories are also random, different trajectories must be generated for each measurement sequence. See Algorithm 6.2.

---

**Algorithm 6.2** Posterior Mean Square Error

---

1. Generate  $M \gg 1$  independent trajectories,  $\mathbb{X}^{(i)}$  for  $i = 1, \dots, M$ .
2. Generate  $M$  measurement sequences, where  $\mathbb{Y}^{(i)}$  is based on the trajectory  $\mathbb{X}^{(i)}$ .
3. Apply the filter to the measurements to get  $M$  estimated trajectories,  $\hat{\mathbb{X}}^{(i)}$ .
4. How well the filter performs compared to the parametric CRLB,  $P^{\text{CRLB}}$ , is now given by,

$$\frac{1}{M} \sum_{i=1}^M (\hat{x}_t^{(i)} - x_t^{(i)})(\hat{x}_t^{(i)} - x_t^{(i)})^T \succsim P_t^{\text{CRLB}},$$

where  $\succsim$  indicate that for finite  $M$  the MSE is only approximately bounded by the CRLB.

---

### 6.3 Cramér-Rao Lower Bounds for Linear Systems

The sequel of this chapter is devoted to studies of the parametric and posterior CRLB of linear systems, *i.e.*, models of the type introduced in Section 4.2.1. Linear models are studied because they allow for analytical analysis and because the Kalman filter offers the BLUE for these systems. Hence, it is easy to compare optimal linear performance to the performance bound given by the CRLB. The understanding of the linear models derived this way can then be used to get better understanding of nonlinear systems, where the mathematics is usually more involved and less intuitive.

First, recall the linear model (4.4) (here without any deterministic input to simplify the notation),

$$x_{t+1} = F_t' x_t + G_t' w_t, \tag{6.1a}$$

$$y_t = H_t' x_t + e_t, \tag{6.1b}$$

with  $\text{cov}(w_t) = Q'_t$  and  $\text{cov}(e_t) = R'_t$ . Note the 's used to distinguish between the system parameters and the quantities used in the CRLB expressions.

For this system, derive an expression for the parametric CRLB using Theorem 6.1. The expressions needed are:

$$\begin{aligned} F_t^T &= \nabla_{x_t} f(x_t, w_t^0) \Big|_{x_t=x_t^0} = F_t'^T \\ G_t^T &= \nabla_{w_t} f(x_t^0, w_t) \Big|_{w_t=w_t^0} = G_t'^T \\ H_t^T R_t^{-1} H_t &= -\mathbb{E}_{y_t} \left( \Delta_{x_t}^{x_t} p(y_t|x_t) \right) = H_t'^T \mathcal{I}_{e_t} H_t' \\ Q_t^{-1} &= -\mathbb{E}_{x_t} \left( \Delta_{w_t}^{w_t} p(x_t, w_t) \Big|_{w_t=w_t^0} \right) = \mathcal{I}_{w_t}. \end{aligned}$$

In conclusion, for linear Gaussian systems, the naming convention in the linear model (4.4) maps directly onto the parametric CRLB notation in Theorem 6.1. For linear non-Gaussian systems, the system matrices still map directly, however, the covariances in the system model should be substituted for the inverse information of the stochastic variable. The inverse can in a way be interpreted as the effective covariance in a filter which is optimal in the sense that it reaches the CRLB.

Using the expressions to calculate the CRLB yields the recursion

$$\begin{aligned} P_{t|t} &= P_{t|t-1} - P_{t|t-1} H_t'^T (H_t' P_{t|t-1} H_t'^T + \mathcal{I}_{e_t}^{-1})^{-1} H_t' P_{t|t-1} \\ &= (P_{t|t-1}^{-1} + H_t'^T \mathcal{I}_{e_t} H_t')^{-1}, \end{aligned} \quad (6.2a)$$

$$P_{t+1|t} = F_t' P_{t|t} F_t'^T + G_t' \mathcal{I}_{w_t}^{-1} G_t'^T \quad (6.2b)$$

initiated with  $P_{0|-1} = \mathcal{I}_{x_0}^{-1}$ . The second form of  $P_{t|t}$  follows from applying the matrix inversion lemma once. Note that this is the same Riccati recursion as used for the error covariances in the Kalman filter.

Next, the posterior CRLB for the linear model (6.1) is derived. The expressions used in Theorem 6.2 simplify to

$$\begin{aligned} Q_t^{-1} &= \mathcal{I}_{\bar{w}_t} = (G_t' \mathcal{I}_{w_t}^{-1} G_t')^{-1}, \\ R_t^{-1} &= H_t'^T \mathcal{I}_{e_t} H_t', \\ S_t &= -F_t'^T \mathcal{I}_{\bar{w}_t} = -F_t'^T (G_t' \mathcal{I}_{w_t}^{-1} G_t')^{-1}, \\ V_t &= F_t'^T \mathcal{I}_{w_t} F_t' = F_t'^T (G_t' \mathcal{I}_{w_t}^{-1} G_t')^{-1} F_t', \end{aligned}$$

where  $\bar{w}_t := G_t' w_t$ . Detailed derivations, in terms of  $\mathcal{I}_{\bar{w}_t}$ , can be found in [15], and the last equalities follows from applying Theorem 2.2. Substituting these into the posterior CRLB expressions yields,

$$\begin{aligned} P_{t+1|t}^{-1} &= \mathcal{I}_{\bar{w}_t} - (-F_t'^T \mathcal{I}_{\bar{w}_t})^T (P_{t|t-1}^{-1} + H_t'^T \mathcal{I}_{e_t} H_t' + F_t'^T \mathcal{I}_{\bar{w}_t} F_t')^{-1} (-F_t'^T \mathcal{I}_{\bar{w}_t}) \\ &= (\mathcal{I}_{\bar{w}_t}^{-1} + F_t' (P_{t|t-1}^{-1} + H_t'^T \mathcal{I}_{e_t} H_t')^{-1} F_t'^T)^{-1} \\ &= (G_t' \mathcal{I}_{w_t}^{-1} G_t' + F_t' (P_{t|t-1}^{-1} + H_t'^T \mathcal{I}_{e_t} H_t')^{-1} F_t'^T)^{-1} \end{aligned} \quad (6.3a)$$

for the prediction bound and for the filtering bound

$$\begin{aligned}
P_{t|t}^{-1} &= \mathcal{I}_{\bar{w}_t} + H_t'^T \mathcal{I}_{e_t} H_t' - (-F_t'^T \mathcal{I}_{\bar{w}_t})^T (P_{t-1|t-1}^{-1} + F_t'^T \mathcal{I}_{\bar{w}_t} F_t')^{-1} (-F_t'^T \mathcal{I}_{\bar{w}_t}) \\
&= H_t'^T \mathcal{I}_{e_t} H_t' + (\mathcal{I}_{\bar{w}_t}^{-1} + F_t' P_{t-1|t-1} F_t'^T)^{-1} \\
&= H_t'^T \mathcal{I}_{e_t} H_t' + (G_t \mathcal{I}_{w_t}^{-1} G_t^T + F_t' P_{t-1|t-1} F_t'^T)^{-1}. \tag{6.3b}
\end{aligned}$$

Inspection of the expressions reveals that the same recursive expression as for the parametric CRLB can be used to calculate the posterior CRLB for linear systems. Hence, for linear systems the parametric CRLB and posterior CRLB yield exactly the same bounds. That is, in this case the results obtained using a classical view of estimation and a Bayesian view are the same. The result follows because the state does not change the dynamics or the measurement equation of the system, hence the specific trajectory does not change the gain in the filter. If the state influences the dynamics or measurements of the system then the uncertainty of the exact system makes the estimation more difficult using the Bayesian approach. Note that for nonlinear systems, where the effective system depends on the state, the bounds should be expected to differ.

### Theorem 6.3 (Cramér-Rao Lower Bounds For Linear Systems)

For linear systems (4.4) the parametric and posterior Cramér-Rao lower bound yield the same lower bound. This lower bound is given by the recursion

$$\begin{aligned}
P_{t|t} &= (P_{t|t-1}^{-1} + H_t^T \mathcal{I}_{e_t} H_t)^{-1}, \\
P_{t+1|t} &= F_t P_{t|t} F_t^T + G_t \mathcal{I}_{w_t}^{-1} G_t^T,
\end{aligned}$$

initiated with  $P_{0|-1} = \mathcal{I}_{x_0}^{-1}$ .

**Proof:** The result follows from Theorems 6.1 and 6.2 and the derivation of the parametric and the posterior CRLB for a linear system as done above.  $\square$

### Corollary 6.1

If the linear system is time invariant and there exists a bounded asymptotic Cramér-Rao lower bound then the prediction bound  $\bar{\kappa}_+(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1}) = \lim_{t \rightarrow +\infty} P_{t|t-1}$  is given as the solution of

$$\bar{\kappa}_+(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1}) = F \left( \bar{\kappa}_+(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1}) + H_t^T \mathcal{I}_{e_t} H_t \right)^{-1} F^T + G \mathcal{I}_{w_t}^{-1} G^T,$$

and the filtering bound  $\bar{\kappa}(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1}) = \lim_{t \rightarrow +\infty} P_{t|t}$  is given as the solution of

$$\bar{\kappa}(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1}) = \left( \bar{\kappa}_+(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1}) + H_t^T \mathcal{I}_{e_t} H_t \right)^{-1}.$$

**Proof:** If asymptotic solutions exist as  $t \rightarrow \infty$ , the bounds are  $P_{t+1|t} = P_{t|t-1} =: \bar{\kappa}_+(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1})$  and  $P_{t|t} = P_{t-1|t-1} =: \bar{\kappa}(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1}) \bar{P}$ . The result then follows from using  $\bar{\kappa}_+(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1})$  and  $\bar{\kappa}(\mathcal{I}_{e_t}^{-1}, \mathcal{I}_{e_t}^{-1})$  in Theorem 6.3.  $\square$

Conditions for when a stationary solution exists can be found in [69].

## Linear Cramér-Rao Lower Bound Properties

For linear systems the noise characteristics have a direct impact on the CRLB that is obtained. It is possible to gain a better understanding for the mechanisms in play by studying the effects of the *intrinsic accuracy* (IA, see Chapter 2) to the CRLB. Lemma 6.1 can be used for this purpose.

### Lemma 6.1

For matrices  $Q_t \succeq 0$ ,  $R_t \succ 0$ , and  $P \succ 0$  following is true:

- (i) If  $\tilde{Q} \preceq Q$  and  $\tilde{P} \preceq P$  this is preserved through a time update,

$$F\tilde{P}F^T + G\tilde{Q}G^T \preceq FPF^T + GQG^T,$$

with equality if and only if  $G\tilde{Q}G^T = GQG^T$  and  $F\tilde{P}F^T = FPF^T$ .

- (ii) If  $\tilde{R} \preceq R$  and  $\tilde{P} \preceq P$  the relation is preserved through a measurement update,

$$(\tilde{P}^{-1} + H^T \tilde{R}^{-1} H)^{-1} \preceq (P^{-1} + H^T R^{-1} H)^{-1},$$

with equality if and only if  $H^T \tilde{R}^{-1} H = H^T R^{-1} H$  and  $\tilde{P} = P$ .

**Proof:** To show property (i) compare the two expressions,

$$(FPF^T + GQG^T) - (F\tilde{P}F^T + G\tilde{Q}G^T) = F(P - \tilde{P})F^T + G(Q - \tilde{Q})G^T \succeq 0,$$

since  $\tilde{P} \preceq P$  and  $\tilde{Q} \preceq Q$ .

Property (ii) follows in a similar manner,

$$\begin{aligned} (P^{-1} + H^T R^{-1} H) - (\tilde{P}^{-1} + H^T \tilde{R}^{-1} H) \\ = (P^{-1} - \tilde{P}^{-1}) + H^T (R^{-1} - \tilde{R}^{-1}) H \succeq 0, \end{aligned}$$

since  $A \succ 0$  and  $B \succ 0$  implies  $A \succeq B \Leftrightarrow A^{-1} \preceq B^{-1}$ , and  $P \succeq \tilde{P}$  and  $Q \succeq \tilde{Q}$ . This in turn provides the inequality.  $\square$

In Lemma 6.1, property (i) describes how the state estimate loses information in the time update step if a less informative process noise is introduced, and property (ii) how information is lost when introducing less informative measurements. In common for both time and measurement updates is that if all involved stochastic contributions are scaled then the result of the update is scaled with the same factor. Another result is that if the information content in a stochastic variable is increased the CRLB for the estimate is improved as well, unless the information is gained in a direction that is neglected by the system. However, introducing more informative noise will never have a negative effect on the CRLB. Furthermore, if a system is affected by non-Gaussian noise the CRLB is likely to improve since according to Theorem 2.1 the Gaussian distribution is the least informative distribution.

The results above deal with individual steps in the recursive formulas used to calculate the Kalman filter and CRLB covariance matrices. By repeatedly applying Lemma 6.1, the results in Theorem 6.4, regarding for the CRLB at a specific time, can be derived.

**Theorem 6.4**

Assume a linear system (4.4), with inverse intrinsic noise accuracies  $Q_\tau \succeq 0$ ,  $R_\tau \succ 0$ , and  $P_{0|-1} \succeq 0$  for  $\tau \leq t$ , and for which the Cramér-Rao lower bounds at time  $t$  are  $P_{t|t}$  and  $P_{t+1|t}$ . The following two properties are true.

- If all  $Q_\tau$  and  $R_\tau$  are changed to  $\gamma Q_\tau$  and  $\gamma R_\tau$ , respectively, and  $P_{0|-1}$  to  $\gamma P_{0|-1}$ , the CRLB of the resulting system are  $\gamma P_{t|t}$  and  $\gamma P_{t+1|t}$ .
- For a new system with the same dynamics as the first and with  $Q_\tau \succeq \tilde{Q}_\tau$ ,  $R_\tau \succeq \tilde{R}_\tau$ , and  $P_{0|-1} \succeq \tilde{P}_{0|-1}$  the CRLB of the two systems relates as  $P_{t|t} \succeq \tilde{P}_{t|t}$  and  $P_{t+1|t} \succeq \tilde{P}_{t+1|t}$  where the  $\tilde{\cdot}$  indicate a property of the new system.

**Proof:** The first property follows since  $\gamma$  factors out in all the filtering steps, and the second property from recursively apply Lemma 6.1.  $\square$

The first part of Theorem 6.4 states that if all intrinsic accuracies in the system is scaled with the same factor, i.e., have the same relative accuracy, the resulting CRLB will also be scaled with the same factor. The second part states that if any intrinsic accuracy is improved, i.e., a noise becomes more informative, this can only improve the resulting CRLB and unless the direction of the improvement is ignored by the system the CRLB will improve. Since non-Gaussian noise is more informative than Gaussian noise, having non-Gaussian distributions in the system description will potentially improve the estimation performance.

For time invariant linear systems the stationary performance is given by the Riccati equation if a stationary bound exists.

**Corollary 6.2**

If the linear system is time invariant and an asymptotic Cramér-Rao lower bound exists, denote  $\bar{\kappa}_+(Q, R) := \lim_{t \rightarrow +\infty} P_{t+1|t}$  and  $\bar{\kappa}(Q, R) := \lim_{t \rightarrow +\infty} P_{t|t}$ , where  $Q$  and  $R$  are the parameters in the CRLB recursion, then:

- For  $\gamma > 0$

$$\gamma \bar{\kappa}_+(Q, R) = \bar{\kappa}_+(\gamma Q, \gamma R) \quad \text{and} \quad \gamma \bar{\kappa}(Q, R) = \bar{\kappa}(\gamma Q, \gamma R).$$

- If  $\tilde{Q} \preceq Q$  and  $\tilde{R} \preceq R$  then

$$\bar{\kappa}_+(\tilde{Q}, \tilde{R}) \preceq \bar{\kappa}_+(Q, R) \quad \text{and} \quad \bar{\kappa}(\tilde{Q}, \tilde{R}) \preceq \bar{\kappa}(Q, R).$$

**Proof:** Follows immediately from Theorem 6.4 given that asymptotic solutions exist.  $\square$

Using the notation in Corollary 6.2,  $\bar{\kappa}(Q, R)$  denotes:

- asymptotic CRLB for an (implicitly defined) linear time-invariant system if  $Q = \mathcal{I}_w^{-1}$  and  $R = \mathcal{I}_e^{-1}$ , and
- the BLUE performance if  $Q = \text{cov}(w)$  and  $R = \text{cov}(e)$ .

At the same time,  $\bar{\kappa}(Q, R)$  is the solution to a Riccati equation which makes it fairly easy to compute with standard methods. This makes  $\bar{\kappa}(Q, R)$  a good quantity to study when deciding if a nonlinear filtering method should be evaluated or not.

## 6.4 Summary

This chapter has introduced the CRLB as lower bound for performance in terms of the second order statistics of the estimate. The main focus is in how the intrinsic accuracy of the involved noise affects the performance bound. Expressions are given for this for linear systems. The CRLB can then be compared to tell if requirements are feasible given the measurements available, and when compared to achieved filter performance as tuning tool. The following chapter explores this further in simulation studies.



# 7

---

## Filter Performance

**T**HE PURPOSE OF THIS chapter is twofold: The first purpose is to illustrate and verify the *Cramér-Rao lower bound* (CRLB) theory developed in Chapter 6 and show how the theory can be used to beforehand indicate if, for a linear system, a nonlinear filter pays off or if a Kalman filter will suffice. The second purpose is to by example motivate why second order statistics and CRLB should not be used as the sole performance criteria when comparing different filters. The Kullback divergence, which was introduced in Chapter 2, is suggested as an alternative. Simulations are provided to motivate this choice.

Analyzing nonlinear systems is much more difficult than the analyzing linear systems, and hence extensive simulation studies are used instead. Simulations are provided to show both the differences between the used methods and hopefully to give some intuition for the problem that can be applied to other problems. However, it is impossible to draw any advance general conclusions from a limited set of simulations.

This chapter is organized in the following way: Section 7.1 performs simulations with a system with multimodal posterior distribution where the second order statistics are rather uninformative. The simplified altitude based terrain navigation example in Section 7.2 exemplifies a case when proper modeling of the measurement noise pays off. Section 7.3, a range-only measurement application, and Section 7.4, an extension of the introductory bearings-only measurement example, both focus on the Kullback divergence as an alternative to second order statistics. Section 7.5, a DC motor simulation, and Section 7.6, a target tracking application, have focus set on analysis based on CRLB. The obtained results are summarized in Section 7.7.

## 7.1 Multimodal Posterior

The first simulation study features a relatively simple linear system,

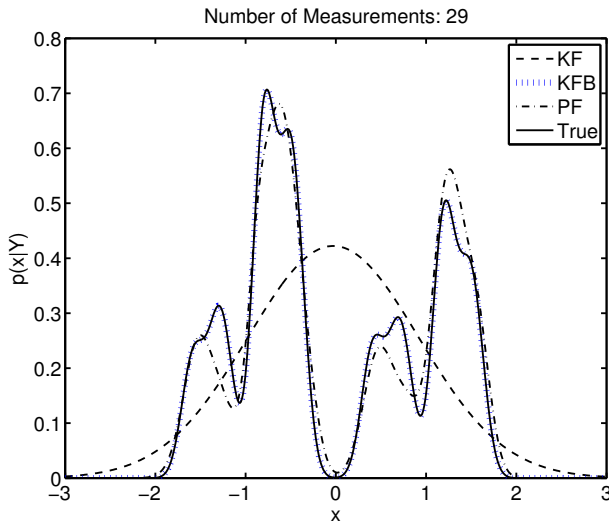
$$x_{t+1} = 0.4x_t + w_t \quad (7.1a)$$

$$y_t = x_t + e_t. \quad (7.1b)$$

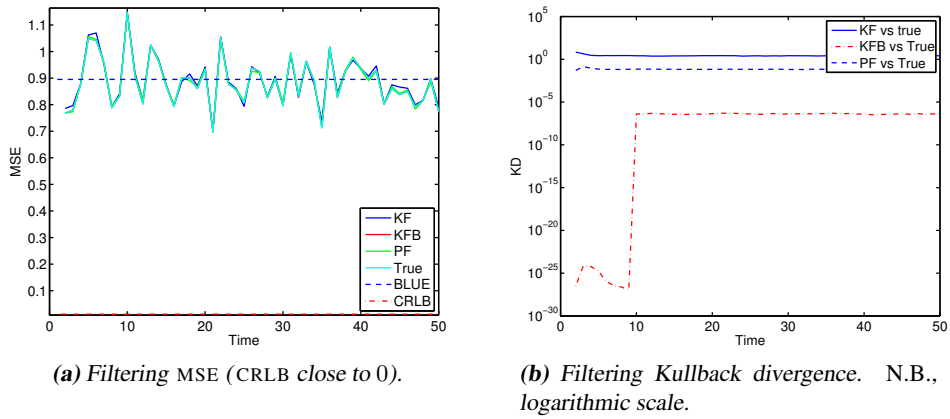
The process noise is chosen bimodal,  $w_t \sim \frac{1}{2}\mathcal{N}(-1, 0.01) + \frac{1}{2}\mathcal{N}(1, 0.01)$ , *e.g.*, representing user input. The measurements are Gaussian and uninformative,  $e_t \sim \mathcal{N}(0, 4)$ , and the initial distribution of the state is  $x_0 \sim \mathcal{N}(0, 0.1)$ . This combination of model and noise gives a posterior state distribution that tends to a multimodal Gaussian sum with approximately four modes. Figure 7.1 shows a typical posterior state distribution, and the result achieved using three different filters to approximate it. The result for the particle filter has been obtained by smoothing the particle cloud with a Gaussian kernel with low variance.

The filters used in this simulation study is a regular EKF, a *Kalman filter bank* (KFB) with  $2^3 = 8$  hypotheses (*i.e.*, a window of length  $L = 3$ ), and a SIR particle filter with 1000 particles. The reference is computed numerically using a very finely gridded point-mass filter.

The simulation study consist of 100 Monte Carlo simulations, each over 50 time steps. The *mean square error* (MSE) from the different filters are presented in Figure 7.2(a). The result from all the filters are surprisingly similar considering the clear difference in estimated posterior distribution indicated in Figure 7.1. Furthermore, there is a substantial difference between the achieved MSE, which is similar to the BLUE, and the CRLB for the system. Hence, the MSE is not able to capture the differences in the estimated posterior distributions.



**Figure 7.1:** Typical posterior state distribution for (7.1). (The Kalman filter bank (KFB) solution coincides with the true PDF, given by a PMF.)



**Figure 7.2:** Simulated MSE and Kullback divergence for (7.1).

The difference between the filters is however apparent if the Kullback divergence is computed. The Kullback divergence clearly separates the different filtering methods, see Figure 7.2(b). According to the Kullback divergence, the filter bank is better than the particle filter, which is better than the Kalman filter. This fits well with what a visual inspection of Figure 7.1 indicates. Hence, the Kullback divergence provides more information about how well  $p(x_t | \mathbb{Y}_t)$  is approximated. The reason that the Kalman filter bank outperforms the particle filter in this case is that its model assumption is correct, and that the mode history is forgotten quickly due to the factor 0.4. Hence, the filter bank reaches its result with fewer parameters to estimate than the particle filter.

## Conclusions

The example in this section shows that the MSE and CRLB do not suffice as the only measures when comparing different filtering algorithms. Using a very simple linear model, with non-Gaussian noise, it is shown how the MSE can be the same while the estimated posterior distribution  $p(x_t | \mathbb{Y}_t)$  differs substantially. This difference is however clearly captured by the Kullback divergence, which suggests it should be used in comparing different filtering options.

## 7.2 Altitude Based Terrain Navigation

The second simulation study is based on an air-borne terrain navigation application. This application has previously been studied in depth in several publications, [15, 16, 48, 104]. The same principles can also be used for other similar types of navigation, e.g., [73] and references therein. Here, the navigation system has been reduced to work in one dimension to better illustrate the theory. The goal is to position an aircraft using only altitude measurements, an altitude map of the ground, and measurements of the aircraft's acceleration. This results in an estimation problem with linear dynamics, where the aircraft is

assumed to follow a constant velocity model, but a nonlinear measurement equation,

$$x_{t+1} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} \frac{1}{2}T^2 \\ T \end{pmatrix} (w_t + a_t) \quad (7.2a)$$

$$y_t = h(x_t) + e_t, \quad (7.2b)$$

where the two states are position and velocity, and  $a_t$  is the measured acceleration. The inertial sensors in an aircraft are generally good enough to provide accurate information about accelerations, therefore it is here assumed that  $a_t = 0$  for simplicity but without loss of generality. (This is a reduced version of the model used in [104].) The function  $h$  returns the level of the ground over sea-level as a function of the position, and the measurements comes from a radar, given that the altitude of the aircraft is known. In the filter it is implemented as a look up in a database, *i.e.*, this can be expected to be very nonlinear.

The process noise in the model has the following characteristics

$$w_t \sim \mathcal{N}(0, 10^{-8}),$$

which assumes that the measurements of the aircraft's acceleration are very accurate. The following measurement noise has been used

$$e_t \sim \frac{3}{4}\mathcal{N}(0, 9) + \frac{1}{4}\mathcal{N}(12, 36),$$

with relative accuracy  $\Psi_{e_t} = 2.9$ . The Gaussian mixture represents a typical radar response from height measurements over a forest. The noise types and levels are typical for the application, [104]. See also Example 2.2 for an illustration of typical radar noise.

In the simulations the true initial state has been

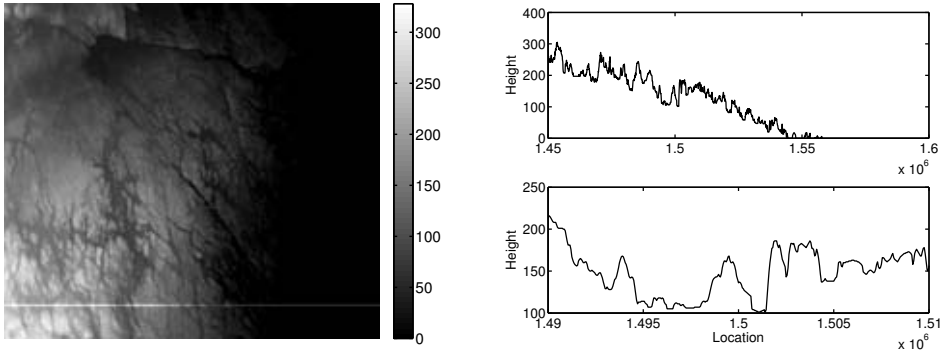
$$x_0 \sim \mathcal{N}\left(\begin{pmatrix} 1.5 \cdot 10^6 \\ 10 \end{pmatrix}, \begin{pmatrix} 10 & 0 \\ 0 & 0.01 \end{pmatrix}\right),$$

where the low velocity has been chosen to keep the aircraft within the map during the entire simulation.

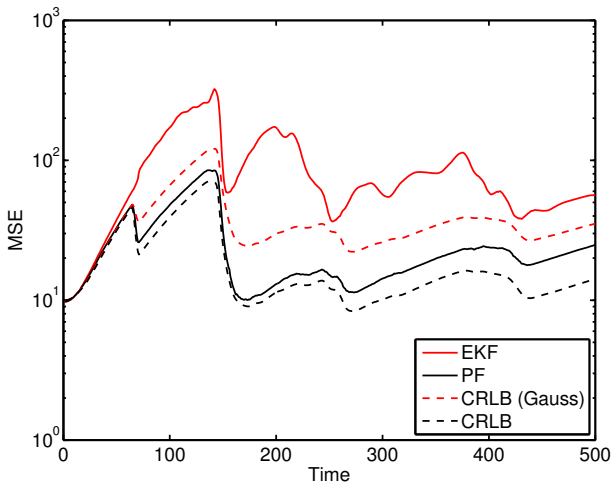
The terrain is chosen as a 1D cut from the eastern part of Sweden close to the town of Västervik. The same level map data has previously been used in [15, 16, 104] for similar simulations. The terrain in the general area is shown in Figure 7.3. The archipelagic area shows clear differences in ground level which makes it suitable for this kind of navigation.

The performance obtained with 1000 Monte Carlo simulations, with different noise and initial state realizations, over a time span of 500 samples is presented in Figure 7.4. It should be noted that the particle filter positions the aircraft very well and comes close to the CRLB, which confirms the excellent results in [16]. The plot also shows the result from the EKF, which does not perform well here. Finally, the CRLB under the assumption of Gaussian noise has been computed. The ratio

$$\frac{P_{t|t}(\mathcal{I}_w^{-1}, \mathcal{I}_e^{-1})}{P_{t|t}(\text{cov}(w), \text{cov}(e))}$$



**Figure 7.3:** The terrain used in the simulations are from the eastern part of Sweden close to the town of Västervik. The left figure shows the map, with a light line indicating the one-dimensional subset used in the simulations. The two right hand side figures show the one-dimensional altitude profile. The top one shows the whole profile while the bottom one zooms in on the start position.



**Figure 7.4:** Resulting performance from navigation example. The EKF performance is compared to the particle filter, and the posterior CRLB. Also included for comparison, the CRLB computed under a Gaussian noise assumption.

indicates the performance gain to be expected if the non-Gaussianity in the measurement noise is not taken into consideration. Two things should be noted here: first, this ratio is time varying as there generally does not exist a stationary error covariance for the EKF; second, the Kalman filter is not the BLUE for nonlinear systems and therefore  $P_{t|t}(\text{cov}(w), \text{cov}(e))$  is just an approximation of the performance to expect. The ratio has not been explicitly computed, but is around 2–3 most of the time, as can be seen in Figure 7.4. The particle filter outperforms the Gaussian approximation, and it is clearly worth the effort to model the measurement noise correctly. This has previously been found empirically for experimental data, see *e.g.*, [16].

## Conclusions

A nonlinear navigation problem with clear connections to a working navigation application has been evaluated in a simulation study. The simulation study shows, in accordance with previous empirical results, that it is, in this case, worth the extra effort to model the non-Gaussian measurement noise and take it into account when filtering. The simulation study also showed that using intrinsic accuracy and CRLB analysis could indicate this gain beforehand.

## 7.3 Range-Only Tracking

In the third simulation study a tracking application using two range-only measurements is evaluated. This can be seen as a very simplified GPS receiver. The measurements are illustrated in Figure 7.5(a). The measurements provide the relative range to an unknown target, with measurement uncertainty. Due to the complete lack of directional information the resulting posterior becomes bimodal. The model used for the system is:

$$x_{t+1} = x_t + w_t \quad (7.3a)$$

$$y_t = \begin{pmatrix} \|x_t - \mathcal{S}_1\|_2 \\ \|x_t - \mathcal{S}_2\|_2 \end{pmatrix} + e_t, \quad (7.3b)$$

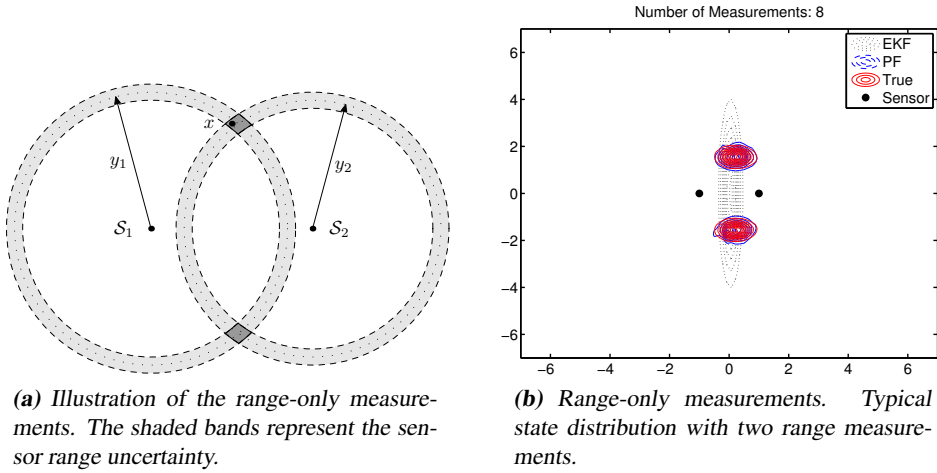
with  $w_t \sim \mathcal{N}(0, 0.1I)$ ,  $e_t \sim \mathcal{N}(0, 0.1I)$ , and initial knowledge  $x_0 \sim \mathcal{N}(0, 3I_2)$ . The sensors are located in  $\mathcal{S}_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$  and  $\mathcal{S}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ .

A typical state distribution is given in Figure 7.5(b). Note the distinctly bimodal characteristics of the distribution, as well as how poorly the EKF manages to describe the situation.

The MSE and Kullback divergence from 100 Monte Carlo simulations are given in Figure 7.6 for an EKF, a particle filter (SIR with 20 000 particles<sup>1</sup>), and a numeric approximation truth using a fine grid.

From Figure 7.5(b) the particle filter appears to estimate the posterior distribution much better than the EKF. Nonetheless, the MSE performance of the particle filter and the EKF are basically the same, with a small advantage for the particle filter. (Two poor realizations of the particle filter has a large negative impact on the MSE.) However, the Kullback divergence gives a clear indication that the PDF estimated from the particle filter

<sup>1</sup>The number of particles has been chosen excessively large to get an accurate PDF estimate. By tuning the filter better the number of particles could likely be significantly reduced.

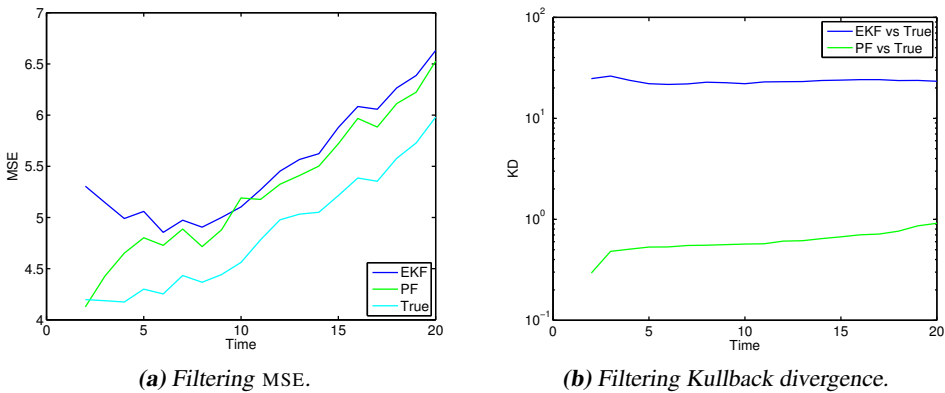


**Figure 7.5:** Scenario and typical PDF for the range-only measurement application.

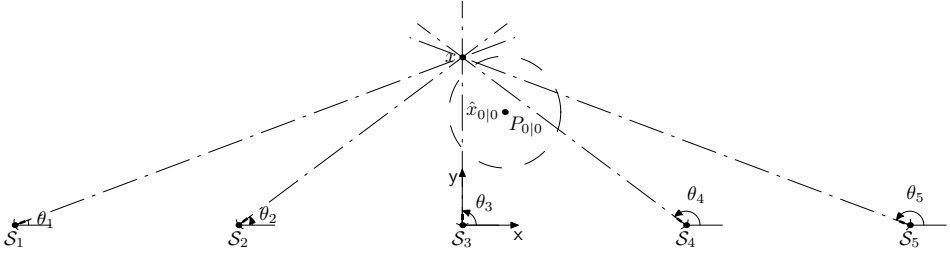
is better than the one from the EKF. Hence, this shows that the Kullback divergence is able to capture effects that the MSE cannot.

### Conclusions

A system with two range-only measurements has been studied in this section. The results are comparable for an EKF and a particle filter when it comes to MSE, however, visual inspection of the posterior state distribution suggest the particle filter estimates this distribution much better than the EKF. This observation is confirmed using the Kullback divergence. Hence, using the MSE as a filter performance measure is not enough when dealing with non-Gaussian posterior distributions.



**Figure 7.6:** Simulated MSE and Kullback divergence for the range-only system.



**Figure 7.7:** The scenario, where the true target position is denoted  $x = \begin{pmatrix} x \\ y \end{pmatrix}$ . The sensors,  $S_i$ , are the positions where the measurements are taken and  $i$  indicates the order.  $S_i = \begin{pmatrix} -30+10i \\ 0 \end{pmatrix}$ , for  $i = 1, \dots, 5$ .

## 7.4 Recursive Triangulation Using Bearings-Only Sensors

This section presents an extension of the bearings-only example that serves as an illustration throughout the thesis. The reason to use this example is that passive ranging or bearings-only tracking is a technique utilized in many applications. Many standard filtering algorithms have problems representing range or range rate uncertainty when the aim is to recursively triangulate objects using only bearing measurements. Especially for targets located close to the sensor or passing close and fast, where the sensor quadrant is changing rapidly.

Here, these problems are dealt with using the simplified setup in Figure 7.7, where the target is stationary and the measurements are taken from different positions. This does not limit the analysis to stationary target, since target movements can be achieved moving the sensors. However, the dynamics of the system is neglected to simplify the analysis. This is motivated when the uncertainty between two measurements is small compared to other errors. Measuring a stationary target with a mobile sensor with a well working positioning system is one such case.

Usually, measurement noise is, implicitly or explicitly, considered to be Gaussian. This noise description is sometimes limiting, and this section therefore analyzes the effects of non-Gaussianity for estimation using the generalized Gaussian distribution and Gaussian mixtures. Both these have the Gaussian distribution as a special case.

### 7.4.1 Sensor Model

The measurement model is a set of passive sensors measuring the direction to the target. The bearings sensors are modeled to measure the bearing to the target,  $\theta_i$ , from  $S_i$ , with additive measurement noise  $e_i$ , according to

$$y_i = h_i(x) + e_i = \theta_i + e_i = \arctan \left( \frac{y - y_i}{x - x_i} \right) + e_i, \quad (7.4)$$

where the target position is  $x = \begin{pmatrix} x \\ y \end{pmatrix}$  and  $S_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$ . The measurement noise is assumed known with the PDF  $p_e$ .

Based on the sensor model,  $n$  measurements of the target, and the initial information  $P_0^{-1}$ , the additivity of independent information yields the CRLB expression

$$\begin{aligned} P_n^{-1} &= P_0^{-1} + \sum_1^n \nabla_x h_i(x) \left( -\mathbb{E}_{e_i} \Delta_{e_i}^{e_i} \log p_e(e_i) \right) \nabla_x^T h_i(x) \\ &= P_0^{-1} + \Psi_e(\text{var}(e))^{-1} \sum_1^n \frac{1}{\bar{x}_i^2 + \bar{y}_i^2} \begin{pmatrix} \bar{y}_i^2 & -\bar{x}_i \bar{y}_i \\ -\bar{x}_i \bar{y}_i & \bar{x}_i^2 \end{pmatrix}, \end{aligned} \quad (7.5)$$

where  $(\bar{x}_i, \bar{y}_i) = (x - x_i, y - y_i)$ . Note that since the measurements are scalar the relative accuracy of the measurement noise turns up as a factor in the CRLB when the initial information is disregarded.

## 7.4.2 Simulations

The problem of a passing target object is here reformulated to coincide with a multiple sensor fusion problem. Hence, instead of a moving target the sensor is moved. The target dynamics is therefore neglected and only measurement noise affects the result.

In the Monte Carlo simulation studies, the MSE is compared to the parametric CRLB. Furthermore, the Kullback divergence, between the true state distribution and the distributions provided by the estimates, is compared to capture more of the differences not seen in the second moment. The EKF, IEKF, UKF, and particle filter are all used in the study.

### Simulation I: Gaussian Measurement Noise

First the system is analyzed for Gaussian measurements,

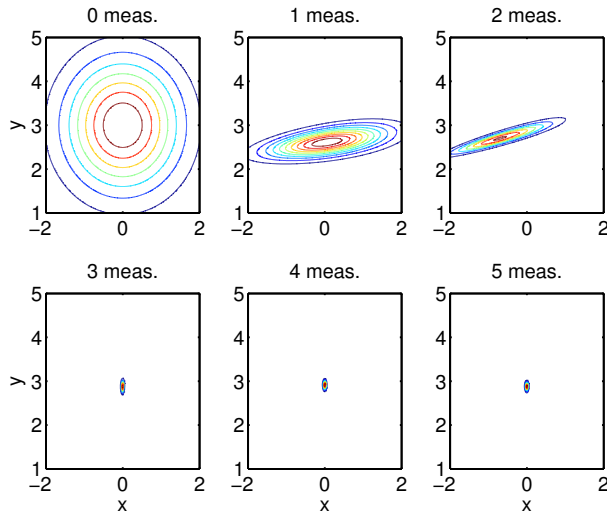
$$e_t \sim \mathcal{N}(0, \Sigma), \quad \Sigma = 10^{-4}, \quad (7.6)$$

for the measurement relation defined in (7.4), while the true target position,  $x$ , is moved along the  $y$ -axis. The five measurements, from  $S_i$ , triangulate the target. The true state distribution for one realization, as obtained from a finely gridded point-mass filter, is given in Figure 7.8.

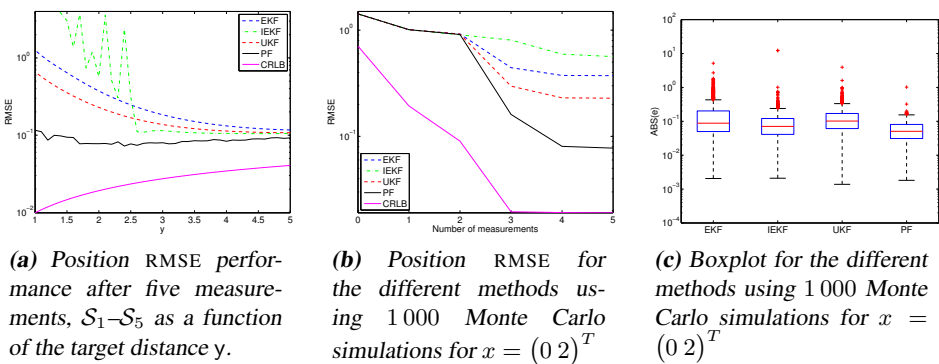
Four different estimators have been applied to this system: an EKF, an IEKF (using 5 iterations), an UKF (modified form with the recommended parameter values), and a particle filter (SIR, 10 000 particles<sup>2</sup>). The *root mean square error* (RMSE) results, from 1 000 Monte Carlo simulations, with respect to the target position,  $y$ , are presented in Figure 7.9(a). The CRLB is also given as reference.

Figure 7.9(a) shows that with the target located relatively close to the sensor ( $y \lesssim 2.5$ ), the difference in performance is substantial between the different filters. It can also be seen that the nonlinearity in the measurement relation is not as severe for targets located far away ( $y \gtrsim 3$ ) since there the difference is less obvious. The performance of the studied methods varies with the true target position, the closer to the  $x$ -axis the greater difference in performance. This is most apparent for  $\theta \approx \frac{\pi}{2}$  rad, *i.e.*, measuring from  $S_3$ .

<sup>2</sup>The number of particles has intentionally been kept high in order make sure to get maximal performance from the particle filter. However, see the comment later about how many particles to use.



**Figure 7.8:** True inferred position distribution for the scenario, utilizing one realization of the measurement noise to illustrate the reduction of uncertainty.



**(a)** Position RMSE performance after five measurements,  $S_1-S_5$  as a function of the target distance  $y$ .

**(b)** Position RMSE for the different methods using 1000 Monte Carlo simulations for  $x = (0\ 2)^T$

**(c)** Boxplot for the different methods using 1000 Monte Carlo simulations for  $x = (0\ 2)^T$

**Figure 7.9:** Simulation I, different performance measures.

In Figure 7.9(b), performance as a function of the number of measurements is evaluated for  $y = 2$ . The particle filter clearly outperforms the other methods.

To see if the degrading performance is an effect of a few outliers all distance errors are plotted in a MATLAB® box plot. The box plot yields the median, quartile values and outliers of the errors, as seen in Figure 7.9(c). See MATLAB® (help boxplot) for details about the plot. Note how the particle filter is more precise than the other methods and has fewer and less severe outliers.

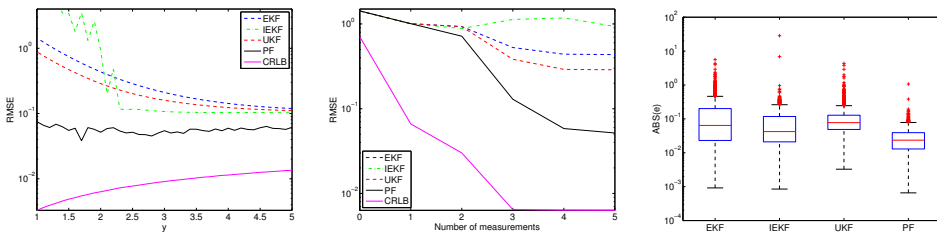
**Simulation II: Outliers**

Measurement outliers are a common problem in applications, *i.e.*, the sensor performs well on the average but in a few cases performance degrades considerably. Here, a Gaussian sum is used to describe this behavior; one mode describes nominal measurements whereas another mode models the outliers. In this section the simulations are repeated with outliers in the measurements,

$$e \sim 0.9\mathcal{N}(0, \Sigma) + 0.1\mathcal{N}(0, 100\Sigma), \tag{7.7}$$

with  $\Sigma$  such that variance  $\text{var}(e) = 10^{-4}$ , *i.e.*, the same variance as in the previous simulations. The relative accuracy for  $e$  is  $\Psi_e = 9.0$ .

The results obtained using this measurement noise are found in Figure 7.10. As seen in Figure 7.10(a), the performance of the different estimators is rather similar, as in Simulation I. However, the particle filter stands out as better, hence utilizing the non-Gaussian noise better than the other methods. As before, it is rather difficult to come close the asymptotic CRLB.

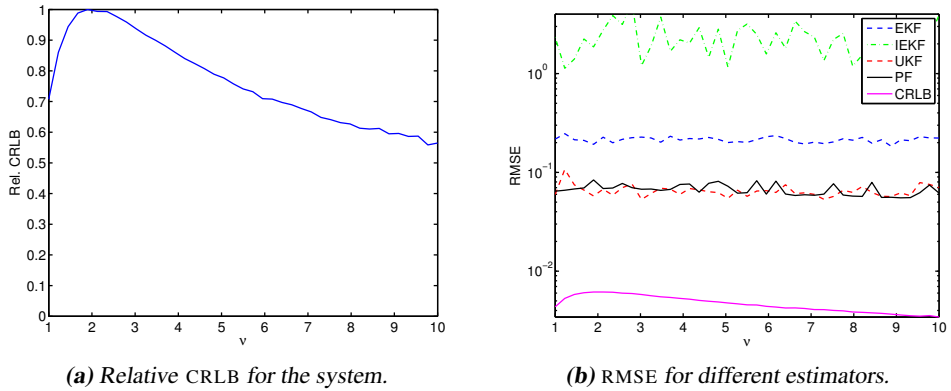


(a) Position RMSE performance after five measurements,  $S_1-S_5$  as a function of the target position  $x = (0 y)^T$ .

(b) RMSE for the different methods using 1000 Monte Carlo simulations for  $x = (0 2)^T$ .

(c) Boxplot for the different methods using 1000 Monte Carlo simulations for  $x = (0 2)^T$ .

**Figure 7.10:** Simulation II, different performance measures for measurements with outliers.



**Figure 7.11:** Simulation III, results from the simulations with generalized Gaussian measurement noise, parametrized in  $\nu$ , with 10 measurements from each sensor position.

### Simulation III: Generalized Gaussian Noise

This simulation uses generalized Gaussian noise instead of Gaussian noise to describe the measurement noise. (See Section 2.4.3 for details.) Figure 7.11(a) shows how the relative CRLB varies with the shape, given by  $\nu$ , of the distribution. The CRLB is computed for 10 independent measurements in each of the sensors according to the configuration in Figure 7.7. To see the effects of non-Gaussian noise more clearly on the CRLB, the results are normalized to be 1 for Gaussian noise ( $\nu = 2$ ). For the relative CRLB achieved this way, values less than 1 indicate, at least theoretically, a gain in performance for this scenario.

Monte Carlo simulations are used to see which performance the different estimators give. The result is presented in Figure 7.11(b) where the RMSE is plotted as a function of the generalized Gaussian parameter  $\nu$ . As seen, the particle filter and the UKF give virtually the same performance. The linearized filters, EKF and IEKF, all give much worse performance. Also note that it is hard to reach the CRLB, and the information gain indicated in Figure 7.11a is not achieved. This difficulty to reach the CRLB is most likely a result of too few measurements being available to get an almost asymptotic behavior.

### Simulation IV: Kullback Divergence

The last simulation study with the bearings-only setup compares the Kullback divergence for the different estimators. The Kullback divergences between the true state distribution and the estimated state distribution from the different filters are found in Table 7.1 for the Gaussian noise in Section 7.4.2 and in Table 7.2 outliers noise in Section 7.4.2. The results presented are from one realization (depicted in Figure 7.7) where all estimators work reasonably well. A reason for just using one realization is that the computations are expensive. The divergence has been computed numerically using a point-mass filter, where each particle has been approximated with a Gaussian with low variance compared

**Table 7.1:** Kullback divergence between true state distribution and estimated state distribution with Gaussian measurement noise.

Filter	Number of measurements					
	0	1	2	3	4	5
EKF	3.16	9.19	8.88	9.31	9.32	9.15
IEKF	3.16	9.25	9.06	11.68	11.62	11.61
UKF	3.16	9.19	8.87	9.31	9.33	9.15
PF	3.32	9.38	9.12	9.33	9.18	9.59

**Table 7.2:** Kullback divergence between true state distribution and estimated state distribution with measurement noise affected by outliers.

Filter	Number of measurements					
	0	1	2	3	4	5
EKF	3.16	10.15	10.64	11.53	10.81	11.23
IEKF	3.16	10.12	10.40	11.55	11.14	11.61
UKF	3.16	10.15	10.62	11.53	11.14	11.63
PF	3.32	9.17	8.99	8.87	9.87	9.98

to the total variance to get a smooth PDF for the particle filter. Another reason for using just one realization is that it is not obvious how to combine the results of several simulations in a fair way. The Kullback divergence compares two distributions and is not an average performance measure as is for instance the RMSE.

Without measurements, the estimated state distribution is Gaussian. The Kullback divergence should be 0, but this is not the case due to numerical errors in the integration. The error in the particle filter is slightly larger because it uses a non-parametric representation of the Gaussian, whereas the other filters represent it analytically.

Once the measurements arrive, the particle filter estimate improves in relation to the other estimates, indicating that it better captures the true nature of the state distribution. The improvement is greater for the case with outliers, indicating that the particle filter better handles the non-Gaussian noise.

### 7.4.3 Conclusions

The bearings-only problem is analyzed for a target passing close and fast to the observing sensor. Various estimators are evaluated in Monte Carlo simulation studies, and compared to fundamental theoretical estimation limits. Many of the methods based on linearization suffer severe performance losses compared to the particle filter for some of the discussed simulations. The outlier problem is discussed in detail, and the sensor modeling error is discussed in terms of the generalized Gaussian distribution. The Kullback divergence is also computed in a study and the particle filter is shown to better capture the inferred state distribution. To capture the complete state distribution instead of just the variance is important in for instance detection applications to make appropriate decisions.

An interesting questions arising from this initial Kullback divergence analysis is how many particles that are needed in a particle filter to approximate higher order moments in the particle filter well. In this study  $N = 10\,000$  particles were used, and other simulations, not shown here, indicate further improvements if  $N$  is increased. The number of particles does probably depend on the choice of proposal distribution in the filter, but how large the difference is and which posterior is the best needs further investigation.

## 7.5 DC Motor

In this section, a DC motor (direct-current motor) is used to further exemplify the Cramér-Rao lower bound theory. The DC motor is a common component in many everyday consumer products, e.g., DVD-players, hard drives in computers, and MP3-players, where the accuracy of the DC motor is important. To use a DC motor for accurate positioning it is important to be able to tell the current state of the motor, *i.e.*, how much the drive shaft is turned and how fast it is turning right now. The task in this simulation study is therefore to estimate the angle and speed of the drive shaft when either the process noise, the uncertainty in the control signal, or the measurement noise is non-Gaussian. Without loss of generality the nominal input can be removed from the system, hence motivating the lack of a regular input to the system. A DC motor can be approximated with the following linear time-invariant model with the state  $x$  consisting of the angle of the drive shaft and its angular velocity:

$$x_{t+1} = \begin{pmatrix} 1 & 1 - \alpha \\ 0 & \alpha \end{pmatrix} x_t + \begin{pmatrix} T - (1 - \alpha) \\ 1 - \alpha \end{pmatrix} w_t \quad (7.8a)$$

$$y_t = (1 \quad 0) x_t + e_t, \quad (7.8b)$$

where  $\alpha = \exp(-T)$ ,  $T = 0.4$ , and  $w_t$  and  $e_t$  are independent, but not necessarily Gaussian noises.

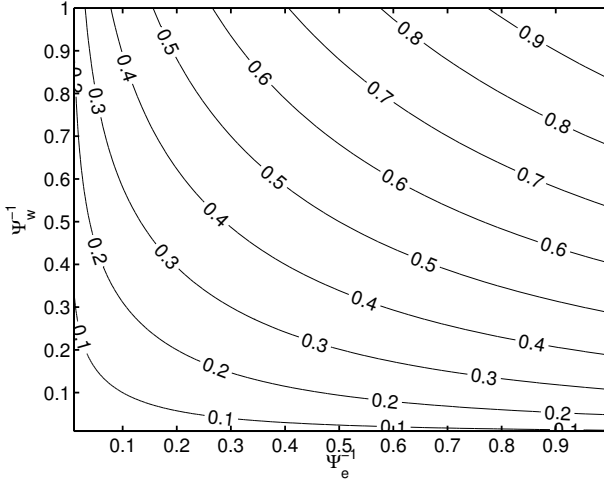
To get an understanding of how the system is affected by non-Gaussian noise study a nominal DC motor system with the noise profile

$$\begin{aligned} w_t &\sim \mathcal{N}\left(0, \left(\frac{\pi}{180}\right)^2\right) \\ e_t &\sim \mathcal{N}\left(0, \left(\frac{\pi}{180}\right)^2\right), \end{aligned}$$

*i.e.*, both process noise and measurement noise are Gaussian and have the same order of magnitude, equivalent to a standard deviation of  $1^\circ$ . This gives the asymptotic estimation performance (in this case the trace of the covariance matrix to get a scalar value)  $\text{tr}\left(\bar{\kappa}\left(\left(\frac{\pi}{180}\right)^2, \left(\frac{\pi}{180}\right)^2\right)\right) = 4.5 \cdot 10^{-4} \approx (1.2^\circ)^2$ . Figure 7.12 shows the ratio

$$\frac{\text{tr}\left(\bar{\kappa}\left(\mathcal{I}_w^{-1}, \mathcal{I}_e^{-1}\right)\right)}{\text{tr}\left(\bar{\kappa}\left(\text{var}(w), \text{var}(e)\right)\right)},$$

which indicates how much better a nonlinear filter can potentially be compared to a Kalman filter, *i.e.*, optimal performance as given by the CRLB is put in relation to the BLUE performance.



**Figure 7.12:** Contour plot of the optimal filter performance, as a function of the relative accuracies  $\Psi_{w_t}$  and  $\Psi_{e_t}$ ,  $\text{tr} \left( \bar{\kappa}(\Psi_{w_t}^{-1}, \Psi_{e_t}^{-1}) \right) / \text{tr} \left( \bar{\kappa}(\text{var}(w_t), \text{var}(e_t)) \right)$  with  $\left( \bar{\kappa}(\text{var}(w_t), \text{var}(e_t)) \right) = 1.9 \cdot 10^{-4}$ .

### 7.5.1 Measurements with Outliers

For the first simulation study of the DC motor, assume the noise configuration

$$w_t \sim \mathcal{N}\left(0, \left(\frac{\pi}{180}\right)^2\right)$$

$$e_t \sim \mathcal{N}_2\left(\left(0.9, 0, \left(\frac{\pi}{180}\right)^2\right), \left(0.1, 0, \left(\frac{10\pi}{180}\right)^2\right)\right),$$

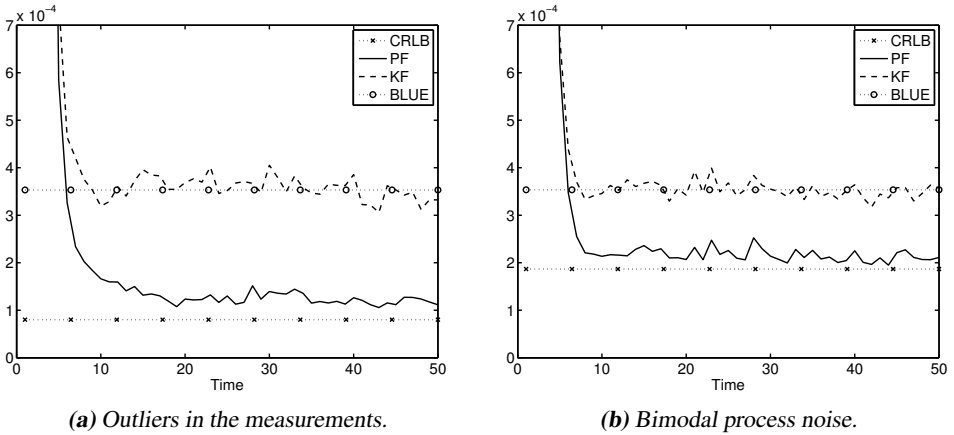
where  $\mathcal{I}_e = 1.1 \cdot 10^4$  and  $\Psi_e = 9$ . The difference between the previously described nominal DC motor is that the measurement noise now contains outliers in 10% of the measurements, and the outliers have ten times the variance of the nominal measurements. This increases the variance of the measurement noise. In a real sensor this could be the effect of temporary errors producing measurements of low quality. The measurement noise is an instance of the noise discussed in Section 2.4.2 with scaled variance.

The high intrinsic accuracy of the measurement noise, is an indication that, unless the model is such that informative measurements do not pay off, nonlinear filtering should not be ruled out without further analysis. Computing the relative filtering performance, this time the trace of the covariance matrix, for the system yields

$$\frac{\text{tr} \left( \bar{\kappa}(\mathcal{I}_w^{-1}, \mathcal{I}_e^{-1}) \right)}{\text{tr} \left( \bar{\kappa}(\text{var}(w), \text{var}(e)) \right)} = 0.23.$$

The potential performance gain is to get 23% of the variance for the BLUE, and nonlinear filtering should definitely be evaluated for this system.

Figure 7.13(a) shows the result of a Monte Carlo simulation performed on this model using a Kalman filter and a particle filter (SIR with 10 000 particles<sup>3</sup>). The result is promising because the variance of the estimate comes close to the CRLB. There is still a gap between the CRLB and the performance obtained using the particle filter but it is comparably small. The reason that the particle filter does not reach the CRLB is derived under the assumption of infinite information.



**Figure 7.13:** MSE for 1 000 MC simulations with Kalman filter and particle filter (10 000 particles) on the DC motor. CRLB and BLUE (variance) indicate asymptotic limits.

## 7.5.2 Load Disturbances

The second DC motor system studied has the noise configuration:

$$w_t \sim \mathcal{N}_2 \left( \left( 0.8, 0, \left( \frac{\pi}{180} \right)^2 \right), \left( 0.2, \frac{-10\pi}{180}, \left( \frac{\pi}{180} \right)^2 \right) \right)$$

$$e_t \sim \mathcal{N} \left( 0, \left( \frac{\pi}{180} \right)^2 \right).$$

The bimodal process noise can be interpreted as a reoccurring load disturbance which is active 20% of the time resulting in a mean of the distribution at  $-10^\circ$ , e.g., it can be that the drive shaft gets stuck slowed down by friction once in a while. Observe that this gives a process noise that is nonzero mean, and the filters must take that into consideration. The noise is a scaled instance of the bimodal noise in Section 2.4.2. The new process noise has larger variance than the nominal system yielding  $\text{tr} \left( \bar{\kappa} (\text{var}(w_t), \text{var}(e_t)) \right) c = 3.9 \cdot 10^{-3}$ . The process noise is characterized by  $\mathcal{I}_w = 3.3 \cdot 10^3$  and  $\Psi_w = 17$ , which gives

$$\frac{\text{tr} \left( \bar{\kappa} (\mathcal{I}_w^{-1}, \mathcal{I}_e^{-1}) \right)}{\text{tr} \left( \bar{\kappa}^{\text{tr}} (\text{var}(w), \text{var}(e)) \right)} = 0.53.$$

<sup>3</sup>For this system 10 000 particles seem to be enough to obtain maximum performance from the particle filter.

This indicates it should be possible to cut the MSE to approximately one half.

Monte Carlo simulations on the system yields the result in Figure 7.13(b). The result is as promising as the outliers example since the CRLB is almost reached and there is a clear improvement compared to the Kalman filter. Hence, this shows that performance can be gained with non-Gaussian process noise.

The fact that the particle filter does not reach the CRLB every time could at first come as a surprise, however the CRLB is just a lower bound without guarantees of being reachable. Lack of information in the measurements is one probable reason for this. The CRLB results are derived for asymptotic measurement information, which is not obtained for the standard filtering formulations. Also, the particle filter only obtains the true posterior distribution in the limit as  $N \rightarrow +\infty$ , which is not fulfilled either.

### 7.5.3 Conclusion

In this section a simplified model of a DC motor has been used to illustrate the theory in Chapter 6 for linear models. The simulation studies shows that it is possible to gain performance using nonlinear filter instead of the Kalman filter if the system has non-Gaussian noise. To beforehand predict if any gain is possible the ratio

$$\frac{\text{tr} \left( \bar{\kappa} \left( \mathcal{I}_w^{-1}, \mathcal{I}_e^{-1} \right) \right)}{\text{tr} \left( \bar{\kappa}^{\text{tr}} \left( \text{var}(w), \text{var}(e) \right) \right)},$$

has been computed. The ratio has been successful in predicting the performance actually possible to get from a particle filter. However, the simulations does not quite reach the CRLB which should be attributed to the fact that the CRLB is obtained under the assumption of infinite information.

## 7.6 Target Tracking

The final simulation study in this chapter treats the linear system:

$$x_{t+1} = \begin{pmatrix} 1 & T \\ 0 & 0.9 \end{pmatrix} x_t + \begin{pmatrix} T^2/2 \\ T \end{pmatrix} w_t \quad (7.9a)$$

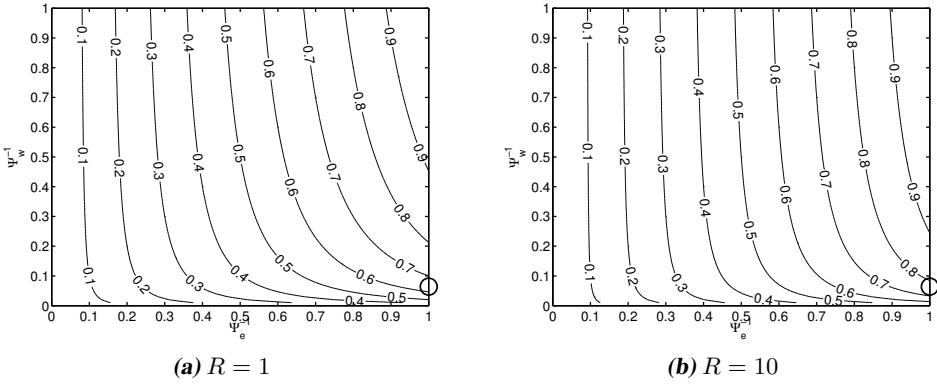
$$y_t = (1 \quad 0) x_t + e_t, \quad (7.9b)$$

where  $T = 1$ . Note that the velocity tends to zero over time, this is a deliberate choice to lessen the impact of old velocity errors. In the simulations the noise configuration is

$$w_t \sim 0.85\mathcal{N}(0, 0.065) + 0.075\mathcal{N}(-2.5, 0.065) + 0.075\mathcal{N}(+2.5, 0.065) \quad (7.9c)$$

$$e_t \sim \mathcal{N}(0, R) \quad (7.9d)$$

where  $R$  is allowed to vary to obtain different *signal-to-noise ratios* (SNR, defined as  $\|Q\|_2/\|R\|_2$ ). The relative accuracy of the process noise is high,  $\Psi_{w_t} = 15.5$ , cf. Figure 2.5(a). The system in (7.9) represents a one dimensional target tracking problem,



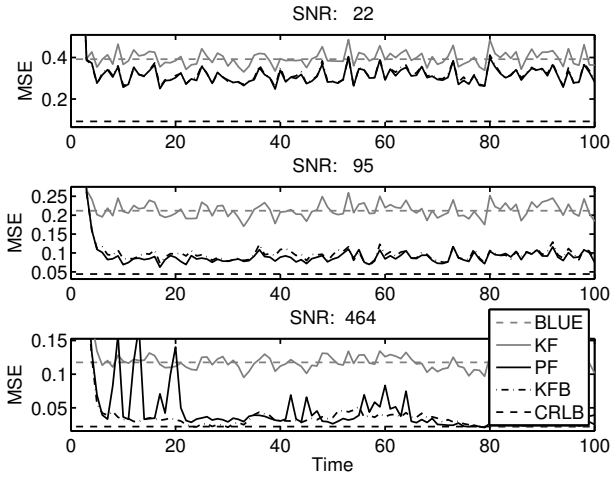
**Figure 7.14:** Relative CRLB for the system (7.9) as a function of  $\Psi_w^{-1}$  and  $\Psi_e^{-1}$ .  $\circ$  denotes the specific noise configuration.

where the multi-modal process noise includes two maneuvering modes. In Figure 7.14 the expected gain in performance for  $R = 1$  and  $R = 0.1$  (SNR 1 and 10), respectively, are shown in terms of  $\|\bar{\kappa}(\mathcal{I}_w^{-1}, \mathcal{I}_e^{-1})\|_2 / \|\bar{\kappa}(\text{cov}(w), \text{cov}(e))\|_2$ . (This time the two norm is used to get a scalar value, this to better match the definition of SNR.) The covariance matrix  $\bar{\kappa}(\text{cov}(w), \text{cov}(e))$  is the result obtained using a Kalman filter. The lower the value, the more to expect from using nonlinear estimation methods, such as the particle filter, whereas values close to 1 indicate that the BLUE is close to optimal. As can be seen in both Figure 7.14(a) and 7.14(b) there is rather large potential performance gain associated with nonlinear filtering methods in this case. To see if a particle filter pays off in this situation Monte Carlo simulations were conducted for different SNR. For each SNR a sequence of 100 time steps was simulated 500 times for  $T = 1$  and  $x_0 \sim \mathcal{N}(0, I)$ . The state was then estimated using three different filters, a standard *Kalman filter* (KF), a particle filter (using SIR and  $10^4$  particles<sup>4</sup>), and a Kalman filter bank keeping a window of 3 time steps. (See Section 5.5 details on filter banks.) The typical outcome in terms of performance varies as seen in Figure 7.15.

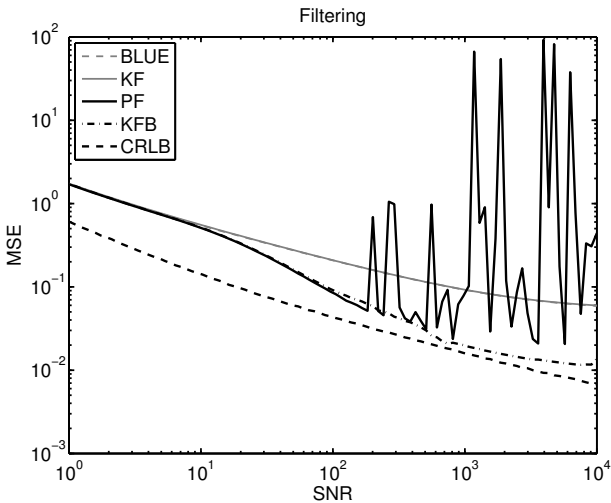
The spikes in the particle filter performance, especially for the high SNR, are due to degeneration problems when the particle cloud is very spread out compared to the uncertainty of the measurement noise. This is expected, since the proposal distribution used in SIR particle filter is not to recommend for high SNR's. The filter bank suffers less from these effects, because the filter is to a greater extent specialized to the given system, which can be reformulated as a switched model. It can however be seen that both the nonlinear filters perform better than the Kalman filter, especially when the SNR is high.

The *mean square error* (MSE) of the estimate over time,  $\text{MSE} = \frac{1}{t_f} \sum_{t=1}^{t_f} \|x_t\|_2^2$ , as a function of the SNR is plotted in Figure 7.16. It is possible to see how the nonlinear filter performs better than the BLUE for high SNR values, whereas for low SNR values the difference is only minor. The intuition to this behavior is that the gain starts to become visible once the measurements are able to distinguish between the modes in the process

<sup>4</sup>The number of particles has intentionally been chosen high to not be a performance factor in the evaluations.



**Figure 7.15:** MSE obtained from simulating the system (7.9) with different  $R$ . The spikes are due to one, or a few, diverged PF.



**Figure 7.16:** MSE for four filters as a function of SNR in the system for the target tracking example. The proposal distribution in SIR PF and limited particles degenerates performance for high SNR, while the Kalman filter bank handles all SNR's.

noise in one instance. That is, with increasing SNR the asymptotic conditions of the CRLB is better met and the performance closer to the CRLB.

## 7.7 Observations

The simulation studies in this chapter have had two major purposes: First, verify the *Cramér-Rao lower bound* (CRLB) theory in Chapter 6 and show how it can be used. Second, show that the MSE is a poor measure of how well a filter approximate the posterior state distribution.

The CRLB theory presented for linear system was used in many of the simulation studies. These simulations show that the filter performance can be improved by using a nonlinear filter on linear models, and that this performance gain is indicated by the ratio

$$\frac{\text{tr} \left( \bar{\kappa} (\mathcal{I}_w^{-1}, \mathcal{I}_e^{-1}) \right)}{\text{tr} \left( \bar{\kappa}^{\text{tr}} (\text{var}(w), \text{var}(e)) \right)},$$

being small. Hence, by computing this ratio it is possible to predict the gain from more advanced filtering techniques. At the same time the last simulation showed that this gain is not always obtained. However, the larger the SNR is, the easier it is.

Other simulations have shown that two filters yielding the same MSE when used can estimate very different posterior distributions. Here the Kullback divergence is used to indicate this difference not captured by the MSE. It is however difficult to tell what is a good and what is a bad Kullback divergence value, further studies including other norms are motivated.

# 8

---

## Change Detection

**D**ETEECTING AN UNEXPECTED change in a system as fast as possible is often vital in order to be able to adapt to new conditions. Consider the range-only tracking application introduced in Chapter 1, and assume that it is not known if there exists a target or not. A first objective is then to determine, or detect, if there is a target present or not. Another scenario is that one of the sensors breaks, and starts to deliver consistently bad and misleading measurements. If the estimate is used for collision avoidance, the result could very well be a crash. Hence, it is important to be able to detect a sensor fault in order to compensate for the effects it may have.

Trying to detect a change in a system is called *change detection* or *fault detection* since a change in a system is often considered to be a fault. The main objective in this chapter is to determine if a change has occurred. This chapter can be read independently of Chapters 5 through 7, however the material in those chapters increases the level of understanding of the detection theory.

In this chapter,  $f_t$  denotes the deterministic but unknown parameter that may change. The models used here were presented in Section 4.3, where

$$x_{t+1} = f(x_t, u_t, w_t, f_t) \quad (8.1a)$$

$$y_t = h(x_t, u_t, f_t) + e_t, \quad (8.1b)$$

is the most general model treated, and most of the time data from a whole window in time is treated at the same time principally described by a stacked model, for a linear model as (see Section 4.3)

$$\mathbb{Y}_t = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^w \mathbb{W}_t^L + \mathbb{E}_t^L + \bar{H}_t^u \mathbb{U}_t^L + \bar{H}_t^f \mathbb{F}_t^L. \quad (8.2)$$

The outline of this chapter is the following: Hypothesis testing is introduced in Section 8.1 since it is, in one way or another, the base for all fault detection. Once the basic concepts are known, general techniques for detection are given in Section 8.2, and optimality results are introduced in Section 8.3 and extended in Section 8.4. Section 8.5 derives results for linear systems. The chapter is summarized in Section 8.6.

## 8.1 Hypothesis Testing

The foundation of change detection is *hypothesis testing*. Hypothesis testing is treated in depth by many text books on change detection, e.g., [11, 77, 90, 144] just to mention a few, and is used to decide between statistical statements called *hypotheses*. Hypotheses are in this thesis denoted with  $\mathcal{H}_i$ , where  $\mathcal{H}_0$  is the *null hypothesis* which is put against one or several *alternative hypotheses*.

**Definition 8.1.** A hypothesis,  $\mathcal{H}$ , is any assumption about a distribution. The hypothesis  $\mathcal{H}$  is a *simple hypothesis* if it reduces to a single point in the parameterization of the distribution, and otherwise  $\mathcal{H}$  constitutes a *composite hypothesis*.

### Example 8.1: Hypothesis

Assume that  $y$  is a measurement of a parameter  $f$  in presence of Gaussian noise, i.e.,  $y \sim \mathcal{N}(f, 1)$ , and that either  $f = 0$  or  $f = 1$ . Deciding which value  $f$  has is to decide between two simple hypotheses:

$$\begin{cases} \mathcal{H}_0 : & \text{If } y \sim \mathcal{N}(0, 1), \text{ i.e., } f = 0 \\ \mathcal{H}_1 : & \text{If } y \sim \mathcal{N}(1, 1), \text{ i.e., } f = 1 \end{cases}.$$

Now assume instead that the question is whether  $f = 0$  or not, and that  $f$  can take on any real value. The two new hypotheses to decide between are then

$$\begin{cases} \mathcal{H}_0 : & \text{If } y \sim \mathcal{N}(0, 1), \text{ i.e., } f = 0 \\ \mathcal{H}_1 : & \text{If } y \sim \mathcal{N}(f, 1), \text{ i.e., } f \neq 0 \end{cases},$$

where the alternative hypothesis,  $\mathcal{H}_1$ , is a composite hypothesis since it corresponds to several different values of  $f$ , and the null hypothesis,  $\mathcal{H}_0$ , is simple. As will be seen further on, the second set of hypotheses is much harder to handle than the first one due to the less specific information about the fault parameter in the composite hypothesis.

The general principle for deciding between hypotheses is to derive a test statistic,  $L(\mathbb{Y})$ , and a rule to choose one hypothesis over the other. The test statistic and the rule can be constructed with many different objectives in mind. The most common objectives are [44]:

- *Probability of false alarm*,  $P_{\text{FA}}$ , which is the probability to incorrectly detect a change. The quantity  $1 - P_{\text{FA}}$  is denoted the *level* of the test.
- *Probability of detection*,  $P_{\text{D}}$ , which is the probability of correctly detecting a change when a change has occurred. This property is also known as the *power* of a test.
- *Time to detect*,  $t_{\text{D}}$ , which is the expected value of the time between the change and when it is detected.
- *Mean time between false alarms*,  $t_{\text{FA}}$ , which is the expected value of the time between two false alarms.

The probability of false alarm and the probability of detection are important properties often used together to characterize a detector. Furthermore, the two are coupled; a given probability of false alarm,  $P_{\text{FA}}$ , limits how large the probability of detection,  $P_{\text{D}}$ , can be, and a given  $P_{\text{D}}$  puts a lower bound on  $P_{\text{FA}}$ . Due to this coupling,  $P_{\text{D}}$  is often plotted against  $P_{\text{FA}}$  in what is called a *receiver operating characteristics* (ROC) diagram. The ROC diagram can then be used to decide a reasonable compromise between the probability of detection and the probability of false alarm.

With the notation adopted in this thesis, a hypothesis test involves estimating, explicitly or implicitly, the parameter  $f_t$  or  $\mathbb{F}_t^L$  in (8.1) and (8.2), respectively. The estimate is then used, often indirectly, in the test statistic to decide which hypothesis to choose. One approach, not necessarily based on stochastic theory, is to compute the least squares estimate of  $f_t$  and decide if it significantly differs from 0, or not. With better estimates of the parameter, e.g., the *best linear unbiased estimate* (BLUE), the *minimum variance estimate* (MVE), or the *maximum likelihood estimate* (MLE), it is often possible to derive tests that are more efficient. This opens up for nonlinear estimators such as the particle filter, which was shown to improve the quality of the estimate in Chapter 7. The particle filter also provides an estimate of the complete PDF of the estimated parameter, which can be utilized in the hypothesis test, see [74]. A general rule is that the more accurate the estimate of the parameter is, the better grounds for deciding between the hypotheses.

---

### Example 8.2: Hypothesis test between Gaussians

---

Let  $y \sim \mathcal{N}(f, 1)$ , where  $f = 0$  or  $f = \mu > 0$ , and design a hypothesis test for

$$\begin{cases} \mathcal{H}_0 : & \text{If } y \sim \mathcal{N}(0, 1), \text{ i.e., } f = 0 \\ \mathcal{H}_1 : & \text{If } y \sim \mathcal{N}(\mu, 1), \text{ i.e., } f = \mu \end{cases}$$

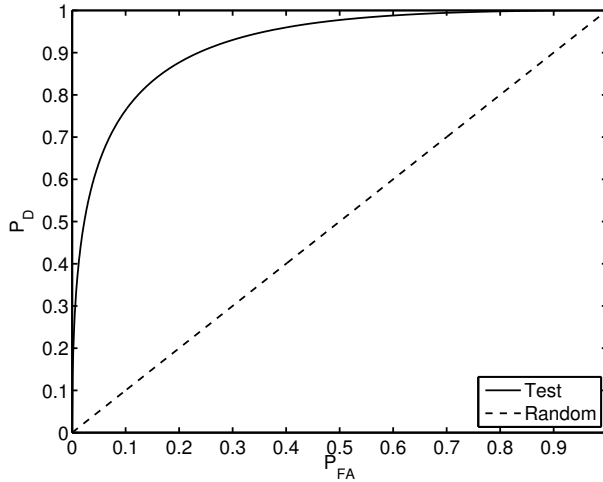
Let the test statistic be  $L(y) = y$  and decide for  $\mathcal{H}_0$  if  $y \leq \gamma$  and otherwise for  $\mathcal{H}_1$ . The probability of false alarm for the test is

$$P_{\text{FA}} = \Pr(y > \gamma | y \sim \mathcal{N}(0, 1)) = 1 - \Phi(\gamma),$$

where  $\Phi$  is the Gaussian CDF and the probability of detection is

$$P_{\text{D}} = \Pr(y > \gamma | y \sim \mathcal{N}(\mu, 1)) = 1 - \Phi(\gamma - \mu).$$

Hence, for  $\mu = 2$  and  $\gamma = 2$  the probability of false alarm is  $P_{\text{FA}} \approx 2\%$  and the probability of detection is  $P_{\text{D}} = 50\%$ . Varying  $\gamma$ , the ROC diagram in Figure 8.1 is obtained. Note the dashed line in the diagram that shows the performance obtained if a change is randomly detected with probability  $P_{\text{FA}}$ . Any test should be at least this good, or else it is better to randomly detect a change. Hence, it is always possible to obtain a test with  $P_{\text{D}} \geq P_{\text{FA}}$ .



**Figure 8.1:** ROC diagram for the detector in Example 8.2 with  $\mu = 2$ . The dashed line represents the result achieved for random detection.

## 8.2 Test Statistics

This section outlines the most commonly used test statistics: the *likelihood ratio* (LR) test, for deciding between simple hypotheses, and the *generalized likelihood ratio* (GLR) test, an extension of the regular likelihood ratio test to composite hypotheses. To conclude this section a *Bayes factor* based test is presented as the Bayesian alternative to the likelihood ratio test.

### 8.2.1 Likelihood Ratio Test

For *simple* hypothesis tests, with only one alternative hypothesis, the *likelihood ratio test* [90] is one of the most well-known tests. The likelihood ratio test uses the ratio between the probabilities of the obtained measurements under the alternative hypothesis and the probability of the null hypothesis as test statistic,

$$L(\mathbb{Y}) = \frac{p(\mathbb{Y}|\mathcal{H}_1)}{p(\mathbb{Y}|\mathcal{H}_0)}. \quad (8.3)$$

The likelier the alternative hypothesis is compared to the null hypothesis, the larger  $L(\mathbb{Y})$  becomes, and *vice versa*. Therefore, if the test statistic is larger than a threshold,  $L(\mathbb{Y}) \geq \gamma$ , the null hypothesis,  $\mathcal{H}_0$ , is rejected, and if  $L(\mathbb{Y}) < \gamma$  the null hypothesis,  $\mathcal{H}_0$ , is accepted. The following notation will be used to represent this decision rule,

$$\begin{cases} \mathcal{H}_0, & \text{if } L(\mathbb{Y}) < \gamma \\ \mathcal{H}_1, & \text{if } L(\mathbb{Y}) \geq \gamma \end{cases} \iff L(\mathbb{Y}) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\geq}} \gamma. \quad (8.4)$$

The probability of a false alarm with a given test statistic and the decision rule (8.4) is

$$P_{\text{FA}} = \Pr(L(\mathbb{Y}) \geq \gamma | \mathcal{H}_0) = \Pr(p(\mathbb{Y} | \mathcal{H}_1) \geq \gamma p(\mathbb{Y} | \mathcal{H}_0) \mid \mathcal{H}_0), \quad (8.5a)$$

and the probability of detection is

$$P_{\text{D}} = \Pr(L(\mathbb{Y}) \geq \gamma | \mathcal{H}_1) = \Pr(p(\mathbb{Y} | \mathcal{H}_1) \geq \gamma p(\mathbb{Y} | \mathcal{H}_0) \mid \mathcal{H}_1). \quad (8.5b)$$

The threshold  $\gamma$  determines how much likelier the alternative hypothesis must be before it is chosen over the null hypothesis. The larger the difference between the likelihoods, the easier it is to make a good decision. The difference can for instance be measured with the Kullback-Leibler information since it is a measure of how difficult it is to tell two distributions apart [85, 91].

— **Example 8.3: Hypothesis test between Gaussians, using likelihood ratio test** —

Assume, as in Example 8.2, a measurement  $y$  with

$$\begin{cases} \mathcal{H}_0 : y \sim \mathcal{N}(0, 1) \\ \mathcal{H}_1 : y \sim \mathcal{N}(\mu, 1) \end{cases},$$

where  $\mu > 0$ . The likelihood ratio test for deciding between  $\mathcal{H}_0$  and  $\mathcal{H}_1$  becomes

$$L(y) = \frac{p(y | \mathcal{H}_1)}{p(y | \mathcal{H}_0)} = \frac{\mathcal{N}(y; \mu, 1)}{\mathcal{N}(y; 0, 1)} = e^{\mu y - \mu^2/2} \underset{\mathcal{H}_0}{\underset{\geq}{\gamma}}.$$

To find a suitable threshold,  $\gamma$ , to achieve a level,  $1 - P_{\text{FA}}$ , for the test, solve for  $\gamma$  in

$$P_{\text{FA}} = \Pr(L(y) \geq \gamma \mid \mathcal{H}_0) = \Pr(e^{\mu y - \frac{\mu^2}{2}} \geq \gamma \mid y \sim \mathcal{N}(0, 1)).$$

It is allowed to apply a logarithm to both sides of the inequality since  $\log(x)$  is strictly increasing for  $x > 0$ . Using the logarithm yields the simplified equation

$$P_{\text{FA}} = \Pr\left(\log(L(y)) \geq \log \gamma \mid \mathcal{H}_0\right) = \Pr\left(y \geq \underbrace{\frac{\log \gamma}{\mu} + \frac{\mu}{2}}_{=:\gamma'} \mid \mathcal{H}_0\right) = 1 - \Phi(\gamma'),$$

which can be solved for  $\gamma'$  when  $y \sim \mathcal{N}(0, 1)$ . Once  $\gamma'$  is chosen, the probability of detection becomes

$$P_{\text{D}} = \Pr(y \geq \gamma' \mid \mathcal{H}_1) = 1 - \Phi(\gamma' - \mu).$$

Compare the test and the test statistic derived here with the identical result in Example 8.2. Hence, it is obvious that the test derived in Example 8.2 is indirectly based on the likelihood ratio test statistic.

To use the logarithm of the likelihood ratio, as in Example 8.3, often simplifies the involved expressions. This is for instance the case for distributions from the exponential family of distributions [117], of which the Gaussian distribution is a member.

## 8.2.2 Generalized Likelihood Ratio Test

The likelihood ratio test has one shortcoming, it requires both hypotheses to be simple to be applicable. In many situations this is overly restrictive, e.g., to determine if a parameter has its nominal value or not does in most situations result in a composite alternative hypothesis. When composite hypothesis are involved, the *generalized likelihood ratio* (GLR) test [77, 94, 95], defined as

$$L(\mathbb{Y}) = \frac{\sup_{f|\mathcal{H}_1} p(\mathbb{Y}|\mathcal{H}_1)}{\sup_{f|\mathcal{H}_0} p(\mathbb{Y}|\mathcal{H}_0)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\geq}} \gamma, \quad (8.6)$$

can be used instead. Basically, the GLR test compares the highest likelihoods obtainable under the respective hypotheses. Implicitly, this is the same as estimating the parameter using a maximum likelihood estimator under the two hypotheses, and use the estimates to construct and use a regular likelihood ratio test.

In analogy with the likelihood ratio test, the probability of false alarm and the probability of detection are given by (8.5). However, the two suprema involved often considerably complicate the task of deriving explicit expressions.

## 8.2.3 Bayes Factor Test

The *Bayes factor*, [41, 117], is the Bayesian alternative to the likelihood ratio. The major difference compared to the likelihood ratio is that it takes the prior of the hypotheses into consideration,

$$B_{01}^\pi(\mathbb{Y}) = \frac{\Pr(\mathcal{H}_1|\mathbb{Y})}{\Pr(\mathcal{H}_0|\mathbb{Y})} \bigg/ \frac{\pi(\mathcal{H}_1)}{\pi(\mathcal{H}_0)}, \quad (8.7)$$

where  $\pi$  is the prior for the hypotheses. The test based on the Bayes factor is

$$B_{01}^\pi(\mathbb{Y}) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\geq}} \gamma, \quad (8.8)$$

where the threshold  $\gamma$  should be chosen to obtain an acceptable compromise between low risk of false alarms and good detection rate. Usually  $\gamma = 1$  is a good choice, but this may differ substantially depending on the degree of precision in the approximations made to obtain the likelihoods.

When the Bayes factor is used to compare two simple hypothesis it becomes identical to the likelihood ratio.

## 8.3 Most Powerful Detector

What determines how useful a detector is depends on what it is supposed to be used for. Sometimes it is very important to have few false alarms,  $P_{FA} \ll 1$ , whereas in other situations it is very important not to miss any detections,  $P_D \gg 0$ . In yet other situations it may be important to detect faults fast, the time to detection  $t_d$  is short, and so forth. Unfortunately, most of these properties contradict each other, e.g., demanding few false alarms will inevitable make it more difficult to obtain a high detection rate. This chapter will from here on deal only with the relationship between  $P_{FA}$  and  $P_D$ , and optimality in terms of them. The following different definitions of optimality are commonly in use [11]:

- A *most powerful test* is a test that amongst all tests at a given level, equivalent to a given  $P_{FA}$ , maximizes the power,  $P_D$ , of the test.
- A *minimax test* is a test that minimizes the maximal risk of failing to detect any hypothesis.
- A *Bayes test* is a test that for a given *a priori* distribution for the hypotheses has the minimal probability of making an incorrect decision.

In this thesis the focus lies on optimality in terms of most powerfulness.

For testing between two simple hypotheses the Neyman-Pearson lemma, first introduced by Neyman and Pearson in the article series [100, 101], provides the statistics of the most powerful detector. The lemma is further elaborated on in [11, 90] and is therefore given without further explanation.

**Theorem 8.1 (Neyman-Pearson lemma)**

*Every most powerful test between two simple hypotheses for a given probability of false alarm,  $P_{FA}$ , uses, at least implicitly, the test statistic*

$$L(\mathbb{Y}) = \frac{p(\mathbb{Y}|\mathcal{H}_1)}{p(\mathbb{Y}|\mathcal{H}_0)},$$

*and the decision rule*

$$L(\mathbb{Y}) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\geq}} \gamma,$$

*where the threshold  $\gamma$  is obtained by solving for  $\gamma$  in*

$$P_{FA} = \Pr(L(\mathbb{Y}) \geq \gamma \mid \mathcal{H}_0).$$

**Proof:** See proof of Theorem 1 in Chapter 3 of [90]. □

The Neyman-Pearson lemma (Theorem 8.1) does not only give a limit on how powerful tests that can be constructed; it also states that the likelihood ratio test is the most powerful test for simple hypothesis, and that every most powerful test will somehow be based on this test statistic.

For hypothesis tests including at least one composite hypothesis the situation is more complicated. One way to get an upper performance bound is to assume the true fault parameter to be known, and construct a simple hypothesis test based on this information. The test obtained under these conditions is called a *clairvoyant* test. If the simple clairvoyant test is optimal, its performance constitutes an upper bound for the performance of any detector working on the original problem.

The concept of most powerful tests is not general enough to handle composite tests because there may be different most powerful tests depending on the actual parameter value. The *uniformly most powerful* (UMP) property is therefore introduced to solve this problem. A test is UMP if it is most powerful for every possible parameter value under the alternative hypothesis.

**Definition 8.2.** A hypothesis test that is most powerful for all  $f$  under  $\mathcal{H}_1$  is said to be a *uniformly most powerful* (UMP) test.

The concept of UMP tests is the natural extension of most powerfulness to handle composite tests. The price paid for introducing the extra degree of freedom in the composite hypothesis is that it is much harder to find UMP tests than to find most powerful tests. Hence, there is no generalized Neyman-Pearson lemma for composite tests. In [90] examples are given both of cases when UMP tests exist and of cases when it is possible to show that no UMP test exists. For instance, there is no UMP test between  $\mathcal{H}_0 : f = f^0$  and  $\mathcal{H}_1 : f \neq f^0$  unless  $f$  is further restricted, or infinite information is available [90]. The latter case will be studied next.

## 8.4 Asymptotic Generalized Likelihood Ratio Test

Theorem 8.1 shows that the likelihood ratio test is optimal for simple hypotheses. Unfortunately, the strong optimality properties of the likelihood ratio test do not carry over to composite hypothesis tests and the GLR test described in Section 8.2.2. In fact, the exact optimality properties of the GLR test are unknown. However, the GLR test is known to be optimal in several special cases [11]. One case when optimality can be shown is when infinite information is available. Therefore, the sequel of this chapter will be used to study the asymptotic properties of the GLR test using data from a window in time.

### Theorem 8.2 (Asymptotic GLR test statistics)

*The generalized likelihood ratio test is asymptotically uniformly most powerful amongst all tests that are invariant (invariance imposes no restriction for the usual cases, see [90] for details), when the value of the fault parameter  $f$  is the only differing between the null hypothesis and alternative hypothesis. Furthermore, the asymptotic test statistic is given by*

$$L'(\mathbb{Y}) := 2 \log L(\mathbb{Y}) \stackrel{a}{\sim} \begin{cases} \chi_{n_f}^2, & \text{under } \mathcal{H}_0 \\ \chi_{n_f}^{\prime 2}(\lambda), & \text{under } \mathcal{H}_1 \end{cases},$$

where

$$L(\mathbb{Y}) = \frac{\sup_{f|\mathcal{H}_1} p(\mathbb{Y}|\mathcal{H}_1)}{\sup_{f|\mathcal{H}_0} p(\mathbb{Y}|\mathcal{H}_0)}$$

is the generalized likelihood ratio test statistic,  $\chi_n^{\prime 2}(\lambda)$  is the non-central  $\chi^2$  distribution of order  $n$  with non-centrality parameter  $\lambda$ , and

$$\lambda = (f^1 - f^0)^T \mathcal{I}(f = f^0)(f^1 - f^0).$$

Here  $\mathcal{I}(f = f^0)$  is the Fisher information of the stacked faults, at the true value  $f^0$ , under  $\mathcal{H}_0$ , and  $f^1$  is the true value of  $f$  under  $\mathcal{H}_1$ .

**Proof:** See [77, Ch. 6]. □

For anything but theoretical argumentation, the assumption of infinite information is unrealistic. However, the asymptotic behavior constitutes a fundamental upper performance limitation and as such it can be compared to the Cramér-Rao lower bound for estimation (see Chapter 6). Furthermore, in practice the GLR test usually performs quite well for moderately sized series of measurements [77]. Hence, the asymptotic behavior indicates what performance to expect.

### 8.4.1 Wald Test

One way to motivate the asymptotic GLR test statistics is to optimally estimate the fault parameter,  $f$ , and based on this estimate, with known statistics, do hypothesis testing. This approach is called the *Wald test* if the parameter is estimated with a maximum likelihood estimate, and it is known that it has the same favorable asymptotic properties as the GLR test [77, 145].

To use the Wald test, the first step is to obtain an estimate,  $\hat{f}$ , of the fault parameter. From Chapter 6, the optimal estimate is known to be asymptotically distributed according to

$$\hat{f} \stackrel{a}{\sim} \mathcal{N}(f, \mathcal{I}_f^{-1}),$$

assuming, without loss of generality, that  $f^0 = 0$ . Normalizing the estimate to obtain an estimate with unit covariance matrix yields

$$\hat{\tilde{f}} := \mathcal{I}_f^{\frac{1}{2}} \hat{f} \stackrel{a}{\sim} \mathcal{N}(\mathcal{I}_f^{\frac{1}{2}} f, I).$$

Under  $\mathcal{H}_0$ , where  $f = 0$ , the estimate is then distributed according to

$$\hat{\tilde{f}} \stackrel{a}{\sim} \mathcal{N}(\mathcal{I}_f^{\frac{1}{2}} f^0, I) = \mathcal{N}(0, I).$$

With  $L(\mathbb{Y}) = \|\hat{\tilde{f}}\|_2^2$  as test statistic, the distribution of  $L(\mathbb{Y})$  under  $\mathcal{H}_0$  becomes

$$L(\mathbb{Y}) := \|\hat{\tilde{f}}\|_2^2 \stackrel{a}{\sim} \chi_{n_f}^2.$$

Under the alternative hypothesis,  $\mathcal{H}_1$ , where  $f = f^1$ ,

$$\hat{\tilde{f}} \stackrel{a}{\sim} \mathcal{N}(\mathcal{I}_f^{\frac{1}{2}} f^1, I)$$

yielding the test statistic

$$L(\mathbb{Y}) = \hat{f}^{1T} \mathcal{I}_f \hat{f}^1 \stackrel{a}{\sim} \chi_{n_f}^2(\lambda),$$

where  $f^{1T}$  is a shorthand notation for  $(f^1)^T$  and  $\lambda = f^{1T} \mathcal{I}_f f^1$ . With this information a suitable threshold can be found for the decision rule

$$L(\mathbb{Y}) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\geq}} \gamma'.$$

The test statistics derived this way should be recognized as the same as for the asymptotic GLR test in Theorem 8.2. However, the test statistics are only valid under the asymptotic assumption, and the behavior of the Wald test and the GLR test may differ for finite information.

The derivation of the Wald test indicates how an approximation of the system noise affects the performance of a test. With a Gaussian approximation of the noise, the variance of the estimate of the fault parameter increases (see Chapter 6), which makes it more difficult to detect a fault.

### 8.4.2 Detection Performance

For the asymptotic GLR test, and consequently the asymptotic Wald test, with the threshold  $\gamma'$  the probability of a false alarm,  $P_{\text{FA}}$ , is

$$P_{\text{FA}} = \mathcal{Q}_{\chi_{n_f}^2}(\gamma'), \quad (8.9)$$

where  $\mathcal{Q}_\star$  denotes the complementary cumulative distribution function of the distribution  $\star$ . This follows directly from the test statistics given by Theorem 8.2. Note that  $P_{\text{FA}}$  depends only on the choice of threshold,  $\gamma'$ , and not on  $f^0$  or  $f^1$ .

The probability of detection is

$$P_{\text{D}} = \mathcal{Q}_{\chi_{n_f}^2(\lambda)}(\gamma'), \quad (8.10)$$

where  $\lambda$  is defined in Theorem 8.2. The function  $\mathcal{Q}_{\chi_{n_f}^2(\lambda)}(\gamma')$  is monotonously increasing in  $\lambda$  (increasing the mean decreases the risk that a detection is missed) thus any increase in  $\lambda$  will improve  $P_{\text{D}}$ . Note, it follows immediately that if the magnitude of  $f^1$  increases it is easier to detect the change, and that if  $\mathcal{I}_f$  increases  $P_{\text{D}}$  increases, *i.e.*, any non-Gaussian noise component will increase the probability of detection with preserved  $P_{\text{FA}}$ , unless the system suppresses the informative noise direction.

---

#### Example 8.4: Asymptotic GLR statistics

---

Consider measurements from

$$y_t = f + e_t, \quad t = 1, \dots, L, \quad (8.11)$$

where  $e_t$  is any noise with  $\text{var}(e_t) = 1$  and relative accuracy  $\Psi_e$ . It then follows that the normalized estimate of  $f$  is

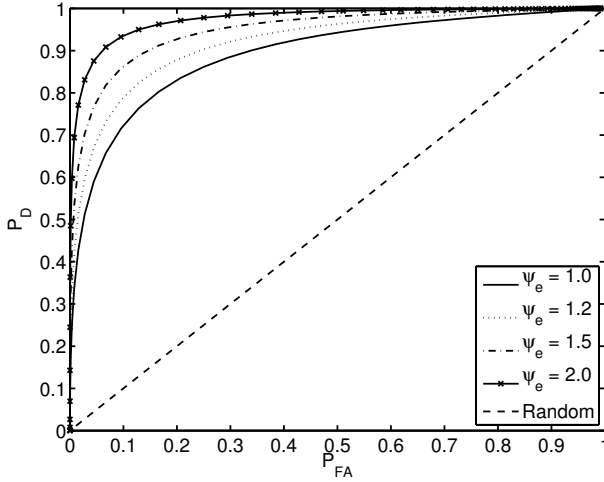
$$\hat{f} \stackrel{a}{\sim} \mathcal{N}\left(\sqrt{\Psi_e L} f, 1\right),$$

and subsequently to  $\lambda = \Psi_e L (f^1)^2 = \Psi_e L$  in the GLR statistics under the assumption  $f^1 = 1$ . The improved detection performance is illustrated with a ROC diagram in Figure 8.2. From the ROC diagram it is clear that there is potentially much to be gained from utilizing information about non-Gaussian noise in this simple model, especially for small  $P_{\text{FA}}$  where the relative increase in  $P_{\text{D}}$  is substantial.

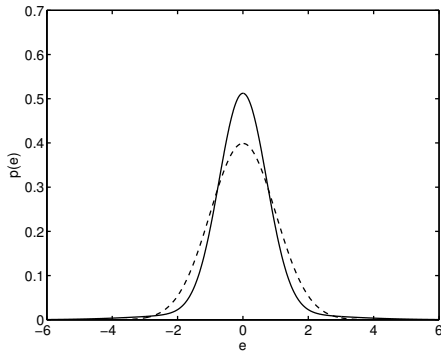
To show the performance actually obtainable, assume that the measurement noise is subject to outliers,

$$e_t \sim 0.9\mathcal{N}(0, \Sigma) + 0.1\mathcal{N}(0, 10\Sigma), \quad (8.12)$$

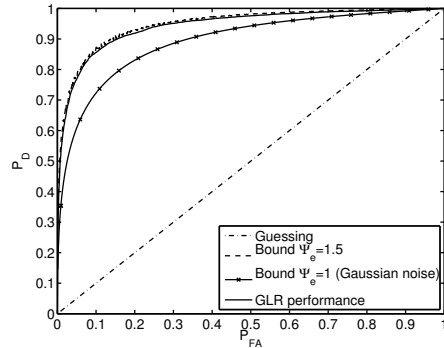
where  $\Sigma = 0.28$ , so that  $\text{var}(e_t) = 1$ . This is an instance of the outlier noise described in Section 2.4.2 with  $\mathcal{I}_e = \Psi_e = 1.5$ . The PDF is shown in Figure 8.3(a).



**Figure 8.2:** ROC diagram for (8.11) with  $n_f = 1$ ,  $L = 5$  measurements, and  $f^1 = 1$  for different values of  $\Psi_e$ . Random denotes what happens if all available information is discarded and a change is signaled randomly with probability  $P_{FA}$ .



**(a)** Solid line, PDF for measurement noise with outliers, (8.12),  $\text{var}(e_t) = 1$ , and  $\Psi_e = 1.5$ . Dashed line shows the Gaussian approximation with the same mean and variance.



**(b)** ROC diagram for (8.11) with measurements (8.12) ( $\Psi_e = 1.5$ ) for 10 000 MC simulations. Optimal performance for  $\Psi_e = 1$  and  $\Psi_e = 1.5$  are found as reference.

**Figure 8.3:** Illustration of GLR test. (a), PDF of a measurement noise. In (b) matching ROC diagram.

Simulations with this setup, using 10 000 Monte Carlo simulations, a complete Kalman filter bank, and a numeric GLR test implementation, yields the ROC diagram in Figure 8.3(b). The result is promising since the simulations seem to come close to the performance bound (cf. Figure 8.2).

## 8.5 Uniformly Most Powerful Test for Linear Systems

In this section, the theory for general systems, presented in the previous sections, is applied to two problems with linear *residuals*. General results are derived and it is then shown how the theory can be applied to two special cases of change detection with linear systems.

### 8.5.1 Linear System Residuals

Most change detection methods do not deal directly with the measurements. The difference between the measurements and the measurements expected under the null hypothesis,

$$r_t = y_t - h(x_t, f_t^0), \quad (8.13)$$

is studied instead. The difference  $r_t$  is called a *residual*. Without noise the residual should be zero under  $\mathcal{H}_0$ , but since noise is always present the residual should be small and average to zero over time if no fault is present. The exact residual distribution is determined by the system and the noise affecting it. For linear models with the fault structure discussed in Section 4.3 the residual is given by

$$r_t = y_t - H_t x_t = H_t^w w_t + e_t + H_t^f f_t = \underbrace{H_t^f \varphi_t^T}_{=: H_t^\theta} \theta + H_t^w w_t + e_t,$$

where  $\theta$  is the fault parameter and  $\varphi_t$  make up a basis to describe incipient faults. If  $L$  measurements in a time window are gathered, the stacked model becomes

$$\begin{aligned} \mathbb{R}_t^L &= \mathbb{Y}_t^L - \mathcal{O}_t x_{t-L+1} = \bar{H}_t^f \mathbb{F}_t^L + \bar{H}_t^w \mathbb{W}_t^L + \mathbb{E}_t^L \\ &= \bar{H}_t^\theta \theta + \underbrace{(\bar{H}_t^w \quad I)}_{=: \bar{H}_t^v} \underbrace{\begin{pmatrix} \mathbb{W}_t^L \\ \mathbb{E}_t^L \end{pmatrix}}_{=: \mathbb{V}_t^L} = \bar{H}_t^\theta \theta + \bar{H}_t^v \mathbb{V}_t^L, \end{aligned} \quad (8.14)$$

which is a linear regression in the fault parameter  $\theta$ , and where  $\mathbb{V}_t^L$  combines the noise effects over the window. Note that the dimension of  $\theta$  does not increase due to the stacking. This is significant because a consequence of this is that if the studied window is increased the information available about the elements in  $\theta$  increases.

Using the residual (8.14), the statistics of the asymptotic GLR test, *i.e.*,

$$L'(\mathbb{Y}_t^L) = 2 \log \left( \frac{\sup_{\mathbb{F}_t^L | \mathcal{H}_1} p(\mathbb{R}_t^L | \mathcal{H}_1)}{\sup_{\mathbb{F}_t^L | \mathcal{H}_0} p(\mathbb{R}_t^L | \mathcal{H}_0)} \right) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\geq}} \gamma', \quad (8.15)$$

becomes

$$P_{\text{FA}} = \mathcal{Q}_{\chi_{n_\theta}^2}(\gamma') \quad (8.16a)$$

$$P_{\text{D}} = \mathcal{Q}_{\chi_{n_\theta}^2}(\lambda)(\gamma'), \quad (8.16b)$$

where  $\gamma'$  is the detection threshold and

$$\begin{aligned}\lambda &= \theta^{1T} \bar{H}_t^{\theta T} \mathcal{I}_{\bar{H}_t^v \mathbb{W}_t^L} \bar{H}_t^\theta \theta^1 = \theta^{1T} \bar{H}_t^{\theta T} (\bar{H}_t^v \mathcal{I}_{\mathbb{V}_t^L}^{-1} \bar{H}_t^{vT})^{-1} \bar{H}_t^\theta \theta^1 \\ &= \theta^{1T} \bar{H}_t^{\theta T} (\bar{H}_t^w \mathcal{I}_{\mathbb{W}_t^L}^{-1} \bar{H}_t^{wT} + \mathcal{I}_{\mathbb{E}_t^L}^{-1})^{-1} \bar{H}_t^\theta \theta^1,\end{aligned}$$

where the last simplification follows if  $\mathbb{W}_t^L$  and  $\mathbb{E}_t^L$  are independent. From this it is clear that if the intrinsic accuracy of  $\mathbb{V}_t^L$  is improved, *i.e.*, by modeling non-Gaussian components in the process or measurement noise, the probability of detection increases. If the improvement is in a direction ignored by the system, the potential gain is lost.

It is sometimes useful to use only part of the residual derived above. This can be achieved by applying a projection when constructing  $\mathbb{R}_t^L$ , this way removing a subspace of the residual space known to be difficult to estimate or otherwise unsuitable for usage in the test. The parity space formulation, discussed below as a special case of this theory, uses a carefully chosen projection to remove the influence of the unknown initial state. Introducing a projection only changes the matrices  $\bar{H}_t^\theta$  and  $\bar{H}_t^v$ , hence the analysis done above holds with minimal adjustments.

### 8.5.2 Prior Initial State Knowledge

One of the unknown components of the residual description (8.14) is the initial state of the studied window,  $x_{t-L+1}$ . If an estimate of the state is available,  $\hat{x}_{t-L+1}$ , from measurements outside the window it can be used. However, an estimate is just an approximation of the true state and an extra noise term is introduced in the residual description. The residual becomes,

$$\begin{aligned}\mathbb{R}_t^L &= \mathbb{Y}_t^L - \mathcal{O}_t \hat{x}_{t-L+1} = \mathcal{O}_t \tilde{x}_{t-L+1} + \bar{H}_t^w \mathbb{W}_t^L + \mathbb{E}_t^L + \bar{H}_t^f \mathbb{F}_t^L \\ &= \bar{H}_t^\theta \theta + (\mathcal{O}_t \quad \bar{H}_t^w \quad I) \begin{pmatrix} \tilde{x}_{t-L+1} \\ \mathbb{W}_t^L \\ \mathbb{E}_t^L \end{pmatrix}, \quad (8.17)\end{aligned}$$

where  $\tilde{x}_{t-L+1} := x_{t-L+1} - \hat{x}_{t-L+1}$ . The distribution of  $\tilde{x}_{t-L+1}$  depends on the estimator used to obtain  $\hat{x}_{t-L+1}$ . Any estimate of  $x_{t-L+1}$  will do, but with a better estimate the detection performance improves. Assuming that  $\hat{x}_{t-L+1}$  is not based on data in the studied window avoids dependencies between the noise terms, which may be difficult to handle. The residual formed this way still fits into the general linear regression framework given by (8.14), hence the analysis in Section 8.5.1 still applies and the test statistics are:

$$P_{\text{FA}} = \mathcal{Q}_{\chi_{n_\theta}^2}(\gamma) \quad (8.18a)$$

$$P_{\text{D}} = \mathcal{Q}_{\chi_{n_\theta}^2}(\lambda)(\gamma), \quad (8.18b)$$

where

$$\begin{aligned}\lambda &= \theta^{1T} \bar{H}_t^{\theta T} \left( (\mathcal{O}_t \quad \bar{H}_t^w \quad I) \mathcal{I}_{\mathbb{V}_t^L}^{-1} (\mathcal{O}_t \quad \bar{H}_t^w \quad I)^T \right)^{-1} \bar{H}_t^\theta \theta^1 \\ &= \theta^{1T} \bar{H}_t^{\theta T} (\mathcal{O}_t \mathcal{I}_{\tilde{x}_{t-L+1}}^{-1} \mathcal{O}_t^T + \bar{H}_t^w \mathcal{I}_{\mathbb{W}_t^L}^{-1} \bar{H}_t^{wT} + \mathcal{I}_{\mathbb{E}_t^L}^{-1})^{-1} \bar{H}_t^\theta \theta^1\end{aligned}$$

with  $\mathbb{V}_t^L = \begin{pmatrix} \tilde{x}_{t-L+1} \\ \mathbb{W}_t^L \\ \mathbb{E}_t^L \end{pmatrix}$ , and the last step follows if  $\tilde{x}_{t-L+1}$ ,  $\mathbb{W}_t^L$ , and  $\mathbb{E}_t^L$  are independent.

### 8.5.3 Parity Space

Sometimes it is favorable to look only at contributions to the residual that could not come from the nominal system. To study only this part of the residual is called to use *parity space* or *analytic redundancy* [11, 39]. One reason to work in parity space is that no estimate of the initial state is available, or that the available estimate is poor and unreliable. The residual are obtained in parity space as

$$\begin{aligned} \mathbb{R}_t^L &= \mathcal{P}_O^\perp (\mathbb{Y}_t^L - \mathcal{O}_t x_{t-L+1} - \bar{H}_t^u \mathbb{U}_t^L) = \mathcal{P}_O^\perp \mathbb{Y}_t^L - \mathcal{P}_O^\perp \bar{H}_t^u \mathbb{U}_t^L \\ &= \mathcal{P}_O^\perp (\bar{H}_t^w \mathbb{W}_t^L + \mathbb{E}_t^L + \bar{H}_t^f \mathbb{F}_t^L) = \mathcal{P}_O^\perp \bar{H}_t^\theta \theta + \mathcal{P}_O^\perp (\bar{H}_t^w \quad I) \begin{pmatrix} \mathbb{W}_t^L \\ \mathbb{E}_t^L \end{pmatrix}, \end{aligned} \quad (8.19)$$

where  $\mathcal{P}_O^\perp$  is an orthogonal projection matrix such that  $\mathcal{P}_O^\perp \mathcal{O} = 0$ , i.e., a projection into the complement of the state space. This way any contribution from  $x_{t-L+1}$  is effectively removed from the residuals, and hence it is unimportant that  $x_{t-L+1}$  is unknown. Other elements can be removed as well, e.g., unknown input, however it is important to make sure that not the whole signal space is removed

The projection  $\mathcal{P}_O^\perp$  can be constructed using a *singular value decomposition* (SVD) [40] of the extended observability matrix  $\mathcal{O}$ . First, take the SVD of  $\mathcal{O}$ ,

$$\mathcal{O} = U \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} V^T = (U_1 \quad U_2) \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} (V_1 \quad V_2)^T, \quad (8.20)$$

where  $U$  and  $V$  are unitary matrices, split into blocks so that  $U_1$  spans the range of  $\mathcal{O}$  and  $V_2$  the null space of  $\mathcal{O}$ , and  $\Sigma$  is a diagonal matrix with the non-zero singular values of  $\mathcal{O}$ . After that, choose  $\mathcal{P}_O^\perp = U_2^T$  to get a projection with the desired properties. Note,

$$\text{cov}(\mathbb{R}_t^L) = \mathcal{P}_O^\perp \bar{H}_t^w \Sigma_{\mathbb{W}_t^L} \bar{H}_t^{wT} \mathcal{P}_O^{\perp T} + \mathcal{P}_O^\perp \Sigma_{\mathbb{E}_t^L} \mathcal{P}_O^{\perp T}, \quad (8.21)$$

and  $\mathcal{P}_O^\perp$  has full row rank and therefore  $\text{cov}(\mathbb{R}_t^L) \succ 0$  if  $\text{cov}(\mathbb{E}_t^L) \succ 0$ . Furthermore, in order for the parity space to contain any interesting information the system must be such that  $\text{rank}(\mathcal{P}_O^\perp \bar{H}_t^f) > 0$  in order for faults to show up in the residuals.

In the parity space setting the signal part of the model is removed using a projection leaving only  $\mathcal{I}_{\mathbb{W}_t^L}$  and  $\mathcal{I}_{\mathbb{E}_t^L}$  as interesting quantities. The statistics are

$$P_{\text{FA}} = \mathcal{Q}_{\chi_{n_\theta}^2}(\gamma') \quad (8.22a)$$

$$P_{\text{D}} = \mathcal{Q}_{\chi_{n_\theta}^{\prime 2}}(\lambda)(\gamma'), \quad (8.22b)$$

where

$$\begin{aligned} \lambda &= \theta^{1T} \bar{H}_t^{\theta T} \mathcal{P}_O^{\perp T} \left( (\bar{H}_t^w \quad I) \mathcal{I}_{\mathbb{W}_t^L}^{-1} (\bar{H}_t^w \quad I)^T \right)^{-1} \mathcal{P}_O^\perp \bar{H}_t^\theta \theta^1 \\ &= \theta^{1T} \bar{H}_t^{\theta T} \mathcal{P}_O^{\perp T} (\bar{H}_t^w \quad I) \mathcal{I}_{\mathbb{W}_t^L}^{-1} (\bar{H}_t^w \quad I)^T + \mathcal{I}_{\mathbb{E}_t^L}^{-1} \mathcal{P}_O^\perp \bar{H}_t^\theta \theta^1 \end{aligned}$$

with  $\mathbb{V}_t^L = \begin{pmatrix} \mathbb{W}_t^L \\ \mathbb{E}_t^L \end{pmatrix}$  and where the last step follows if  $\mathbb{W}_t^L$  and  $\mathbb{E}_t^L$  are independent.

## 8.6 Summary

This chapter has studied asymptotic properties of change detection focusing on the asymptotic UMP property of the generalized likelihood ratio test. An important factor for the potential detection performance is the non-centrality parameter in the non-central  $\chi^2$  distribution of  $\lambda$  in Theorem 8.2. The parameter essentially describes the *fault-to-noise ratio* (FNR). The noise in this case is a combination of process and measurement noise affecting the ability to estimate the fault parameter. The probability of detection,  $P_D$ , given by the asymptotic GLR test, derived under an asymptotic information assumption, is then a strictly increasing but nonlinear function of  $\lambda$ .



# 9

---

## Detection Performance

**T**HIS CHAPTER CONTAINS simulation studies of detection algorithms with the purpose to illustrate the contents in Chapter 8. It also shows how the asymptotic *generalized likelihood ratio* (GLR) test statistics can be used as an indication of if it is worth the extra effort to model and handle non-Gaussian effects in the detection setting.

The chapter is organized in the following way: In Section 9.1 a constant velocity model is evaluated with respect to non-Gaussian measurement and process noise. In Section 9.2 the DC motor from Chapter 7 is evaluated with respect to an incipient fault and different noise characteristics. Section 9.3 contains an example of how important the posterior distribution can be when computing the probability of a hypothesis. The simulation results in this chapter are summarized in Section 9.4.

### 9.1 Constant Velocity Model

The first simulation uses the constant velocity model (*cf.* Example 4.1),

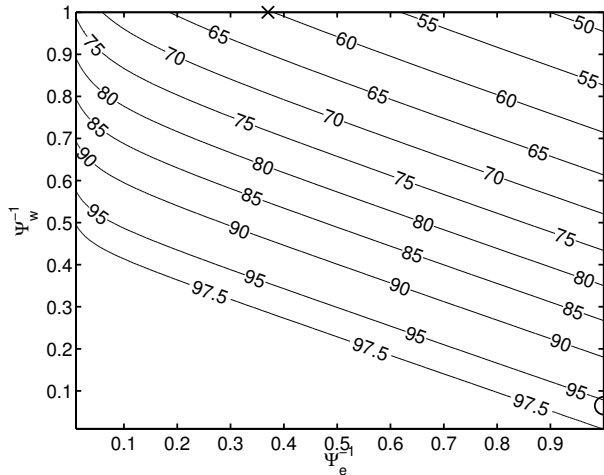
$$x_{t+1} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} \frac{1}{2}T^2 \\ T \end{pmatrix} w_t + \begin{pmatrix} \frac{1}{2}T^2 \\ T \end{pmatrix} f_t \quad (9.1a)$$

$$y_t = (1 \ 0) x_t + e_t, \quad (9.1b)$$

with  $T = 1$  and  $f_t$  as the fault term. The process noise and measurement noise are known to be mutually independent white noise with  $\text{var}(w_t) = \text{var}(e_t) = 1$ . The state is constructed as

$$x = \begin{pmatrix} x \\ v \end{pmatrix},$$

where  $x$  and  $v$  represent position and velocity, respectively. This is however just one interpretation of the state-space model, it can also be interpreted as a second order random



**Figure 9.1:** Probability of detection (in %) for the given constant velocity model (9.1) at  $P_{\text{FA}} = 1\%$  for detection in parity space and a window size  $L = 6$ . For Gaussian noise ( $\Psi_w = \Psi_w = 1$ ) the probability of detection is 48%. (The  $\circ$  indicates the bi-Gaussian noise used in the simulations, and the  $\times$  the tri-Gaussian process noise used.)

walk or a double integrator, and it can then be used to model a slowly changing state, e.g., a fluid level in a container.

Two simulation studies are based on this setup. The first has non-Gaussian measurement noise and the second trimodal process noise. To lower the complexity of the detection problem the fault is assumed to be either present or not, i.e.,  $f_t = \theta$  without any incipient behavior. Throughout both the simulations the true fault profile is

$$f_t = \begin{cases} 0, & t \leq 25 \\ 1.5, & t > 25, \end{cases}$$

i.e., the fault appears suddenly after time  $t = 25$ . In terms of the constant velocity model this can be interpreted as an unexpected maneuver or that the measurement platform has started to move.

Detection is performed on a window of  $L = 6$  samples, for which a batched model has been formulated according to Chapter 4. To separate the detection problem from the estimation problem, which is thoroughly treated in Chapters 5–7, detection will be done in parity space. The parity space effectively excludes the effect of the initial state in each window, hence the initial state does not have to be estimated. How to construct the parity space for batched linear models is described in Chapter 8.

The optimal detection performance in terms of the asymptotic GLR performance for the constant velocity model (9.1) can be computed from the test statistics in (8.22), given  $L = 6$ ,  $f_t^1 = 1.5$ , and the noise characteristics. Figure 9.1 shows the probability of detection, if  $P_{\text{FA}} = 1\%$ , as a function of the intrinsic accuracy of the process and measurement

noise. For Gaussian noise the probability of detection is  $P_D = 48\%$ . In this way, Figure 9.1 is similar to the figures in Chapter 7 which illustrate the potential performance gain

$$\frac{\kappa(\mathcal{I}_w^{-1}, \mathcal{I}_e^{-1})}{\kappa(\text{cov}(w), \text{cov}(e))},$$

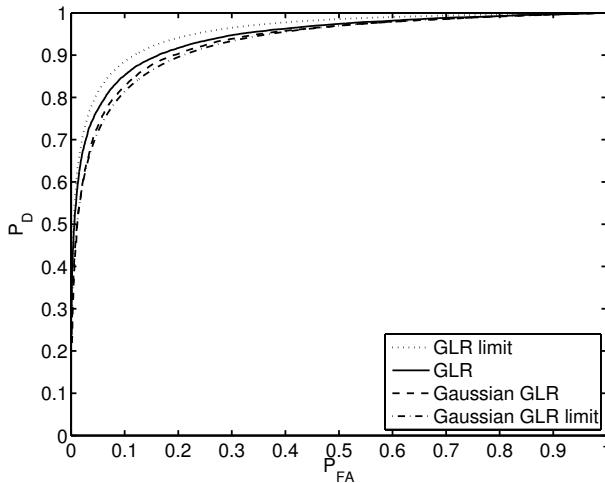
both types of plots illustrates the effect non-Gaussian noise characteristics have on the potential performance. Note that with a different  $P_{FA}$  or magnitude of the fault, a different result is obtained due to the nonlinearities present in the functions to compute  $P_D$  for a given  $P_{FA}$ , but the main characteristics of the result will remain. Figure 9.1 indicates that for non-Gaussian noise it is asymptotically possible to gain detection performance. Furthermore, the performance increases faster in the relative accuracy of the process noise than in the relative accuracy of the measurement noise. This indicates that describing the process noise correctly is more important for the performance than describing the measurement noise.

### 9.1.1 Bimodal Measurement Noise

The first simulation, using the constant velocity model, features Gaussian process noise,  $\Psi_w = 1$ , and the non-Gaussian measurement noise

$$e_t \sim \frac{9}{10}\mathcal{N}(0.2, 0.3) + \frac{1}{10}\mathcal{N}(-1.8, 3.7),$$

with  $\Psi_e = 2.7$ . Figure 9.1 indicates that with this setup the detection performance should go from  $P_D = 47\%$  to  $P_D = 60\%$  for  $P_{FA} = 1\%$ . This suggests that the non-Gaussian measurement noise should be handled in the detector.



**Figure 9.2:** ROC diagram derived from 1 000 Monte Carlo simulations for the parity space formulation of the constant velocity model (9.1), with  $L = 6$ , the fault  $f^1 = 1.5$ , and bi-Gaussian measurement noise.

Figure 9.2 shows the ROC diagram obtained from 1 000 Monte Carlo simulations. The Gaussian GLR curve has been obtained by assuming Gaussian noise when designing the detector, that is Gaussian PDFs are used to create the likelihood ratio. The GLR test uses numerical maximization to find the likelihood of the residual under the alternative hypothesis. The ROC diagram shows that, by utilizing the extra information available in the non-Gaussian measurement noise, the probability of detection is improved, however not as much as indicated by the asymptotic results. Furthermore, the performance obtained when assuming Gaussian noise is better than expected for large  $P_{FA}$ . For  $P_{FA}$  close to 1, it is difficult to tell the difference between the approximative test and the test based on correct statistics. Hence, the behavior when using a Gaussian approximation is not as predictable as for estimation, where the BLUE can be used.

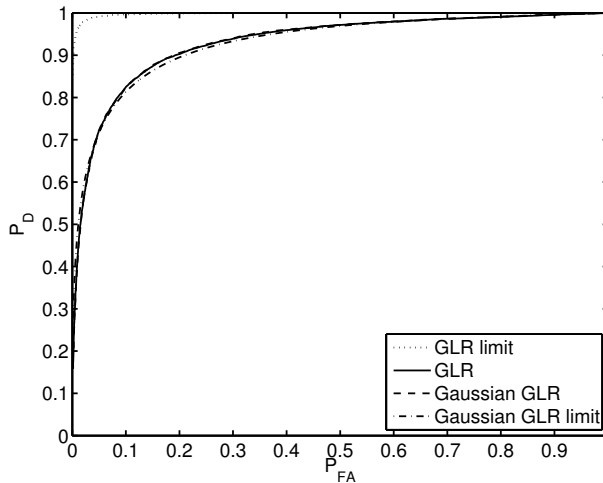
### 9.1.2 Tri-Gaussian Process Noise

For the second simulation using the constant velocity model the measurement noise,  $e_t$ , is kept Gaussian,  $\Psi_e = 1$ , and the process noise is tri-Gaussian

$$w_t \sim 0.075\mathcal{N}(-2.5, 0.065) + 0.85\mathcal{N}(0, 0.065) + 0.075\mathcal{N}(+2.5, 0.065),$$

with  $\Psi_w = 15.4$ . Figure 9.1 suggests that this will improve the probability of detection from 48% to 95% with 1% probability of false alarm. To use the correct process noise description should hence improve detection performance.

The ROC diagram in Figure 9.3 is the result of 1 000 Monte Carlo simulations. In this case practically no difference can be observed between the GLR test that uses a correct noise description and the one using the Gaussian approximation. Hence, there is no improvement achieved from correctly modeling the non-Gaussian process noise. The reason



**Figure 9.3:** ROC diagram derived from 1 000 Monte Carlo simulations for the parity space formulation of the constant velocity model (9.1),  $L = 6$ , the fault  $f^1 = 1.5$ , and tri-Gaussian process noise.

for this is probably that the asymptotic information assumption of the GLR test is poorly met. Similar results (not presented here) have been observed for filtering, where it is very difficult to perform better than the BLUE with this setup.

### 9.1.3 Conclusions

Two simulation studies of the constant velocity model (9.1) have conducted. The result in the first of them shows that it is possible to gain performance by modeling non-Gaussian effects. In the second simulation no apparent gain is achieved. Hence, improvements are possible but not guaranteed. It is not easy to predict the performance that will be obtained under an approximate noise assumption or how difficult it is to reach the asymptotic GLR performance.

## 9.2 DC Motor

The second studied system is the DC motor previously evaluated with respect to estimation properties in Chapter 7. The DC motor is affected by a load disturbance,  $f_t$ , according to the model

$$x_{t+1} = \begin{pmatrix} 1 & 1 - \alpha \\ 0 & \alpha \end{pmatrix} x_t + \begin{pmatrix} T - (1 - \alpha) \\ 1 - \alpha \end{pmatrix} f_t + \begin{pmatrix} T - (1 - \alpha) \\ 1 - \alpha \end{pmatrix} w_t \quad (9.2a)$$

$$y_t = (1 \ 0) x_t + e_t, \quad (9.2b)$$

where  $\alpha = \exp(-T)$ ,  $T = 0.4$ , and  $w_t$  and  $e_t$  are independent, but not necessarily Gaussian noises. In the nominal model  $w_t \sim \mathcal{N}(0, (\frac{\pi}{180})^2)$  and  $e_t \sim \mathcal{N}(0, (\frac{\pi}{180})^2)$ . The fault enters the system in a way that could be caused by an extra load on the drive shaft, a bad bearing, or an offset in the voltage used to control the motor.

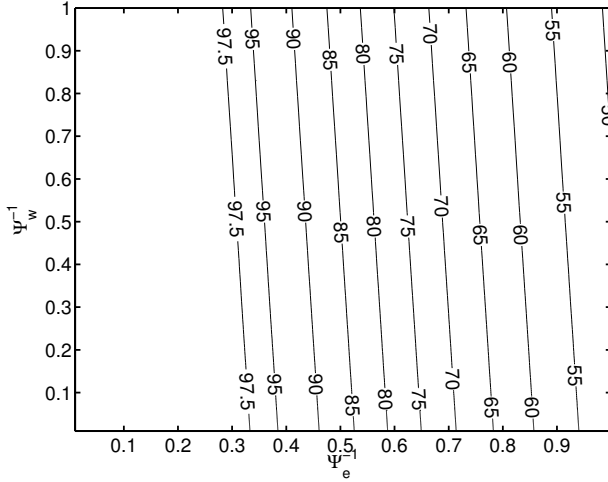
The fault is in this case assumed to be incipient,

$$f_t = \begin{cases} 0, & t \leq 20T \\ (t - 20T)/(100T), & 20T < t \leq 30T \\ 1/10, & 30T < t, \end{cases} \quad (9.3)$$

where  $T = 0.4$ . The result is an acceleration of the motor, e.g., due to an offset in the control input. The fault magnitude is modeled using a Chebyshev polynomial basis with two base vectors handling affine fault behavior. How to use this basis for the fault magnitude is described in Section 4.3.

Under these assumptions the probability of detection, for  $P_{FA} = 1\%$  and  $f_t = \frac{1}{10}$ , as a function of the intrinsic accuracy of the involved noise, is given in Figure 9.4. According to the figure it will be difficult to improve the detection performance by improving the process noise description.

It will now be investigated, with two different noise configurations, if it is possible to obtain improved detection performance taking the true noise characteristics into account.



**Figure 9.4:**  $P_D$  (in %) for the DC motor (9.2) given  $P_{FA} = 1\%$  for  $f_t = \frac{1}{10}$ , i.e.,  $\theta = (\sqrt{6}/10, 0)^T$  in the Chebyshev polynomial basis, and a window of length  $L = 6$ . For Gaussian noise  $P_D = 47\%$ .

## 9.2.1 Measurements with Outliers

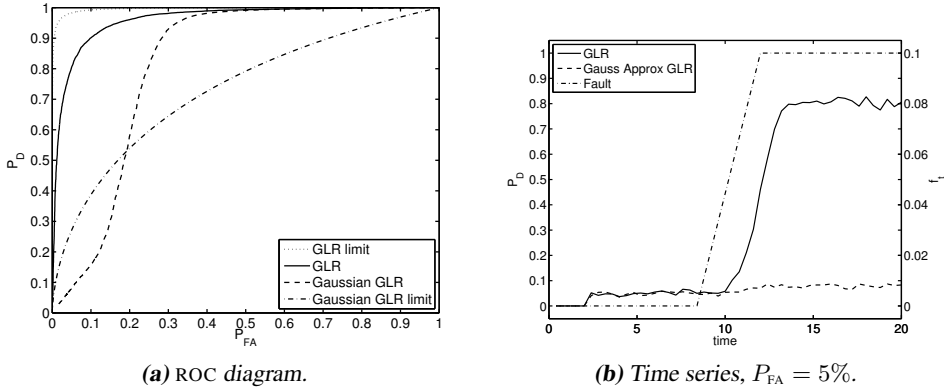
The first setup involves Gaussian process noise and measurements with outliers (cf. Section 7.5.1),

$$\begin{aligned} w_t &\sim \mathcal{N}\left(0, \left(\frac{\pi}{180}\right)^2\right) \\ e_t &\sim 0.9\mathcal{N}\left(0, \left(\frac{\pi}{180}\right)^2\right) + 0.1\mathcal{N}\left(0, \left(\frac{10\pi}{180}\right)^2\right), \end{aligned}$$

with  $\Psi_w = 1$  and  $\Psi_e = 9$ . The contour plot in Figure 9.4 indicates that detection performance could be improved substantially, approximately from 50% to almost 100%, if the outliers are correctly handled in the test. Note however that Figure 9.4 is derived for the nominal noise levels and that the outliers increases the variance of the measurement noise. Computing the values for the new noise levels gives a gain from  $P_D = 27\%$  to 97%. Hence, the figure is only approximately correct, but gives an indication of when performance can be gained.

The result of the simulations is presented in the ROC diagram in Figure 9.5(a). Especially for low false alarm rates,  $P_{FA} \ll 1$ , the gain is substantial and close to what is possible to achieve with an asymptotic GLR test. Considering detection in terms of the Wald test the good detection performance is not surprising since it was shown in Section 7.5.1 that the estimation performance for this setup is close to the CRLB.

Once again, the GLR statistics under the Gaussian approximation proves to be a rather poor description of the behavior when non-Gaussian components are neglected. For low probability of false alarm the detector derived with the noise approximation performs worse than the asymptotic GLR test predicts, whereas for large  $P_{FA}$ , the approximate test is better. This only shows that it is not easy to predict how detection performance is affected by noise approximations.



**Figure 9.5:** Simulation results from 1 000 Monte Carlo simulations of the DC motor with a parity space formulation for a window  $L = 6$ . The fault is modeled with a Chebyshev polynomial basis with 2 components and  $f^1 = \frac{1}{10}$ . The measurement noise has outliers,  $\Psi_e = 9$ .

Figure 9.5(b) gives the probability of detection over time, as  $P_{FA} = 5\%$ . Note that to obtain this probability of false alarm different thresholds had to be used for the different detectors. The plot illustrates the difference in behavior, and that the detection delay in this case is approximately half the window size.

### 9.2.2 Load Disturbances

In the second study of the DC motor the measurement noise is Gaussian,  $\Psi_e = 1$ , and the process noise bimodal,  $\Psi = 17$ ,

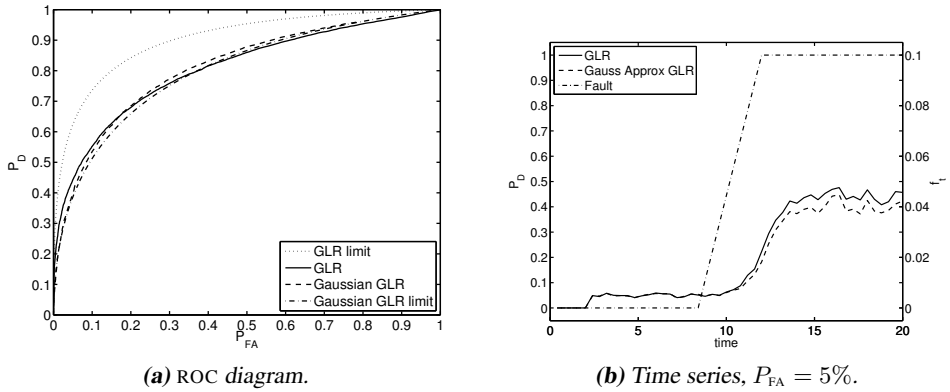
$$w_t \sim 0.8\mathcal{N}\left(0, \left(\frac{\pi}{180}\right)^2\right) + 0.2\mathcal{N}\left(\frac{-10\pi}{180}, \left(\frac{\pi}{180}\right)^2\right)$$

$$e_t \sim \mathcal{N}\left(0, \left(\frac{\pi}{180}\right)^2\right).$$

The same system was studied in Section 7.5.2 with respect to estimation performance. In that case the performance was improved. Based on the previous detection experiments, and the indication in Figure 9.4 that the detection performance could be improved using the actual noise distributions for the test statistic, but not very much. This is surprising given the improvements that were obtained for filtering in Section 7.5.2.

The result of 1 000 Monte Carlo simulations is found as a ROC diagram in Figure 9.6(a). The performance using the correct noise distributions is only a little better than the performance achieved approximating all distributions with Gaussian noise. Furthermore, there is a clear difference between the asymptotic GLR test and the ROC achieved. One possible explanation is that the numerical maximization used has issues with the multimodal distribution for the residuals.

Computing the  $P_D$  as a function of time for  $P_{FA}$  shows the same behavior as the ROC diagram. There is no great difference between the detector based on the true noise description and the noise approximation.



**Figure 9.6:** Simulation results from 1 000 Monte Carlo simulations of the DC motor with a parity space formulation for a window  $L = 6$ . The fault is modeled with a Chebyshev polynomial basis with 2 components and  $f^1 = \frac{1}{10}$ . The process noise is bimodal,  $\Psi_w = 17$ .

### 9.2.3 Conclusions

It has in two simulation studies of a DC motor been demonstrated that it is possible to gain detection performance by taking non-Gaussian effects into consideration. In the case with outliers in the measurements the improvement was clear compared to the result assuming Gaussian noise. In the second case where the process noise was non-Gaussian there was no gain. Hence, it is not always easy to meet the asymptotic assumptions of the GLR test.

## 9.3 Range-Only Simulation

The last simulation study in this chapter is of another type than the previous two. Here the ability to compute the probability of an event is evaluated,

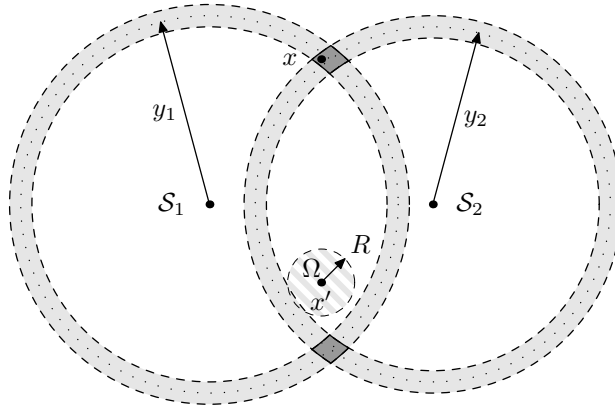
$$\Pr(\mathcal{A}(x_t)) = \int_{\Omega} p(x_t | \mathbb{Y}_t) dx_t. \quad (9.4)$$

To be able to compute  $\Pr(\mathcal{A}(x_t))$  it is important to have an accurate estimate of the posterior distribution. To compute the probability from a particle filter or a point-mass filter (used here to represent the truth) is a matter of summing the weights in the interesting region of the state-space, whereas for filters that approximate the posterior with a Gaussian distribution numeric integration is needed.

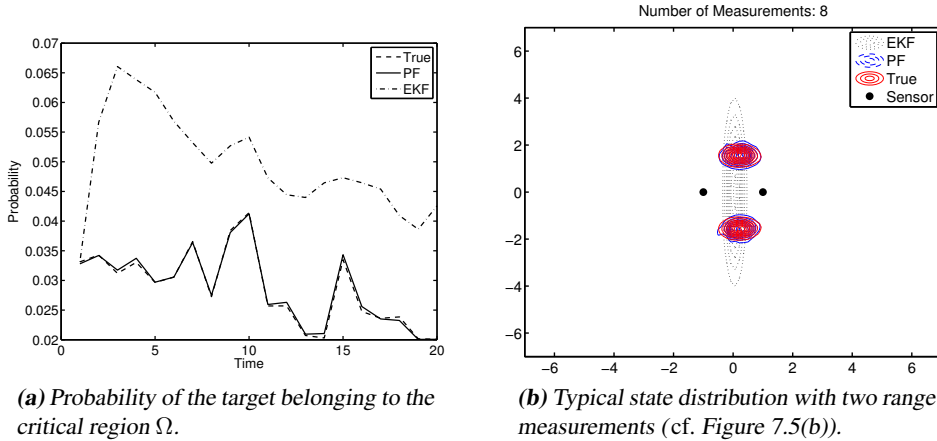
To evaluate this the range-only tracking application from Section 7.3 is revisited,

$$x_{t+1} = x_t + w_t \quad (9.5a)$$

$$y_t = \begin{pmatrix} \|x_t - \mathcal{S}_1\|_2 \\ \|x_t - \mathcal{S}_2\|_2 \end{pmatrix} + e_t, \quad (9.5b)$$



**Figure 9.7:** Range-only tracking setup for detecting target in  $\Omega$ .



**(a)** Probability of the target belonging to the critical region  $\Omega$ .

**(b)** Typical state distribution with two range measurements (cf. Figure 7.5(b)).

**Figure 9.8:** Results from 100 Monte Carlo simulations of range-only tracking.

with  $w_t \sim \mathcal{N}(0, 0.1I)$ ,  $e_t \sim \mathcal{N}(0, 0.1I)$ , and initial knowledge  $x_0 \sim \mathcal{N}(0, 3I_2)$  with the objective to compute the probability that the object is in the region  $\Omega = \{x : \|x - x'\|_2 < R\}$  with  $x' = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  and  $R = \frac{1}{2}$ , as illustrated in Figure 9.7. The sensors are located in  $S_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$  and  $S_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ .

The ability to correctly compute this probability is evaluated in 100 Monte Carlo simulations using an EKF, an particle filter (SIR, 20 000 particles<sup>1</sup>), and the truth is obtained from a very finely gridded point-mass filter. The average result of the simulations is displayed in Figure 9.8(a). Note how the EKF throughout the whole simulation indicates a much higher probability than it should, and that the particle filter is very close to the truth. The lack of descriptive power of the EKF is here very apparent. One reason for the

<sup>1</sup>The number of particles has been chosen excessively large to get an accurate PDF estimate, with further tuning could be reduced significantly.

poor behavior of the EKF is indicated in Figure 9.8(b), which displays a typical posterior distribution. The difference between the truth and the EKF is large. The EKF indicates much higher probability between the modes, close to the  $x$ -axis, than it should, which results in a too large probability. The particle filter on the other hand does better capturing the bimodal posterior distribution, which is reflected in a much better estimate of the probability.

## 9.4 Summary

Three simulation studies have been conducted in this chapter. Two of these have been used to verify the theoretical results in Chapter 8, whereas the third one clearly indicates the importance of properly estimating the posterior distribution in some situations.

The results from the simulations conducted with the constant velocity model and the DC motor can be summarized with:

- By studying the asymptotic GLR performance, it is indicated that there is potentially much to be gained from taking care of non-Gaussian effects.
- It is not enough to study the asymptotic GLR test performance for a system with Gaussian noise to predict the effects of a Gaussian approximation. The simulations indicate that this approach is often overly optimistic for  $P_{FA} \ll 1$ , whereas it for  $P_{FA} \gg 0$  is too pessimistic. With this in mind, the asymptotic GLR test with Gaussian noise can be used to hint the behavior of an approximative detector.
- It is in the conducted experiments more difficult to extract information from non-Gaussian process noise compared to non-Gaussian measurement noise.
- The simulations indicate that it is for small  $P_{FA}$  the most is gained by using a correct noise description.

The result of the range-only tracking simulation is to clearly show that there are problem formulations where the actual posterior distribution is of large importance, and hence the choice of method is very important.

# 10

---

## Implementation Details

**W**HEN USING A NAVIGATION system the user expects to get rapid response from the system, the position a few minutes ago is of less or no interest when trying to navigate through the streets of a busy city. For this to be possible it is important to have a good and powerful algorithm to get a good estimate of the position. However, this is not enough, it is also important to make sure that the result comes within reasonable time. The latter is true for most signal processing applications that interact with people.

Up until now this thesis has mostly regarded theoretical aspects of filtering and detection. However, in order for signal processing to be of any practical use it is important to have correct and efficient implementations of the algorithms available. This chapter will describe two different implementations. The first is a general purpose filtering framework for state-space estimation written in C++. The purpose is mainly to provide an easy to use environment for large simulations, that are often infeasible to do in MATLAB®. The second is parallel implementation of a particle filter on a *graphics processing unit* (GPU), as found in most new computers. This implementation was the first of its kind reported.

In this chapter the filtering framework is presented in Section 10.1 and the parallel particle filter implementation in Section 10.2.

### 10.1 A c++ filtering framework — F++

The *de facto* standard tool for scientific computations, especially in the field of engineering, has become MATLAB®. It has many advantages. One of them being a large user-base and another one that many toolboxes exist to perform various common nontrivial tasks. Yet another advantage is that it is relatively easy to learn and it is possible to express thoughts in a way close to how an engineer would do on paper. However, this simplicity of use comes with some drawbacks, such as a too forgiving syntax that makes debugging difficult, and a large run time overhead compared to compiled code. The latter can make

MATLAB® infeasible for use in large Monte Carlo simulation studies and user products. Under these circumstances, compiled code is often needed, and this comes with an extra level of complexity.

To support large Monte Carlo simulations, an object-oriented framework for filtering has been designed using modern software design methodology. The main goal with the framework is to provide a high-level interface to a compiled language, in this case C++, that is easily accessible without extensive programming experience, at the same time as it is readily expandable when needed. Depending on the usage it is reasonable to expect a speed-up of at least 2–3 times, and under extremely advantageous circumstances 10–100 times. This improvement comes at a relatively low cost in terms of it being more difficult to write the code and that it needs to be compiled before it can be run. The framework to do this is called F++<sup>1</sup>.

In this section the main design idea will be described, and a short and a longer example will demonstrate the ease of usage.

### 10.1.1 Design

The framework is designed using modern computer science theories and methodology. To allow for as much flexibility as possible and to make maintenance and expansion as easy as possible, object orientation is used. Inheritance makes it possible to express relations between types with the same purpose but different implementations. For example, in a filtering setup, it should make no difference if the estimator is an EKF or a particle filter, and to change between the two should be as transparent as possible.

Only object orientation does not achieve this in C++, it leaves too much book keeping for the user to be really useful. Therefore F++ uses reference counting proxies to hide the memory management and pointer arithmetic from the end user. This and many other solutions used are examples of software *design patterns* [38], that is, well tested solutions to generic problems. Using design patterns, years of experience, knowledge, and testing, are incorporated into a project at minimal cost.

### 10.1.2 Class Description

The elements of the kind of signal processing applications that are discussed in this thesis can be summarized with the following conceptually important object types:

**Noise** — representing all types of stochastic phenomena.

**Model** — describing system behaviors.

**Estimator** — different methods, filters, to extract state information from measurements.

**Detector** — different methods to perform fault/change detection. (Could be seen as a special case of an estimator.)

**Data** — comprising states, estimates, measurements, *etc.*, with or without time stamp.

---

<sup>1</sup>The F++ homepage is <http://www.control.isy.liu.se/resources/f++>, from where F++ can be acquired.

The design choice in the filtering framework is to model each of these concepts as a class tree (there is so far no detectors available). This gives a natural division into parts that suit users with a signal processing background, at the same time as it is a sensible choice from a computer science point of view as well, [119].

There is also a whole framework of helper classes providing support for counted access to objects, convenient access to different file formats, random number generation, *etc.* There is for example a level of abstraction that lets the user work with noises, models, and estimators as if they were regular objects, while a reference counting proxy keeps track of the underlying pointer representation needed for the implementation. Furthermore, object factories allows for objects to be read and saved to file using different file formats.

The presentation in this section focuses on the representation of noises, models, and estimators as that is what is most important from a signal processing point of view; other aspects of the code, though sometimes interesting from a computer science perspective, have been left out of the description.

## Noise

The interface for users of the framework to all noise objects is the class `Noise`. However, when implementing new noise classes they should be derived from the mother class `NoiseImpl`, which defines a common interface for all other noise classes. A new noise must provide functionality to:

- generate samples,
- compute the PDF and CDF in a given point, and
- return its own mean and (co)variance.

A number of different noises are available with F++:

- `GaussNoise`, implements a standard Gaussian variable as described in Section 2.4.1;
- `GaussSumNoise`, a Gaussian sum noise, see Section 2.4.2;
- `SumNoise`, which is similar to the Gaussian sum noise but allows for greater freedom in the choice of the combined distributions; and
- `ComboNoise`, a stacked noise, which allows for stacking independent stochastic variables into a larger stochastic variable  $x = (x_1^T \quad x_2^T \quad \dots \quad x_n^T)^T$ .

Finally, there is also the possibility to have `DeterministicNoise`, a Dirac distribution, just to allow for the situation where no noise is present but a noise object is required.

The following example provides the complete code listing for a ready F++ program that demonstrates some features of noise objects. The complete listing is presented here to give the reader an idea how much code must be produced to get a complete program. Remaining listings in this section will only present selected parts of special interest to the discussion. The complete listings can be found in Appendix B.

---

**Example 10.1: Creating and using a noise object in F++**


---

Listing 10.1 contains all source code needed to generate and use a noise in F++.

---

*Listing 10.1: Noise usage example: noiseex.cc.*


---

```

1  #include "noise/noise.h"
   #include "io/xmlfile.h"
   #include <iostream>

5  using namespace std;

   int main(int argc, char* argv[])
   {
10     XMLfile is("noiseex.xml", XMLfile::read);

       Noise E(is, argv[1]);

       Noise::vector_type e = E.random();

15     cout << "NOISE: " << argv[1] << endl
           << "Random sample (e):\t" << e << endl
           << "           p(e):\t" << E.pdf(e) << endl
           << "   Expected value:\t" << E.mean() << endl
           << "           Variance:" << endl
20     << E.var() << endl
           << endl;

       return 0;
   }

```

---

Important lines in source code are:

- 9:** Open the F++ XML file `noiseex.xml` for reading.
- 11:** Read a noise object from the file, using the first argument on the command line to determine which object to read.
- 13:** Draw a random number from the distribution and save it in `e`.
- 17:** Compute and display the PDF in `e`.
- 18, 20:** Display mean and covariance of the distribution.

The remaining rows are just template code that could be copied from any example program.

To use the program with several different noises all that is needed is to add more noise descriptions to `noiseex.xml`, and supply the name of the interesting noise when running the program. An example file containing a Gaussian is given in Listing 10.2.

---

*Listing 10.2: Example of input for noiseex.cc: noiseex.xml*


---

```

1  <object name='E' class='Noise::GaussNoise'>
   <vector name='mu' dim='1'>
     0
   </vector>
5  <matrix name='Sigma' rows='1' cols='1'>
     1
   </matrix>
</object>

```

---

The result from compiling and then executing this code for the different noises in the file (the complete `noiseex.xml` contains three noise descriptions) yields Listing 10.3.

**Listing 10.3:** *Example of running noiseex*

```

1 > ./noiseex E
  NOISE: E
  Random sample (e): [0.133919]
                    p(e): 0.395381
5   Expected value: [0]
      Variance:
1
    1

> ./noiseex E2
10 NOISE: E2
    Random sample (e): [2.18939,0.0323979]
                    p(e): 0.157123
      Expected value: [2,0]
      Variance:
15 2 1
    1 1

> ./noiseex Esum
20 NOISE: Esum
    Random sample (e): [2.28038]
                    p(e): 0.0756379
      Expected value: [0.2]
      Variance:
4.66

```

## Model

Models are by F++ users accessed through the class `Model`, which is a reference-counted interface class to a `ModelImpl` pointer. New models are constructed by deriving new classes with `ModelImpl` as mother class. Typically, a model owns at least two noise objects to describe process and measurement noise. The framework comes with three implemented model classes:

- `LinModel`, a linear model, *cf.* Section 4.2.1, which is initiated with the three system matrices  $F$ ,  $G$ , and  $H$ , and process and measurement noise;
- `MultiModel`, a switched model, *cf.* Section 4.2.2, comprising several different other models switching on the input; and
- `GenericModel`, a general model where templates are used to allow for any functions implementing dynamics and measurement to be combined to a model.

The generality of the last model comes with a performance price, and it is therefore often a good idea to implement new nonlinear models as new classes. If a user decides to implement new functionality to F++ it will most likely be a model.

When designing a new model, the following functions must be supplied:

- the dynamics, *i.e.*, how to propagate the state through time given the previous state and input;

- the measurement equation;
- access to the process and measurement noise the model uses;
- gradients of the dynamics and the measurements, this is however only necessary if these functions will be used.

An example of a customized model is given by the two Listings B.3 and B.4 in Appendix B. They implement the model for the range-only tracking example described in Section 7.3. These two source code listings are quite lengthy, but most of the code could be copied and pasted from any other model implementation. The interesting lines can be found in Listing 10.4, where lines 1–6 implement the system dynamics and lines 7–16 implement the measurement equation. The explicit inclusion of a time stamp with data adds some complexity but prepares the system for handling out of sequence data and irregular sampling. The variables `S1_` and `S2_` are the sensors locations.

---

**Listing 10.4:** *Parts of range-only measurement model code: romodel.cc*

---

```

1 RangeOnlyModel::vector_t RangeOnlyModel::f(const vector_t& x,
                                             const vector_t& w,
                                             const vector_t& u) const
{
5   return vector_t(x.datum + w.datum, x.time+1);
}

RangeOnlyModel::vector_t RangeOnlyModel::h(const vector_t& x,
                                             const vector_t& e,
                                             const vector_t& u) const
10 {
    vector_type y(2);
    y[0] = (S1_-x.datum).length();
    y[1] = (S2_-x.datum).length();
15   return vector_t(y, x.time);
}

```

---

To use the range-only measurement model together with an EKF, more code must be written to define gradients. Listing 10.5 provides the code. Once again, handling the time stamp makes the code listing longer, but allows for greater freedom in how C++ can be used.

---

**Listing 10.5:** *Parts of range-only measurement model code: romodel.cc*

---

```

1 RangeOnlyModel::matrix_t RangeOnlyModel::h_x(const vector_t& x,
                                                const vector_t& e,
                                                const vector_t& u) const
{
5   vector_type d1 = x.datum-S1_;
   vector_type d2 = x.datum-S2_;
   double r1 = d1.length();
   double r2 = d2.length();
   matrix_type grad(2,2);
10   grad(0, 0) = d1[0]/r1;   grad(0, 1) = d1[1]/r1;
   grad(1, 0) = d2[0]/r2;   grad(1, 1) = d2[1]/r2;
   return matrix_t(grad, x.time);
}

```

---

## Estimator

Filtering in F++ is performed using an Estimator object. The Estimator class is a reference counted interface to classes derived from EstimatorImpl. The filtering framework does provide five different filters:

- particle filter, PF, see Section 5.2;
- (extended) Kalman filter, EKF, see Sections 5.3 and 5.4;
- unscented Kalman filter, UKF, see Section 5.4.4;
- interacting multiple model filter, IMM, see Section 5.5;
- Rao-Blackwellized particle filter<sup>2</sup>, RBPF, see Section 5.6.

Every estimator should provide functions to:

- obtain a point estimate,
- a covariance matrix for this estimate,
- the likelihood of a measurement, and
- implement the time update and the measurement update steps of the filtering loop.

How these primitives can be used to create a filtering loop is illustrated in Listing 10.6, which is a small part of the code needed for the example presented in the next section.

**Listing 10.6:** Part of range-only measurement simulation code: *rangeonly.cc*

---

```

1   for ( cItr y = Y[i].begin(); y!=Y[i].end(); ++y) {
      // PF filtering
      pf.mUpdate(*y);
      Xhatpf[i].push_back(pf.xhat());
5   pf.tUpdate();
  }
```

---

The purpose of the code is explained line by line (*i* is the current simulation):

- 1:** Start a loop over all available measurements.
- 3:** Perform a measurement update.
- 4:** Extract and save a point estimate from the filter.
- 5:** Time update the filter.

### 10.1.3 Example: Range-Only Measurement Tracking

The range only measurement problem, previously discussed in Section 7.3, will in this section be revisited, and an implementation in F++ is presented. Surprisingly few lines of code are needed, even though the post-processing of the results, in this case RMSE computations, are done in F++, and the model for the system does not directly fit into one of the available ones, so a new one must be derived.

The code setting up the problem, running the Monte Carlo simulations, and doing the post-processing is found in Listing B.1 in Appendix B, and the input data in Listing B.2. Parts of the code of principal interest will be discussed in this section, and for a complete picture the reader is referred to Appendix B for complete listings of the source code.

---

<sup>2</sup>Trying to implement the RBPF in F++ was the starting point for deriving the formulation of the RBPF found in Section 5.6 of this thesis.

Listing 10.7 shows how the Monte Carlo simulation is set up: at lines 2 and 3 the true trajectory and measurements are declared; at lines 5 and 6 the true trajectory is generated and is then used to generate measurements; and at lines 10 and 11 the two filters are declared.

**Listing 10.7:** Part of range-only measurement simulation code: *rangeonly.cc*

---

```

1 // Create truth for the simulations
  Data<Data<Model::vector_t> > Xtrue;
  Data<Data<Model::vector_t> > Y;
  for (long i=0; i<NMC; ++i) {
5   Xtrue.push_back(generate_trajectory(model, x0.random(), tfin));
   Y.push_back(generate_measurement(model, Xtrue.back()));
  }
  // *** DO MONTE CARLO SIMULATIONS ***
  for (long i=0; i<NMC; ++i) {
10   Estimator pf = createPF(model, 10000, x0, "ParticleFilter");
   Estimator ekf = createEKF(model, x0, "ExtendedKalmanFilter");
   // PF filtering
   // EKF filtering
  }
15 }

```

---

The post-processing of the data is done with the source code in Listing 10.8. The RMSE computations are done on lines 1 and 2 for the particle filter and EKF, respectively. Line 4 opens a MATLAB® file for writing, and the results are written to file on line 5 and 6.

**Listing 10.8:** Part of range-only measurement simulation code: *rangeonly.cc*

---

```

1 Data<double> RMSEpf = rmse(Xtrue, Xhatpf);
  Data<double> RMSEekf = rmse(Xtrue, Xhatekf);

  MATfile os("rangeonly.out", MATfile::write);
5 write(os, "RMSEpf", RMSEpf);
  write(os, "RMSEekf", RMSEekf);

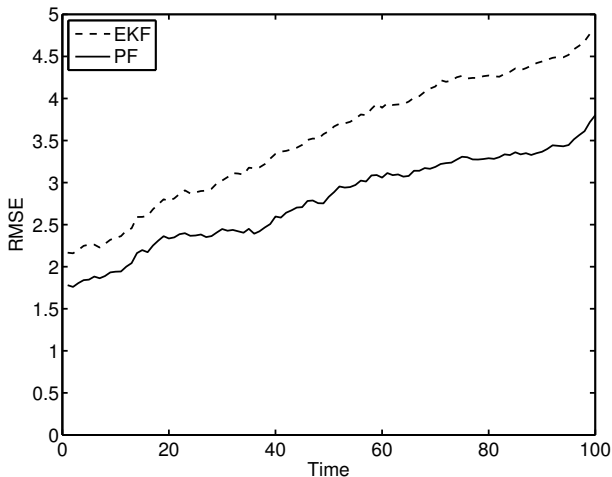
```

---

The model needed for the range-only measurement setup was previously discussed in connection with how models are implemented. In total, less than 50 non-trivial lines of code are necessary to complete this Monte Carlo simulation of the range-only tracking problem. It is hence not much more work to implement and do the simulations in F++ compared to MATLAB®.

The result from the simulations are shown in Figure 10.1. The reader is referred to Section 7.3 for a discussion about the results, this section is meant to present how the same problem was solved using F++.

No formal experiments have yet been conducted to compare the speed obtained with F++ to code written in for example MATLAB®. The general experience from using programs in F++ in everyday simulations is that it improves performance, especially when the system involves many nonlinear components. The authors of F++ have since initiating the work on F++ used it for various simulation studies, some of which would not have been feasible to perform in MATLAB®, e.g., the simulations in [56], all indicating that F++ delivers good performance.



**Figure 10.1:** Result of Monte Carlo simulations of the range-only measurement problem.

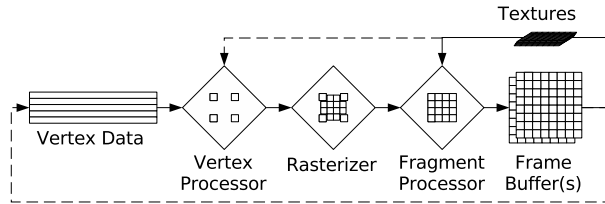
## 10.2 Using Graphics Hardware for Particle Filtering

The last few decades the role of computers has changed. A computer is today a competent multi-media machine capable of rendering complex graphics for games, displaying and editing movies, and much more. This in addition to what is to be considered as more traditional computer tasks, e.g., complex computations, text editing, and very repetitive and error prone exercises. The development has changed the layout of what comprises a computer. One example of what has improved drastically is the *graphics processing unit* (GPU) hardware used to display graphics. Still, the price of the hardware is relatively low and affordable. A strong driving force for this has been the ever increasing demands from computer game industry to make more realistic games.

Modern GPUs are designed to handle huge amounts of data about an often complex scene and to render output to screen in real time. To achieve this, the GPU is equipped with a *single instruction multiple data* (SIMD) parallel instruction set architecture.<sup>3</sup> GPUs are developing rapidly in order to meet the ever increasing demands from the computer game industry, and as a side-effect, *general-purpose computing on graphics processing units* (GPGPU) has emerged to utilize this new source of computational power [43, 97, 109]. For highly parallelizable algorithms the GPU may even outperform the sequential *central processing unit* (CPU).

The particle filter as previously described in Section 5.2, is in large parts very parallel in its structure. A major part of the algorithm is to perform identical operations on many particles. From this point of view the particle filtering algorithm is well suited for parallelization. Successful parallelization may lead to a drastic reduction of computation time and open up for new applications requiring large state-space descriptions with

<sup>3</sup>This has changed slightly with the newest generation of GPUs, however this thesis will not address these changes.



**Figure 10.2:** The graphics pipeline. The vertex and fragment processors can be programmed with user code which will be evaluated in parallel on several pipelines.

many particles. Nonetheless, filtering and estimation algorithms have only recently been investigated in this context, see for instance [96, 98].

There are many types of parallel hardware available nowadays; examples include multicore processors, *field-programmable gate arrays* (FPGAs), computer clusters, and GPUs. GPUs offer low cost and easily accessible SIMD parallel hardware — almost every new computer comes with a decent graphics card. Hence, GPUs are an interesting option for speeding up a particle filter and to test parallel implementations. A first GPU implementation of the particle filter was reported in [99] for a visual tracking computer vision application. In contrast, in this paper a general particle filter GPU implementation is developed. To the best of the author’s knowledge no successful complete implementation of a general particle filter on a GPU was previously reported when this material was first published in [59]. This section is GPGPU techniques are used to implement a particle filter on a GPU and its performance is compared to that of a CPU implementation.

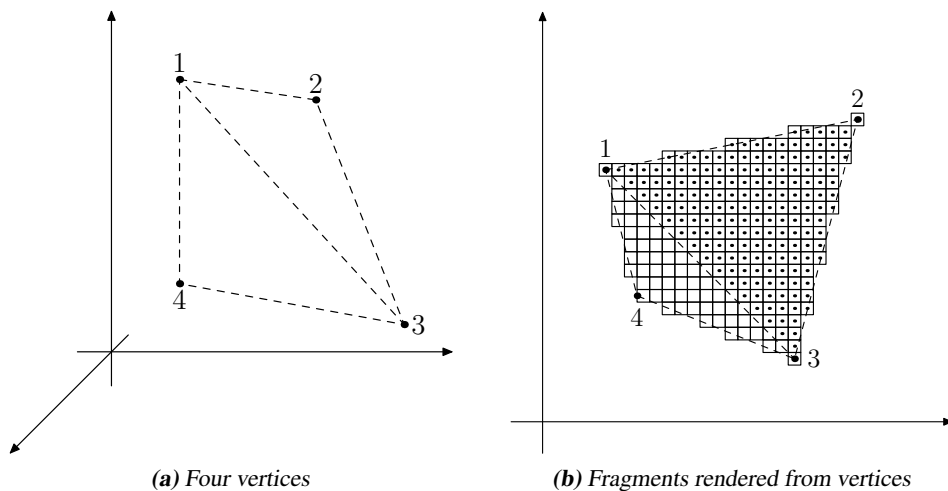
## 10.2.1 General Purpose Graphics Programming

Graphics processing units operate according to the standardized *graphics pipeline* (see Figure 10.2), which is implemented at hardware level [109]. This pipeline, which defines how the graphics should be processed, is highly optimized for the typical graphics application, *i.e.*, displaying 3D objects. The purpose of the different steps are described below.

### Graphics Hardware

Graphics cards are designed to primarily produce graphics, which makes their design different from general purpose hardware, such as a CPU. The GPU is designed around the standardized graphics pipeline, as depicted in Figure 10.2 [109]. It consists of three processing steps, that all have their own purpose when it comes to producing graphics, and some dedicated memory units. From having predetermined functionality, GPUs have moved towards providing more and more freedom for the programmer to decide what tasks should be performed. Recent graphics cards allow for customized code in two out of the three computational units: the vertex shader and the fragment shader.

The graphics pipeline is fed with vertices and matching properties. The vertices are connected in the *vertex shader* to create polygons, see Figure 10.3(a). The properties include information about the color, texture, and normal vector of the surface. It is further



**Figure 10.3:** Illustration of vertices and fragments as used by the graphics pipeline to render graphics.

possible to compute new properties for the vertices. This could include specific normal vectors in the vertices, color and/or shading values, and move the points. The standard behavior is to translate, scale, and rotate the vertices to account for watching the scene through a camera.

The next step in the pipeline is the *rastering*, a step hard coded into the GPU that cannot be changed by a programmer. It takes the polygons defined by the (transformed) vertices and fills them with fragments, as illustrated in Figure 10.3(b). Fragments are the potential pixels that will make up the final picture. Not all fragments will make it that far, but be thrown away later because they are hidden behind other fragments. The number of fragments rendered is usually orders of magnitude larger than the number of vertices passed to the GPU.

The fragments are then processed in the *fragment processor*. This is a highly programmable part of the GPU. Here the color, transparency, *etc.* of the fragment is computed, and then sent off to be displayed. Often *textures*, dedicated memory units, are used to put pictures on surfaces. To support this, the GPU has built in support for effective interpolation in the textures. It is also possible to drop certain fragments before they reach the final destination. The result is written to the *frame buffer*. The frame buffer can be either connected to the display, or a texture. The latter allows for scenes that are iteratively computed, but it is also a convenient way to extract data from the GPU to the CPU.

The computations within the steps of the pipeline are performed in parallel with SIMD instructions, *i.e.*, the hardware supports doing the same operation on many data at the same time. For the vertex processor this means that all vertices can potentially be computed simultaneously, and a vertex computation cannot rely on the result in any other vertex. The situation is similar for the fragment shader, where each fragment is processed independently. The number of parallel pipelines, or data that can be computed at the

**Table 10.1:** Table describing how the number of pipelines in the GPU has changed. (The latest generation of graphics cards from both the main manufacturers, NVIDIA and ATI, has unified shaders performing all the steps in the process. These are marked with †.)

Model	Vertex pipelines	Fragment pipelines	Released
NVIDIA GeForce 6800 Ultra	6	16	2004
ATI Radeon X850 XT PE	6	16	2005
NVIDIA Geforce 7900 GTX	8	24	2006
NVIDIA Geforce 7950 GX2	16	48	2006
ATI Radeon X1900 XTX	8	48	2006
NVIDIA GeForce 8800 Ultra	128 <sup>†</sup>	128 <sup>†</sup>	2007
ATI Radeon HD 2900 XT	320 <sup>†</sup>	320 <sup>†</sup>	2007

same time, differs between the two programmable units, but in common for both is that the number of pipelines is increasing, as indicated by Table 10.1. The latest generation of graphics cards (late 2007) has hundreds of pipelines, however with a slightly different design as described in this section. (N.B., this section is based on work with a graphics card from the NVIDIA 7000 series.)

Internally the GPU works with quadruples of floating point numbers that represent colors (red, green, blue, and ambient) and data is passed to the GPU as *textures*. Textures are intended to be pictures to be fit onto the surface of the objects rendered on screen.

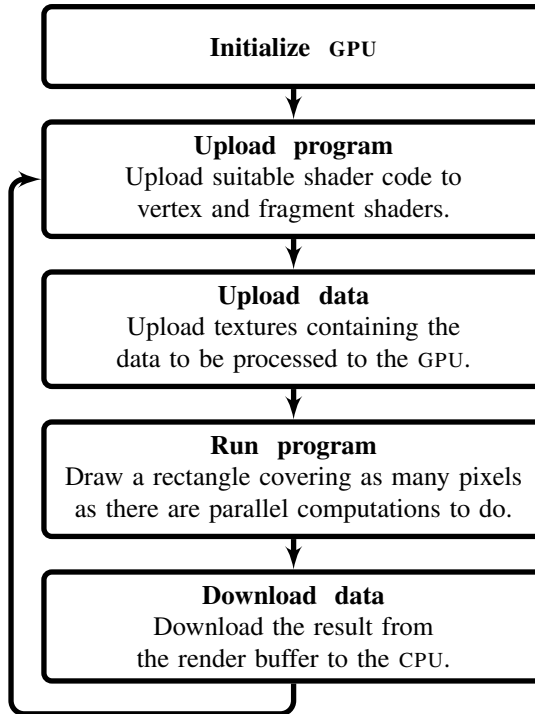
## Programming the GPU

The two steps in the graphics pipeline open to programming are the vertex processor and the fragment processor. Both these processors can be controlled with programs called *shaders*.

Shaders, or GPU programs, were introduced to replace, what used to be, fixed functionality in the graphics pipeline with more flexible programmable processors. They were mainly intended to allow for more advanced graphics effects, but they also got GPGPU started as people could now utilize the GPU for their own purposes. Programming the vertex and fragment processors is in many aspects very similar to programming a CPU, with limitations and extensions made to better support the graphics card and its intended usage, but it should be kept in mind that the code runs in parallel on multiple pipelines of the processor. Furthermore, since the GPU has a SIMD architecture, conditional expressions can be very expensive as the GPU may end up having to run all conditional branches and then throw away all but one result if the same branch is not taken for all data.

Some prominent differences include the basic data types which are available; colors and textures. In newer generations of GPUs 32 bit floating point operations are supported, but the rounding units do not fully conform to the IEEE floating point standard, hence providing somewhat poorer numerical accuracy.

In order to use the GPU for general purpose calculations, a typical GPGPU application applies a program structure similar to Figure 10.4. These very simple steps make sure that the fragment program is executed once for every element of the data, in a separate fragment. The workload is automatically distributed over the available processor pipelines.



**Figure 10.4:** Work flow for GPGPU programming using the OpenGL shading language (GLSL). (The stream processing capability of the latest GPU generation may simplify this somewhat.)

## GPU Programming Language

There are various ways to access the GPU resources as a programmer. Some of the available alternatives are:

- OpenGL [129] which includes the *OpenGL Shading Language* (GLSL), [120]
- *C for graphics* (Cg), [105]
- DirectX High-Level Shader Language (HLSL).

A short description of the alternatives is given below and more information about these and other alternatives can be found in [43, 105, 109]. The development here has been conducted using GLSL.

**OpenGL Shading Language** To run GLSL code on the GPU, the OpenGL *application programming interface* (API) is used [120, 129]. The GLSL code is passed as text to the API where the shaders are compiled and linked into binary code that is then sent to the GPU and executed the next time the graphics card is asked to render a scene. The code

is inspired by C, but changes have been made to suit the GPU. The GLSL is the most low level language of the three alternatives discussed in this thesis.

**C for graphics** This is a high level language developed by NVIDIA in close collaboration with Microsoft. Cg borrows syntax from C but has been modified in order to be better suited for the target GPU. The modifications include new types and vector operations, and encourage coding in terms of data streams. The result of compiling Cg code is shader programs according to the DirectX or the OpenGL standard.

**High Level Shader Language** This is a proprietary shading language designed by Microsoft to inter-operate in a way similar to the OpenGL Shading Language with the Direct3D API from Microsoft. This has much in common with Cg as the two languages were developed at the same time.

## 10.2.2 GPU Based Particle Filter

To implement a parallel particle filter on a GPU there are several aspects of Algorithm 5.1 that require special attention. Resampling is the most challenging step to implement in a parallel fashion since all particles and their weights interact with each other. The main difficulties are cumulative summation, and selection and redistribution of particles. In the following sections, solutions suitable for parallel implementation are proposed for these tasks. Another important issue is how random numbers are generated, since this can prove to take a substantial part of the time spent in the particle filter. The remaining steps, likelihood evaluation as part of the measurement update, and state propagation as part of the time update, are only briefly discussed since they cause no parallelization problem.

### Random Number Generation

At time of writing this thesis, state-of-the-art graphics cards do not have sufficient support for random number generation for usage in a particle filter, since the statistical properties of the built-in generators are too poor. The algorithm in this paper therefore relies on random numbers generated on the CPU to be passed to the GPU. This introduces substantial data transfer as several random numbers per particle are required for one iteration of the particle filter. Uploading data to the graphics card is rather quick, but performance is still lost. Furthermore, this makes generation of random numbers an  $\mathcal{O}(N)$  operation instead of an  $\mathcal{O}(1)$  operation, as would be the case if the generation was completely parallel.

Generating random numbers on the GPU suitable for use in Monte Carlo simulation is an ongoing research topic, see e.g., [30, 134, 137]. Implementing the random number generation in the GPU will not only reduce data transport and allow a standalone GPU implementation, an efficient parallel version will improve overall performance as the random number generation itself takes a considerable amount of time.

### Likelihood Evaluation and State Propagation

Both likelihood evaluation (as part of the measurement update) and state propagation (in the time update) of Algorithm 5.1, can be implemented straightforwardly in a parallel

fashion since all particles are handled independently. As a consequence of this, both operations can be performed in  $\mathcal{O}(1)$  time with  $N$  parallel processors, *i.e.*, one processing element per particle. To solve new filtering problems, only these two functions have to be modified. As no parallelization issues need to be addressed, this is easily accomplished.

In the presented GPU implementation the particles  $x^{(i)}$  and the weights  $\omega^{(i)}$  are stored in separate textures which are updated by the state propagation and the likelihood evaluation, respectively. One texture can only hold four-dimensional state vectors in a natural way, but using multiple rendering targets the state vectors can easily be extended when needed. The idea for that is to store the state in several textures. For instance, with two textures to store the state, the state vector can grow to eight states. With the multi-target capability that modern graphics cards have the changes needed are minimal.

When the measurement noise is low-dimensional the likelihood computations can be replaced with fast texture lookups utilizing the fast texture interpolation. The result is not as exact as if the likelihood was computed the regular way, but the increase in speed is large.

Furthermore, as discussed above, the state propagation uses externally generated process noise, but it would also be possible to generate the random numbers on the GPU.

---

### Example 10.2: Shader program

---

To exemplify GLSL source code, Listing 10.9 contains the code needed for a measurement update in the range-only measurement example,

$$y_t = \left( \begin{array}{c} \|x_t - S_1\|_2 \\ \|x_t - S_2\|_2 \end{array} \right) + e_t,$$

where  $S_1$  and  $S_2$  are sensor locations and  $x_t$  contains the position of the object. (Compare with previous examples.)

#### Listing 10.9: GLSL code used in fragment shader to perform a measurement update

---

```

1  uniform vec2 y;

   uniform sampler2D x;
   uniform sampler2D w;
5
   uniform sampler2D pdf;
   uniform mat2 sqrtSigmainv;

   const vec2 S1 = vec2(1., 0);
10  const vec2 S2 = vec2(-1., 0);

   void main(void)
   {
15  vec2 xtmp = texture2D(x, gl_TexCoord[0].st).xy;
      vec2 e = y-vec2(distance(xtmp, S1), distance(xtmp, S2));
      e = sqrtSigmainv * e + vec2(.5, .5); // Now normalized...
      gl_FragColor.x = texture2D(pdf, e).x
        * texture2D(w, gl_TexCoord[0].st).x;
   }

```

---

The code is very similar to C, and is executed once for each particle, *i.e.*, fragment. To run the program a rectangle is fed as vertices to the graphics card. The size of the

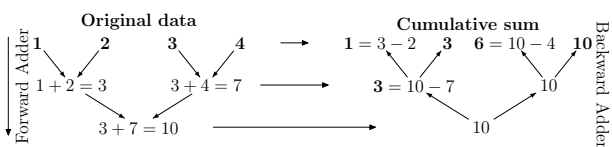
rectangle is chosen such that there will be exactly one fragment per particle, and that way the code is executed once for every particle.

Some comments on the code: The keyword `uniform` indicates that the following variable is set by the API before the program is executed. The variable `y` is hence a two-component vector, `vec2`, with the measurement, and `S1` and `S2` contain the locations of the sensors. Variables of the type `sampler2D` are pointers to specific texture units, and hence `x` and `w` point out particles and weights, respectively, and `pdf` the location of the look up table for the measurement likelihood.

Now the first line of code makes a texture look up and retrieves the state, stored as the two first components of the vector (color data) as indicated by the `xy` suffix. The next line computes the difference between the measurement and the predicted measurement, before the error is scaled and shifted to allow for a quick texture look up. The final line, 19, writes the new weight to the output.

## Summation

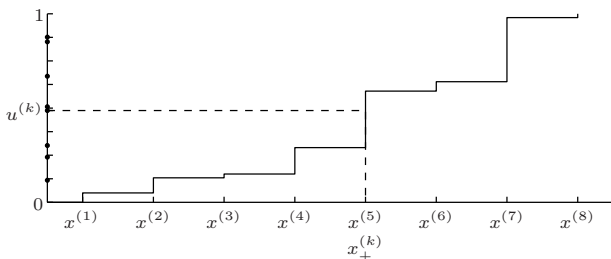
Summations are part of the weight normalization (during measurement updates) and cumulative weight calculation (during resampling) of Algorithm 5.1. A cumulative sum can be implemented using a multi-pass scheme, where an adder tree is run forward and then backward, as illustrated in Figure 10.5. When executing the forward pass the total sum is computed. This multi-pass scheme is a standard method for parallelizing seemingly sequential algorithms based on scatter and gather principles. The reference [109] describes these concepts for the GPU setting. In the forward pass partial sums are created that are used in the backward pass to compute the missing partial sums to complete the cumulative sum. The resulting algorithm is  $\mathcal{O}(\log N)$  in time given  $N$  parallel processors and  $N$  particles.



**Figure 10.5:** Illustration of a parallel implementation of cumulative sum generation of the numbers 1, 2, 3, 4. First the sum, 10, is calculated using a forward adder tree. Then the partial summation results are used by the backward adder to construct the cumulative sum; 1, 3, 6, 10.

## Particle Selection

To prevent sample impoverishment, the resampling step of Algorithm 5.1 replaces the unlikely particles with more likely ones. This is done by drawing a new set of particles  $\{x_+^{(i)}\}$  with replacement from the original particles  $\{x^{(i)}\}$  in such a way that



**Figure 10.6:** Particle selection by comparing uniform random numbers (·) to the cumulative sum of particle weights (–).

$\Pr(x_+^{(i)} = x^{(j)}) = \omega^{(j)}$ . Standard resampling algorithms [33, 62, 79] select new particles using uniformly distributed random numbers as input to the inverse CDF given by the particle weights,

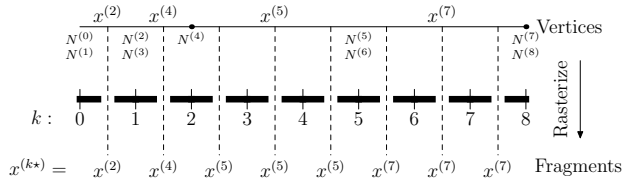
$$x_{i+}^{(k)} = x_i^{(i)}, \quad \text{with } i = P^{-1}(u^{(k)}), \tag{10.1}$$

where  $P$  is the CDF given by the particle weights. For more details about resampling see Section 5.2.2.

The idea for the GPU implementation is to use the rasterizer to do stratified resampling. Stratified resampling is especially suitable for parallel implementation because it produces ordered random numbers, and guarantees that if the interval  $(0, 1]$  is divided into  $N$  interval, there will be exactly one random number in each sub-interval of length  $N^{-1}$ . The selection of what particles to keep is done by drawing a line, consisting of one line segment for each particle in the original set and let the length of the segments indicate how many times the particles should be replicated. With the right segments, the rastering will create evenly spaced fragments from the line, hence giving more fragments from long line segments and consequently more particles from probable particles. For this the properties of the stratified resampling are perfect. They make it possible to compute how many particles have been selected once a certain point in the original distribution was selected. The expression for this is

$$N^{(i)} = \left\lceil Nc^{(i)} - u(\lfloor Nc^{(i)} \rfloor) \right\rceil, \tag{10.2}$$

where  $N$  is the total number of particles,  $c^{(i)} = \sum_{j=1}^i \omega^{(j)}$  is the  $i$ th cumulative weight sum, and  $N^{(i)}$  the number of particles selected when reaching the  $i$ th particle in the original set. The existence of this expression for stratified resampling is vital for the ability to parallelize the resampling step, and hence make a GPU implementation possible. By drawing the line segment for particle  $i$  from  $N^{(i-1)}$  to  $N^{(i)}$ , with  $N^{(0)} = 0$ , the particles that should be selected for the resampled set of particles have a line segment as long as the number of times they will be in the new set. Particles which should not be selected get line segments of zero length. Rastering with unit length between the fragments will therefore produce the correct set of resampled particles, as illustrated in Figure 10.7 for weights in Figure 10.6. The computational complexity of this is  $\mathcal{O}(1)$  with  $N$  parallel processors, as the vertex positions can be calculated independently. Unfortunately, the current generation of GPUs has a maximal texture size limiting the number of particles that can be



**Figure 10.7:** Particle selection on the GPU. The line segments are made up by the points  $N^{(i-1)}$  and  $N^{(i)}$ , which define a line where every segment represents a particle. Some line segments have length 0, i.e., no particle should be selected from them. The rasterizer creates particles  $x$  according to the length of the line segments. The line segments in this figure match those given by the situation in Figure 10.6.

resampled as a single unit. To solve this, multiple subsets of particles are simultaneously being resampled and then redistributed into different sets, similarly to what is described in [23]. This modification of the resampling step does not seem to significantly affect the performance of the particle filter as a whole.

### Complexity Considerations

From the descriptions of the different steps of the particle filter algorithms it is clear that the resampling step is the bottleneck that gives the time complexity of the algorithm,  $\mathcal{O}(\log N)$  compared to  $\mathcal{O}(N)$  for a sequential algorithm.

The analysis of the algorithm complexity above assumes that there are as many parallel processors as there are particles in the particle filter, i.e.,  $N$  parallel elements. Today this is a bit too optimistic, there are hundreds of parallel pipelines in a modern GPU, hence much less than the typical number of particles and the situation is even worse on the GPU used here. However, the number of parallel units is constantly increasing so the degree of parallelization is improving.

Especially the cumulative sum suffers from a low degree of parallelization. With full parallelization the time complexity of the operation is  $\mathcal{O}(\log N)$  whereas a sequential algorithm is  $\mathcal{O}(N)$ , however the parallel implementation uses  $\mathcal{O}(2N)$  operations in total. That is, the parallel implementation uses about twice as many operations as the sequential implementation. This is the price to pay for the parallelization, but is of less interest as the extra operations are shared between many processors. As a result, with few pipelines and many particles the parallel implementation will have the same complexity as the sequential one, roughly  $\mathcal{O}(N \log(N)/M)$  where  $M$  is the number of processors. However, as the degree of parallelization increases the time complexity of the parallel implementation is better than that of a sequential.

### 10.2.3 Evaluation Using Range-Only Measurement Example

The GPU particle filter has been evaluated by applying it to the range only measurement application previously discussed in Section 7.3. To verify the correctness of the implementation a particle filter, using the exact same resampling scheme, has been designed for the CPU. The resulting filters give practically identical results, though minor differences

**Table 10.2:** Hardware used for the evaluation.

GPU	
Model:	NVIDIA GeForce 7900 GTX
Driver:	2.1.0 NVIDIA 96.40
Bus:	PCI Express, 14.4 GB/s
Clock speed:	650 MHz
Processors:	8/24 (vertex/fragment)
CPU	
Model:	Intel Xeon 5130
Clock speed:	2.0 GHz
Memory:	2.0 GB
Operating System:	CentOS 4.5 (Linux)

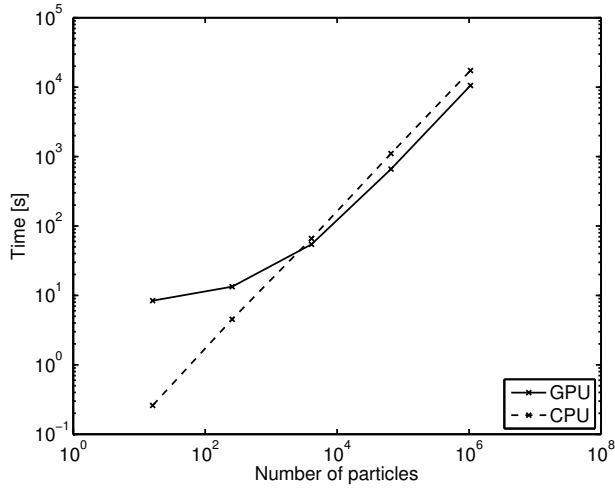
exist due to the less sophisticated rounding unit available at the GPU. Furthermore, the performance of the filters is comparable to what has been achieved previously for this problem.

To evaluate the complexity gain obtained from using the parallel GPU implementation the GPU and the CPU implementations of the particle filter were run and timed. Information about the hardware used for this is gathered in Table 10.2. Figure 10.8 gives the total time for running the filter for 100 time steps 100 times for a set of different number of particles ranging from  $2^4 = 16$  to  $2^{20} \approx 10^6$ . In no way 16 particles are enough for this problem, nor is as many as  $10^6$  needed, however the large range helps showing the complexity trends better.

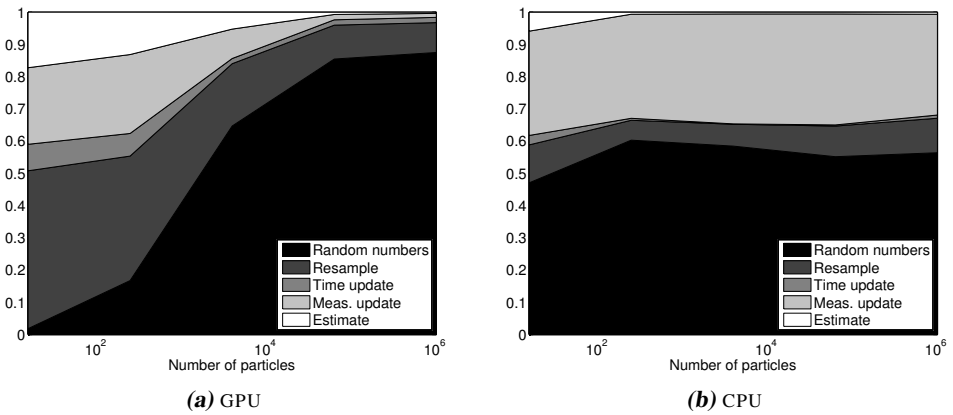
Some observations should be made: for few particles the overhead from initializing and using the GPU is large and hence the CPU implementation is the fastest. The CPU complexity follows a linear trend, whereas at first the GPU time hardly increases when using more particles; parallelization pays off. For even more particles there are not enough parallel processing units available and the complexity becomes linear, but the GPU implementation is still faster than the CPU. Note that the particle selection is performed on 8 processors and the other steps on 24, see Table 10.2, and hence that the degree of parallelization is not very high with many particles.

A further analysis of the time spent in the GPU implementation shows in what part of the algorithm most of the time is spent, see Figure 10.9. The main cost in the GPU implementation quickly becomes the random number generation (performed on the CPU), which shows that if that step can be parallelized there is much to gain in performance. For both CPU and GPU the time update step is almost negligible, which is an effect of the simple dynamic model. The GPU would have gained from a computationally expensive time update step, where the parallelization would have paid off better. To produce an estimate from the GPU is relatively more expensive than it is with the CPU. For the CPU all steps are  $\mathcal{O}(N)$  whereas for the GPU the estimate is  $\mathcal{O}(\log N)$  where both the measurement update and the time update steps are  $\mathcal{O}(1)$ . Not counting the random number generation, the major part of the time is spent doing resampling in the GPU, whereas the measurement update is a much more prominent step in the CPU implementation. One reason for this is probably using the hardware implemented texture lookups for the measurement likelihood in the GPU.

The conclusion from the evaluation is that it is actually possible to implement a parti-



**Figure 10.8:** Comparison of used time for GPU and CPU.



**Figure 10.9:** Comparison of the relative time used for the different steps in the GPU and CPU implementations of the particle filter.

cle filter in parallel, and that doing so could actually improve computational performance and allow for new filtering problems to be solved.

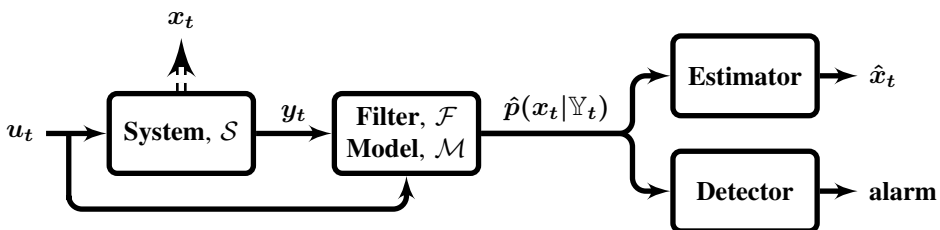
# 11

## Concluding Remarks

**T**HE INTRODUCTION TO this thesis discusses the need for a framework for nonlinear signal processing. The driving force is to simplify the process of effectively using advanced signal processing methods in new applications. Today, using something other than the most basic standard tools demands a skilled and experienced developer to choose algorithm and to fine-tune it. A framework with design guidelines would speed up the design phase and make the techniques more widely available.

Ideally such a signal processing framework should take all major aspects of the signal processing problem into consideration. Figure 11.1 illustrates the main components and how they relate to each other:

- System — Good understanding of the system is important in order to be able to deliver a design that fulfills given requirements and also to be able to design a good signal processing solution.
- Filter — Filter design is a combination of selecting a suitable algorithm for the problem, and to use an appropriate model of the system suitable for the result needed.



**Figure 11.1:** Illustration of the major components of a signal processing framework and how they relate to each other. This figure previously appeared as Figure 1.2.

- Estimator — The final estimation step converts the posterior distribution  $p(x_t|\mathbb{Y}_t)$  from the filter to a point estimate. A method to evaluate the result is almost as important as the estimator itself.
- Detector — An alternative to extracting a point estimate is to detect a certain condition. As with the point estimate it is important to be able to evaluate the result.

If it is possible to provide decision support for one or more of these components it would greatly simplify designing new signal processing applications. The ultimate goal is to be able to automatically provide a model, an algorithm, and reasonable tuning parameters, based on an accurate system description, such that a sufficient posterior distribution and an estimator/detector can be obtained.

Formulated more in detail, given the true system,  $\mathcal{S}$ , containing the complete system description, how should the model,  $\mathcal{M}$ , and the filtering algorithm  $\mathcal{F}$ , be chosen, such that the posterior distribution  $\hat{p}(x_t|\mathbb{Y}_t)$  becomes as close to the truth  $p(x_t|\mathbb{Y}_t)$  as possible? And how should then this result be used to produce a point estimate or alarm?

Assuming a general state-space formulation of the system

$$\begin{aligned}x_{t+1} &= f(x_t, u_t, w_t) \\ y_t &= h(x_t, u_t, e_t),\end{aligned}$$

where  $w_t \sim p_w$  and  $e_t \sim p_e$ , the task is to select approximations of  $f$ ,  $h$ ,  $p_w$ , and  $p_e$  for  $\mathcal{M}$  as close as possible to  $\mathcal{S}$ . The approximation should be good enough to fulfill all given design goals and at the same time as simple as possible to make realizing the chosen filter easy. Therefore, the model should be chosen in parallel with the filtering algorithm,  $\mathcal{F}$ , as different algorithms require different types of models and degrees of accuracy. The objective is to approximate the true posterior distribution  $p(x_t|\mathbb{Y}_t)$  with  $\hat{p}(x_t|\mathbb{Y}_t)$ . A nontrivial question in this context is what measure should be used to compare the posterior distributions, especially if  $\mathcal{S}$  and  $\mathcal{M}$  use different state-space representations.

Another important detail needed to make new more advanced methods more available is efficient implementations. Fast and well written implementations make it possible to use the methods in time critical applications. Evaluation and tuning is also simplified when the time it takes to evaluate a scenario is shortened.

## 11.1 Results

This thesis addresses several of the issues discussed above. The contributions are both theoretical, with analyses of system and algorithm properties, and more practical, dealing with implementation issues.

The *Cramér-Rao lower bound* (CRLB) analysis in Chapter 6 is useful in the early phase of the design process. By computing the CRLB for a system it is possible to determine if it is possible to fulfill the design requirements. It is shown that by considering the *intrinsic accuracy* (IA, Chapter 2) of the involved noise it is possible to get more results. For linear systems it is possible to indicate if computationally expensive nonlinear filtering techniques will pay off compared to linear methods and to indicate how well the system noise should be modeled. The results are verified and exemplified in extensive simulation studies in Chapter 7.

Detection performance is analyzed in Chapter 8 using methods closely related to the CRLB analysis in Chapter 6. Derived asymptotic results, in terms of the intrinsic accuracy for the involved noise, indicate if it is worth the extra effort to carefully model the involved noise. Explicit results are derived for residuals from linear systems. Simulation studies in Chapter 9 verify the theoretical analysis.

For nonlinear systems there are several alternative algorithms that can be used. Many of the algorithms are a combination of the *Kalman filter* and a model approximation, e.g., the *extended Kalman filter* and the *unscented Kalman filter* both discussed in Chapter 5. Depending on which approximation is used, the results differ and some methods are better than others. The analysis in Chapter 3 of the underlying approximations makes it easier to apply the appropriate algorithm in a situation. Special attention is given the *unscented transform* (UT), as it is a relatively new and widely used method, and it is compared to the *Gauss approximation* of first and second order.

In Chapter 5 an analysis of the *Rao-Blackwellized particle filter* (RBPF, denoted the *marginalized particle filter* by some authors) leads up to a new interpretation of the method, giving new intuition for the method. The analysis also leads up to a different formulation of the algorithm that is more suitable for implementation than other formulations found in the literature.

Simulation studies in Chapters 7 and 9 demonstrates the importance of obtaining a correct posterior distribution  $p(x_t|\mathbb{Y})$ . It is also shown that the standard performance measure, the *root mean square error* (RMSE), is sometimes a misleading performance measure for filtering applications as it takes too little of the distribution into consideration. Therefore, the *Kullback divergence* introduced in Chapter 2 is introduced as an alternative to CRLB, and it is in simulation studies shown, in Chapter 7, that it captures important aspects that are lost by the RMSE.

Chapter 10 addresses implementation aspects building a C++ filtering framework and a parallel particle filter. The filtering framework is an alternative to MATLAB<sup>®</sup> which utilizes the power of a compiled language such C++ at the same time as it provides an interface that should be easy to learn for engineers. The parallel particle filter was the first complete implementation of a parallel particle filter on a *graphics processing unit* (GPU) reported. The techniques used there open up for implementing other filters on this inexpensive hardware.

## 11.2 Future Work

Many questions remain to be answered before it is possible to automatically design a filter based solely on a system description. However, the contributions in this thesis take steps in the right direction. For the future, two questions stand out as suitable and important next steps: better methods to evaluate the results from the filtering step and a more thorough evaluation of the interaction between the choice of model approximation and filtering algorithm.

The thesis starts to explore alternative ways of comparing the estimated posterior distribution  $\hat{p}(x_t|\mathbb{Y}_t)$  with the true distribution  $p(x_t|\mathbb{Y}_t)$ , and the Kullback divergence is suggested as a measure. However, important questions are still unanswered in this context. One important being what posterior distributions to compare with. Assume  $\mathcal{S}$  and  $\mathcal{M}$  have

different state-space representations, for example as the result of a model approximation, what should be compared then? One option to explore is to use the posterior measurement distribution  $p(y_{t+1}|\mathbb{Y}_t)$  instead. That distribution is completely state independent. Another option is to use a common subset,  $x^r$ , of the state-space and compare  $p(x_t^r|\mathbb{Y}_t)$ . If the rest of the state space description differs,  $x^r$  should contain all information of interest. Remaining, uninteresting, states,  $x^a$ , would then have to be marginalized away, *i.e.*,

$$p(x_t^r|\mathbb{Y}_t) = \int_{x_t^a} p(x_t^r, x_t^a|\mathbb{Y}_t) dx_t^a.$$

Another question not treated in depth in the thesis is how the model  $\mathcal{M}$  should be chosen and how this choice relates to the choice of filtering algorithm. At a conceptual level, the error in the estimated posterior could be separated into two parts,

$$\|p - \hat{p}\| = \|\mathcal{S} - \mathcal{M}\| + \|\mathcal{F} - \mathcal{T}\|,$$

where  $\mathcal{T}$  is used to denote ideal Bayesian filtering solution. That is, the first term covers errors in the posterior distribution that are due to approximations of the system in the model, and the second term is due to the approximation of the Bayesian solution made in the filtering algorithm. By choosing a simple enough model the filtering algorithm can be made exact, but then the model error will be large, whereas a perfect model probably results in larger errors due to an inexact algorithm. The trade-off between this and also the possibility to trade bias for a more consistent estimate deserves further attention.



---

## Notational Conventions

### Abbreviations and Acronyms

Abbreviation	Meaning
AFMM	Adaptive forgetting through multiple models
API	Application programming interface
BLUE	Best linear unbiased estimate/estimator
CDF	Cumulative distribution function
CPU	Central processing unit
CRLB	Cramér-Rao lower bound
CV	Constant velocity
EKF	Extended Kalman filter
FI	Fisher information
FPGA	Field-programmable gate array
GLR	Generalized likelihood ratio
GLSL	OpenGL shading language
GPB	Generalized pseudo-Bayesian
GPGPU	General-purpose computation on GPU
GPU	Graphics processing unit
HMM	Hidden Markov model
IA	Intrinsic accuracy
IEKF	Iterated extended Kalman filter
IID	Identically and independently distributed
IMM	Interacting multiple models
KFB	Kalman filter bank
KF	Kalman filter
LSE	Least square estimate/estimator
MAP	Maximum <i>a priori</i>
MC	Monte Carlo

---

Abbreviation	Meaning
MLE	Maximum likelihood estimate/estimator
MMSE	Minimum mean squares error
MSE	Mean square error
MVE	Minimum variance estimate/estimator
PDF	Probability density function
PF	Particle filter
RA	Relative accuracy
RBPF	Rao-Blackwellized particle filter
RMSE	Root mean square error
ROC	Receiver operating characteristics
SIR	Sampling importance resampling
SIS	Sequential importance sampling
SNR	Signal-to-noise ratio
SVD	Singular value decomposition
TTI	First order Gauss approximation
TTI	Second order Gauss approximation
UKF	Unscented Kalman filter
UMP	Uniformly most powerful
UT	Unscented transform

## Symbols and Mathematical Notation

Notation	Meaning
$\mathbb{X}_t$	Stacked variables, $\mathbb{X}_t^T = (x_1^T, \dots, x_t^T)$ .
$\mathbb{X}_t^L$	Stacked variables in a window of length $L$ , $\mathbb{X}_t^T = (x_{t-L+1}^T, \dots, x_t^T)$ .
$x \sim y$	$x$ is distributed as $y$ .
$x \stackrel{a}{\sim} y$	$x$ is asymptotically distributed as $y$ .
$\nabla_x$	Gradient with respect to $x$ , see (A.1) below.
$\Delta_x^y$	Second derivative, $\nabla_x \nabla_y$ , see (A.2) below.
$L(\cdot) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma$	If $L(\cdot) \geq \gamma$ , reject $\mathcal{H}_0$ otherwise $\mathcal{H}_0$ is accepted.
$\Psi_x$	Relative accuracy of $x$ .
$\arg \max_x f(x)$	The $x$ maximizing $f(x)$ .
$\arg \min_x f(x)$	The $x$ minimizing $f(x)$ .
$\text{cov}(x)$	Covariance of $x$ .
$\delta(\cdot)$	Generalized Dirac function.
$\delta_t$	Mode indicator for time $t$ .
$\text{diag}(x_1, \dots, x_n)$	A diagonal matrix with $x_i$ in the diagonal.
$e_t$	Measurement noise at time $t$ .
$E(x)$	Expected value of $x$ .
$f(\cdot)$	State propagation function.

Notation	Meaning
$f_t$	Deterministic but unknown fault at time $t$ .
$\mathcal{H}_i$	A hypothesis.
$h(\cdot)$	Measurement function.
$\mathcal{I}_x(\mu)$	Fisher information of $x$ with respect to $\mu$ .
$\mathcal{I}_x$	Intrinsic accuracy of $x$
$\mathcal{I}^{\text{KL}}(p, q)$	Kullback-Leibler information between the PDFs $p(\cdot)$ and $q(\cdot)$ .
$\mathcal{J}^{\text{K}}(p, q)$	Kullback divergence between the PDFs $p$ and $q$ .
$L(\cdot)$	Likelihood ratio.
$L$	Window size.
$\mu_x$	Mean of $x$ .
$\mathcal{N}(x; \mu, \Sigma)$	Gaussian PDF for mean $\mu$ and covariance $\Sigma$ .
$\mathcal{N}_n(x; (\omega_\delta, \mu_\delta, \Sigma_\delta)_{\delta=1}^n)$	Gaussian sum PDF with $n$ modes.
$\mathcal{N}^\nu(x; \sigma)$	Generalized Gaussian PDF with variance $\sigma^2$ .
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean $\mu$ and covariance $\Sigma$ .
$\mathcal{N}_n((\omega_\delta, \mu_\delta, \Sigma_\delta)_{\delta=1}^n)$	Gaussian sum distribution with $n$ modes.
$\mathcal{N}^\nu(\sigma)$	Generalized Gaussian distribution with variance $\sigma^2$ .
$n_x$	Dimension of the variable $x$ .
$\omega_\delta$	Mode/particle probability.
$P_{t \tau}$	Estimation error covariance at time $t$ given the measurements $\mathbb{Y}_\tau$ .
$p(\cdot)$	Probability density function.
$\text{Pr}(\mathcal{A})$	Probability of the statement $\mathcal{A}$ .
$Q_t$	Covariance of process noise at time $t$ .
$\mathcal{Q}_\star$	Complementary cumulative distribution function of the distribution $\star$ .
$R_t$	Covariance of measurement noise at time $t$ .
$r_t$	Residual
$\text{rank}(A)$	Rank of $A$ .
$\Sigma_x$	Covariance of $x$ .
$\text{tr}(A)$	Trace of $A$ .
$A^T$	$A$ transposed.
$u_t$	Known input at time $t$ .
$\text{var}(x)$	Variance of $x$ .
$w_t$	Process noise at time $t$ .
$x_t$	State at time $t$ .
$\hat{x}_{t \tau}$	Estimate of $x_t$ given the measurements $\mathbb{Y}_\tau$ .
$y_t$	Measurement at time $t$ .

## Definition of Derivatives

The derivative of  $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ , often denoted the *gradient* or the *Jacobian*, used is

$$\nabla_x f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}. \quad (\text{A.1})$$

With this definition the second derivative of a scalar function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  becomes

$$\Delta_x^y f = \nabla_x \nabla_y f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial y_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial y_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial y_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial y_m} \end{pmatrix}. \quad (\text{A.2})$$

# B

---

## F++ Listings

### Range-Only Measurement — Simulation Loop

*Listing B.1: Range-only measurement simulation code: rangeonly.cc*

---

```
1  #include "noise/noise.h"
   #include "romodel.h"
   #include "estimator/pf.h"
   #include "estimator/ekf.h"
5  #include "io/xmlfile.h"
   #include "io/matfile.h"
   #include "misc/data.h"

   #include <iostream>
10

   Data<double> rmse(const Data<Data<Model::vector_t>>& truth ,
                   const Data<Data<Model::vector_t>>& est);

15  using namespace std;

   int main()
   {

20  // *** INITIALIZE SIMULATION SETUP ***
     XMLfile is("rangeonly.xml", XMLfile::read);

     // Initialize model
     Noise x0(is, "x0");
25  Noise W(is, "W");
     Noise E(is, "E");
     long tfin = readScalar<long>(is, "tfin");
     long NMC = readScalar<long>(is, "NMC");

30  Model::vector_type S1(readVector<double>(is, "S1"));
```

```

Model::vector_type S2(readVector<double>(is, "S2"));

Model model = createRangeOnlyModel(S1, S2, W, E);

35 // Create truth for the simulations
Data<Data<Model::vector_t>> Xtrue;
Data<Data<Model::vector_t>> Y;
for (long i=0; i<NMC; ++i) {
    Xtrue.push_back(generate_trajectory(model, x0.random(), tfin));
40    Y.push_back(generate_measurement(model, Xtrue.back()));
}

// Set up place to save result
Data<Data<Model::vector_t>> Xhatpf(NMC);
45 Data<Data<Model::vector_t>> Xhatekf(NMC);
typedef Data<Model::vector_t>::const_iterator cItr;

// *** DO MONTE CARLO SIMULATIONS ***
for (long i=0; i<NMC; ++i) {
50    Estimator pf = createPF(model, 10000, x0, "ParticleFilter");
    Estimator ekf = createEKF(model, x0, "ExtendedKalmanFilter");

    for (cItr y = Y[i].begin(); y!=Y[i].end(); ++y) {
        // PF filtering
55        pf.mUpdate(*y);
        Xhatpf[i].push_back(pf.xhat());
        pf.tUpdate();

        // EKF filtering
60        ekf.mUpdate(*y);
        Xhatekf[i].push_back(ekf.xhat());
        ekf.tUpdate();
    }
}
65

// *** POSTPROCESS SIMULATION ***
Data<double> RMSEpf = rmse(Xtrue, Xhatpf);
Data<double> RMSEekf = rmse(Xtrue, Xhatekf);

70 MATfile os("rangeonly.out", MATfile::write);
write(os, "RMSEpf", RMSEpf);
write(os, "RMSEekf", RMSEekf);
return 0;
}
75

Data<double> rmse(const Data<Data<Model::vector_t>>& truth,
                const Data<Data<Model::vector_t>>& est)
{
80    long NMC = est.size();
    long tfin = est.front().size();

    typedef Model::vector_type vector_type;
    Data<double> RMSE(tfin);
85

    for (long i=0; i<NMC; ++i) {
        for (long j=0; j<tfin; ++j) {
            vector_type diff = truth[i][j].datum-est[i][j].datum;

```

---

```

    RMSE[j] += diff.inner_product(diff);
90   }
    }

    for (long i=0; i<tfin; ++i)
    RMSE[i] = std::sqrt(RMSE[i]/NMC);
95

    return RMSE;
}

```

---

## Range-Only Measurement — Input

*Listing B.2: Example input for rangeonly.cc: rangeonly.xml*

---

```

1 <vector name='S1' dim='2'>
  1 0
</vector>

5 <vector name='S2' dim='2'>
  -1 0
</vector>

<object name='x0' class='Noise::GaussNoise'>
10 <matrix name='Sigma' rows='2' cols='2'>
  3 0
  0 3
</matrix>
  <vector name='mu' dim='2'>
15   0 0
  </vector>
</object>

<object name='W' class='Noise::GaussNoise'>
20 <matrix name='Sigma' rows='2' cols='2'>
  0.1 0
  0 0.1
</matrix>
  <vector name='mu' dim='2'>
25   0 0
  </vector>
</object>

<object name='E' class='Noise::GaussNoise'>
30 <matrix name='Sigma' rows='2' cols='2'>
  0.1 0
  0 0.1
</matrix>
  <vector name='mu' dim='2'>
35   0 0
  </vector>
</object>

<scalar name='tfin'>
40   100
</scalar>

```

```

<scalar name='NMC'>
  100
45 </scalar>

```

---

## Range-Only Measurement — Model

*Listing B.3: Range-only measurement model code: romodel.h*

```

1 #ifndef __ROMODEL_H__
# define __ROMODEL_H__

#include "model/model.h"

5
class RangeOnlyModel : public ModelImpl
{
public:
  RangeOnlyModel(const vector_type& S1, const vector_type& S2,
10                 Noise W, Noise E,
                 const std::string& name = "RangeOnlyModel");
  ~RangeOnlyModel()
  {}
  RangeOnlyModel* clone() const;

15
  size_t nx(const vector_t& u) const;
  size_t nu(const vector_t& u) const;
  size_t ny(const vector_t& u) const;

20
  vector_t f(const vector_t& x,
             const vector_t& w,
             const vector_t& u) const;
  vector_t h(const vector_t& x,
             const vector_t& e,
25             const vector_t& u) const;

  matrix_t f_x(const vector_t& x,
              const vector_t& w,
              const vector_t& u) const;
30
  matrix_t f_w(const vector_t& x,
              const vector_t& w,
              const vector_t& u) const;

  matrix_t h_x(const vector_t& x,
              const vector_t& e,
              const vector_t& u) const;
35
  matrix_t h_e(const vector_t& x,
              const vector_t& e,
              const vector_t& u) const;

40
  Noise w(const vector_t& u) const;
  Noise e(const vector_t& u) const;

  FileStruct* writeFileStruct() const
45  { throw; }

```



```

    return vector_t(x.datum + w.datum, x.time+1);
}

45 RangeOnlyModel::vector_t RangeOnlyModel::h(const vector_t& x,
                                             const vector_t& e,
                                             const vector_t& u) const
{
50   vector_type y(2);
   y[0] = (S1_-x.datum).length();
   y[1] = (S2_-x.datum).length();
   return vector_t(y, x.time);
}

55

RangeOnlyModel::matrix_t RangeOnlyModel::f_x(const vector_t& x,
                                             const vector_t& w,
                                             const vector_t& u) const
60 {
   return matrix_t(I2_, x.time);
}

65 RangeOnlyModel::matrix_t RangeOnlyModel::f_w(const vector_t& x,
                                             const vector_t& w,
                                             const vector_t& u) const
{
70   return matrix_t(I2_, x.time);
}

RangeOnlyModel::matrix_t RangeOnlyModel::h_x(const vector_t& x,
                                             const vector_t& e,
                                             const vector_t& u) const
75 {
   vector_type d1 = x.datum-S1_;
   vector_type d2 = x.datum-S2_;
   double r1 = d1.length();
80   double r2 = d2.length();
   matrix_type grad(2,2);
   grad(0, 0) = d1[0]/r1;   grad(0, 1) = d1[1]/r1;
   grad(1, 0) = d2[0]/r2;   grad(1, 1) = d2[1]/r2;
   return matrix_t(grad, x.time);
85 }

RangeOnlyModel::matrix_t RangeOnlyModel::h_e(const vector_t& x,
                                             const vector_t& e,
                                             const vector_t& u) const
90 {
   return matrix_t(I2_, x.time);
}

95 Noise RangeOnlyModel::w(const vector_t& u) const
{
   return W_;
}

```

---

```
100 Noise RangeOnlyModel::e(const vector_t& u) const
    {
105   return E_;
    }

Model createRangeOnlyModel(const Model::vector_type& S1,
                           const Model::vector_type& S2,
110                           Noise W, Noise E,
                           const std::string& name)
    {
    return Model(new RangeOnlyModel(S1, S2, W, E, name));
    }
```

---



---

## Bibliography

- [1] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions: with formulas, graphs, and mathematical tables*. Dover Publications, New York, 1965.
- [2] Bruno Aiazzi, Luciano Alparone, and Stefano Bartoni. Estimation based on entropy matching for generalized Gaussian PDF modeling. *IEEE Signal Processing Letters*, 6(6):138–140, June 1999.
- [3] Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian estimation using Gaussian sum. *IEEE Transactions on Automatic Control*, 17(4):439–448, 1972.
- [4] Brian D. O. Anderson and John B. Moore. *Optimal Filtering*. Prentice-Hall, Inc, Englewood Cliffs, NJ, 1979. ISBN 0-13-638122-7.
- [5] Peter Andersson. Adaptive forgetting in recursive identification through multiple models. *International Journal of Control*, 42(5):1175–1193, 1985.
- [6] Christophe Andrieu and Arnaud Doucet. Particle filter for partially observed Gaussian state space models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 64(4):827–836, 2002.
- [7] Christoph Arndt. *Information Measures*. Springer-Verlag, 2001. ISBN 3-540-40855-X.
- [8] Michael Athans, Richard P. Wishner, and Anthony Bertolini. Suboptimal state estimation for continuous-time nonlinear systems from discrete noisy measurements. *IEEE Transactions on Automatic Control*, 13(5):504514, October 1968.
- [9] Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and Data Association*. Academic Press, Inc, 1988. ISBN 0-12-079760.

- [10] Ole E. Barndorff-Nielsen. Normal inverse Gaussian distributions and stochastic volatility modelling. *Scandinavian Journal of Statistics*, 27:1–13, March 1997.
- [11] Michèle Basseville and Igor V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, Inc, 1993. ISBN 0-13-126780-9.
- [12] Stella N. Batalama and Demetrios Kazakos. On the generalized Cramér-Rao bound for the estimation of the location. *IEEE Transactions on Signal Processing*, 45(2): 487–492, February 1997.
- [13] Thomas Bayes. An essay towards solving a problem in the doctrine of chances. *The Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763. Reproduced in [17].
- [14] Niclas Bergman. *Bayesian Inference in Terrain Navigation*. Licentiate thesis no 649, Department of Electrical Engineering, Linköpings universitet, Sweden, 1997.
- [15] Niclas Bergman. *Recursive Bayesian Estimation: Navigation and Tracking Applications*. Dissertations no 579, Linköping Studies in Science and Technology, SE-581 83 Linköping, Sweden, May 1999.
- [16] Niclas Bergman, Lennart Ljung, and Fredrik Gustafsson. Terrain navigation using Bayesian statistics. *IEEE Control Systems Magazine*, 19(3):33–40, June 1999.
- [17] George A. Bernard and Thomas Bayes. Studies in the history of probability and statistics: IX. Thomas Bayes’s essay towards solving a problem in the doctrine of chances. *Biometrika*, 45(3/4):293–315, December 1958.
- [18] Samuel S. Blackman and Robert Popoli. *Design and analysis of modern tracking systems*. Artech House radar library. Artech House, Inc, 1999. ISBN 1-5853-006-0.
- [19] David Blackwell. Conditional expectation and unbiased sequential estimation. *The Annals of Mathematical Statistics*, 18(1):105–110, March 1947.
- [20] Henk A. P. Blom and Yaakov Bar-Shalom. The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33(8):780–783, August 1988.
- [21] Ben Zion Bobrovsky and Moshe Zakai. A lower bound on the estimation error for Markov processes. *IEEE Transactions on Automatic Control*, 20(6):785–788, December 1975.
- [22] Milodrag Bolić, Petar Djurić, and Sangjin Hong. Resampling algorithms for particle filters: A computational complexity perspective. *EURASIP Journal on Applied Signal Processing*, 15:2267–2277, 2004.
- [23] Milodrag Bolić, Petar M. Djurić, and Sangjin Hong. Resampling algorithms and architectures for distributed particle filters. *IEEE Transactions on Signal Processing*, 53(7):2442–2450, July 2005.

- [24] George E. P. Box and George C. Tiao. *Bayesian Inference in Statistical Analysis*. Addison-Wesley, 1973.
- [25] Richard S. Bucy and Kenneth D. Senne. Digital synthesis of non-linear filters. *Automatica*, 7(3):287–298, 1971.
- [26] Rong Chen and Jun S. Liu. Mixture Kalman filters. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 62(3):493–508, 2000.
- [27] David R. Cox and David V. Hinkley. *Theoretical Statistics*. Chapman and Hall, New York, 1974.
- [28] Charlotte Dahlgren. Nonlinear black box modelling of JAS 39 Gripen’s radar altimeter. Master’s thesis no LiTH-ISY-EX-1958, Department of Electrical Engineering, Linköpings universitet, Sweden, October 1998.
- [29] Fred Daum and Jim Huang. Curse of dimensionality and particle filters. In *Proceedings of IEEE Aerospace Conference*, volume 4, pages 1979–1993, Big Sky, MT, USA, March 2003. IEEE.
- [30] A. De Matteis and S. Pagnutti. Parallelization of random number generators and long-range correlation. *Numerische Mathematik*, 53(5):595–608, 1988.
- [31] Peter C. Doerschuck. Cramer-Rao bounds for discrete-time nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 40(8):1465–1469, August 1995.
- [32] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10:197–208, 2000.
- [33] Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. Springer-Verlag, New York, 2001. ISBN 0-387-95146-6.
- [34] Ronald A. Fisher. On the foundations of mathematical statistics. *The Philosophical Transactions of the Royal Society of London*, A(222):309–368, 1922.
- [35] Ronald A. Fisher. Theory of statistical estimation. In *Proceedings of the Cambridge Philosophical Society*, volume 22, pages 700–725, 1925.
- [36] Philippe Forster and Pascal Larzabal. On the lower bounds for deterministic parameter estimation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1137–1140, Orlando, FL, USA, May 2002.
- [37] Jorge I. Galdos. A Cramér-Rao bound for multidimensional discrete-time dynamical systems. *IEEE Transactions on Automatic Control*, 25(1):117–119, February 1980.

- [38] Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. ISBN 0-201-63361-2.
- [39] Janos J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, Inc, 1998. ISBN 0-8247-9427-3.
- [40] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. John Hopkins University Press, 3 edition, 1996. ISBN 0-2018-54-14-8.
- [41] I. John Good. Significance test in parallel and in series. *Journal of the American Statistical Association*, 53(284):799–813, December 1958.
- [42] Neil J. Gordon, David J. Salmond, and Adrian F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F Radar and Signal Processing*, 140(2):107–113, April 1993.
- [43] GPGPU. GPGPU programming web-site. URL: <http://www.gpgpu.org>, 2006.
- [44] Fredrik Gustafsson. *Adaptive Filtering and Change Detection*. John Wiley & Sons, Ltd, Chichester, West Sussex, England, 2000. ISBN 0-471-49287-6.
- [45] Fredrik Gustafsson. *Adaptive Filtering and Change Detection*. John Wiley & Sons, Ltd, 2000. ISBN 0-471-49287-6. 2 reprint.
- [46] Fredrik Gustafsson. Stochastic fault diagnosability in parity spaces. In *Proceedings of 15th Triennial IFAC World Congress on Automatic Control*, Barcelona, Spain, July 2002.
- [47] Fredrik Gustafsson and Gustaf Hendeby. On nonlinear transformations of stochastic variables and its application to nonlinear filtering. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Las Vegas, NV, USA, March 2008. Accepted for publication.
- [48] Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and Per-Johan Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, February 2002.
- [49] Allan Gut. *An Intermediate Course in Probability*. Springer-Verlag, 1995. ISBN 0-387-94507-5.
- [50] John M. Hammersley and K. William Morton. Poor man’s Monte Carlo. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 16(1):23–38, 1954.
- [51] Alfred Hanssen and Tor Arne Øigård. The normal inverse Gaussian distribution: A versatile model for heavy-tailed stochastic processes. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3985–3988, Salt Lake City, UT, USA, May 2001.

- [52] Gustaf Hendeby. *Fundamental Estimation and Detection Limits in Linear Non-Gaussian Systems*. Licentiate thesis no 1199, Department of Electrical Engineering, Linköpings universitet, Sweden, November 2005.
- [53] Gustaf Hendeby and Fredrik Gustafsson. Fundamental filtering limitations in linear non-Gaussian systems. In *Proceedings of 16th Triennial IFAC World Congress*, Prague, Czech Republic, July 2005.
- [54] Gustaf Hendeby and Fredrik Gustafsson. Fundamental fault detection limitations in linear non-Gaussian systems. In *Proceedings of 44th IEEE Conference on Decision and Control and European Control Conference*, Sevilla, Spain, December 2005.
- [55] Gustaf Hendeby and Fredrik Gustafsson. Detection limits for linear non-Gaussian state-space models. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Beijing, P. R. China, August–September 2006.
- [56] Gustaf Hendeby and Rickard Karlsson. Target tracking performance evaluation — a general software environment for filtering. In *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, March 2007.
- [57] Gustaf Hendeby, Rickard Karlsson, Fredrik Gustafsson, and Neil Gordon. Recursive triangulation using bearings-only sensors. In *Proceedings of The IEE Seminar on Target Tracking: Algorithms and Applications*, Birmingham, UK, March 2006.
- [58] Gustaf Hendeby, Rickard Karlsson, Fredrik Gustafsson, and Neil Gordon. Performance issues in non-Gaussian filtering problems. In *Proceedings of Nonlinear Statistical Signal Processing Workshop*, Cambridge, UK, September 2006.
- [59] Gustaf Hendeby, Jeroen D. Hol, Rickard Karlsson, and Fredrik Gustafsson. A graphics processing unit implementation of the particle filter. In *Proceedings of European Signal Processing Conference*, Poznań, Poland, September 2007.
- [60] Gustaf Hendeby, Rickard Karlsson, and Fredrik Gustafsson. A new formulation of the Rao-Blackwellized particle filter. In *Proceedings of IEEE Workshop on Statistical Signal Processing*, Madison, WI, USA, August 2007.
- [61] Jeroen D. Hol. Resampling in particle filters. Student thesis no. LiTH-ISY-EX-ET-0283-2004, Department of Electrical Engineering, Linköpings universitet, Sweden, May 2004.
- [62] Jeroen D. Hol, Thomas B. Schön, and Fredrik Gustafsson. On resampling algorithms for particle filters. In *Proceedings of Nonlinear Statistical Signal Processing Workshop*, Cambridge, UK, September 2006.
- [63] Peter J. Huber. *Robust Statistics*. John Wiley & Sons, Ltd, 1981. ISBN 0-471-41805-6.
- [64] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*, volume 64 of *Mathematics in Science and Engineering*. Academic Press, Inc, 1970.

- [65] Simon J. Julier. The scaled unscented transformation. In *Proceedings of American Control Conference*, volume 6, pages 4555–4559, Anchorage, AK, USA, May 2002.
- [66] Simon J. Julier and Jeffery K. Uhlmann. Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations. In *Proceedings of American Control Conference*, volume 2, pages 887–892, Anchorage, AK, USA, May 2002.
- [67] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, March 2004.
- [68] Simon J. Julier, Jeffrey K. Uhlmann, and Hugh F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3), March 2000.
- [69] Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear Estimation*. Prentice-Hall, Inc, 2000. ISBN 0-13-022464-2.
- [70] Rudolph E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the American Society of Mechanical Engineering — Journal Basic Engineering, Series D*, 82:35–45, March 1960.
- [71] Rudolph E. Kalman and Richard S. Bucy. New results in linear filtering and prediction theory. *Transactions of the American Society of Mechanical Engineering — Journal Basic Engineering, Series D*, 83(Series D):95–108, March 1961.
- [72] Rickard Karlsson. *Particle Filtering for Positioning and Tracking Applications*. Dissertations no 924, Linköping Studies in Science and Technology, SE-581 83 Linköping, Sweden, March 2005.
- [73] Rickard Karlsson and Fredrik Gustafsson. Bayesian surface and underwater navigation. *IEEE Transactions on Signal Processing*, 54(11):4204–4213, November 2006.
- [74] Rickard Karlsson, Jonas Jansson, and Fredrik Gustafsson. Model-based statistical tracking and decision making for collision avoidance application. In *Proceedings of American Control Conference*, pages 3435–3440, Boston, MA, USA, July 2004.
- [75] Rickard Karlsson, Thomas B. Schön, and Fredrik Gustafsson. Complexity analysis of the marginalized particle filter. *IEEE Transactions on Signal Processing*, 53(11): 4408–4411, November 2005.
- [76] Steven M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*, volume 1. Prentice-Hall, Inc, 1993. ISBN 0-13-042268-1.
- [77] Steven M. Kay. *Fundamentals of Statistical Signal Processing: Detection Theory*, volume 2. Prentice-Hall, Inc, 1998. ISBN 0-13-504135-X.

- [78] Steven M. Kay and Dabasis Sengupta. Optimal detection in colored non-Gaussian noise with unknown parameter. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 12, pages 1087–1089, Dallas, TX, USA, April 1987.
- [79] Genshiro Kitagawa. Monte Carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, March 1996.
- [80] Augustine Kong, Jun S. Liu, and Wing H. Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, March 1994.
- [81] Jayesh H. Kotecha and Petar M Djurić. Gaussian particle filtering. *IEEE Transactions on Signal Processing*, 51(10):2592–2601, October 2003.
- [82] Jayesh H. Kotecha and Petar M Djurić. Gaussian sum particle filtering. *IEEE Transactions on Signal Processing*, 51(10):2602–2612, October 2003.
- [83] Stuart C. Kramer and Harold W. Sorenson. Bayesian parameter estimation. *IEEE Transactions on Automatic Control*, 33(2):217–222, February 1988.
- [84] Stuart C. Kramer and Harold W. Sorenson. Recursive Bayesian estimation using pice-wise constant approximations. *Automatica*, 24(6):789–801, November 1988.
- [85] Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951.
- [86] Solomon Kullback, John C. Keegel, and Joseph H. Kullback. *Topics in Statistical Information Theory*, volume 42 of *Lecture Notes in Statistics*. Springer-Verlag, 1987. ISBN 0-387-96512-2.
- [87] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. Comment on “A new method for the nonlinear transformation of means and covariances in filters and estimators”. *IEEE Transactions on Automatic Control*, 47(8), August 2002. Comments on [68].
- [88] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. Kalman filters for nonlinear systems: a comparison of performance. *International Journal of Control*, 77(7), May 2004.
- [89] Eric L. Lehmann. *Theory of Point Estimation*. Probability and Mathematical Statistics. John Wiley & Sons, Ltd, 1983. ISBN 0-471-05849-1.
- [90] Eric L. Lehmann. *Testing Statistical Hypotheses*. Probability and Mathematical Statistics. John Wiley & Sons, Ltd, 2 edition, 1986. ISBN 0-471-84083-1.
- [91] Robert Leland. The Kulback-Leibler information divergence for continuous systems using white noise theory. In *Proceedings of 38th IEEE Conference on Decision and Control*, pages 1903–1907, Phoenix, AZ, USA, December 1999.

- [92] Rong X. Li and Vesslin P. Jilkov. Survey of maneuvering target tracking. part I: Dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4): 1333–1364, October 2003.
- [93] Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, January 1991.
- [94] Gary Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, 42(6):1897–1908, December 1971.
- [95] Gary Lorden. Open-ended tests for Koopman-Darmois families. *The Annals of Statistics*, 1(4):633–643, July 1973.
- [96] Simon Maskell, Ben Alun-Jones, and Malcolm Macleod. A single instruction multiple data particle filter. In *Proceedings of Nonlinear Statistical Signal Processing Workshop*, Cambridge, UK, September 2006.
- [97] Michael D. McCool. Signal processing and general-purpose computing on GPUs. *IEEE Signal Processing Magazine*, 24(3):109–114, May 2007.
- [98] Antonio S. Montemayor, Juan J. Pantrigo, Ángel Sánchez, and Felipe Fernández. Particle filter on GPUs for real time tracking. In *Proceedings of SIGGRAPH*, Los Angeles, CA, USA, August 2004.
- [99] Antonio S. Montemayor, Juan J. Pantrigo, Raúl Cabido, Bryson R. Payne, Ángel Sánchez, and Felipe Fernández. Improving GPU particle filter with shader model 3.0 fir visual tracking. In *Proceedings of SIGGRAPH*, Boston, MA, USA, August 2006.
- [100] Jerzy Neyman and Egon S. Pearson. On the use and interpretation of certain test criteria for purposes of statistical inference: Part I. *Biometrika*, 20A(1/2):175–240, July 1928.
- [101] Jerzy Neyman and Egon S. Pearson. On the use and interpretation of certain test criteria for purposes of statistical inference: Part II. *Biometrika*, 20A(3/4):263–294, December 1928.
- [102] Wolfgang Niehsen. Generalized Gaussian modeling of correlated sources. *IEEE Transactions on Signal Processing*, 47(1):217–219, January 1999.
- [103] Wolfgang Niehsen. Robust Kalman filtering with generalized Gaussian measurement noise. *IEEE Transactions on Aerospace and Electronic Systems*, 38(4):1409–1412, October 2002.
- [104] Per-Johan Nordlund. *Sequential Monte Carlo Filters and Integrated Navigation*. Licentiate thesis no 945, Department of Electrical Engineering, Linköpings universitet, Sweden, May 2002.
- [105] NVIDIA. NVIDIA developer web-site. URL: <http://developer.nvidia.com>, 2006.

- [106] Man-Suk Oh. Monte Carlo integration via importance sampling: Dimensionality effect and an adaptive algorithm. In Nancy Flournoy and Robert K. Tsutakawa, editors, *Statistical Multiple Integration*, volume 115 of *Contemporary Mathematics*, pages 165–187. American Mathematical Society, Providence, RI, USA, 1991.
- [107] Tor A. Øigård, Alfred Hanssen, Roy E. Hansen, and Fred Godtliebsen. EM-estimation and modeling of heavy-tailed processes with the multivariate normal inverse Gaussian distribution. *Signal Processing*, 85(8):1655–1673, August 2005.
- [108] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill Book Co, 3 edition, 1991.
- [109] Matt Pharr, editor. *GPU Gems 2. Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley, 2005. ISBN 0-321-33559-7.
- [110] Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, June 1999.
- [111] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286 1989.
- [112] C. Radhakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, 37:81–91, 1945.
- [113] C. Radhakrishna Rao. Minimum variance and the estimation of several parameters. *Proceedings of the Cambridge Philosophical Society*, 43:280–283, 1946.
- [114] Brian D. Ripley. *Stochastic simulation*. John Wiley & Sons, Ltd, 1987. ISBN 0-471-81884-4.
- [115] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, Inc, 2004. ISBN 1-58053-631-X.
- [116] Theodore J. Rivlin. *The Chebyshev Polynomials*. John Wiley & Sons, Ltd, 1974. ISBN 0-471-72470-X.
- [117] Christian P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundation to Computational Implementation*. Springer texts in Statistics. Springer-Verlag, 2 edition, 2001.
- [118] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer-Verlag, New York, 1999. ISBN 0-387-98707-X.
- [119] Jakob Rosén. A framework for nonlinear filtering in MATLAB. Master’s thesis no LiTH-ISY-EX-05/3733--SE, Department of Electrical Engineering, Linköpings universitet, Sweden, October 2005.

- [120] Randi J. Rost. *OpenGL Shading Language*. Addison-Wesley, 2 edition, 2006. ISBN 0-321-33489-2.
- [121] Wilson J. Rugh. *Linear System Theory*. Prentice-Hall, Inc, 1996. ISBN 0-13-441205-2.
- [122] Yong Rui and Yunqiang Chen. Better proposal distributions: Object tracking using unscented particle filter. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 786–793, Kauai, HI, USA, December 2001.
- [123] Tina H. Rydberg. The normal inverse Gaussian Lévy process: Simulation and approximation. *Communications in Statistics — Stochastic Models*, 13(4):887–910, 1997.
- [124] Stanley F. Schmidt. Application of state-space methods to navigation problems. *Advances in Control Systems*, 3:293–340, 1966.
- [125] Thomas Schön, Fredrik Gustafsson, and Per-Johan Nordlund. Marginalized particle filters for mixed linear / nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7):2279–2289, July 2005.
- [126] Thomas B. Schön. *Estimation of Nonlinear Dynamic Systems — Theory and Applications*. Dissertations no 998, Linköping Studies in Science and Technology, SE-581 83 Linköping, Sweden, February 2006.
- [127] Thomas B. Schön, Rickard Karlsson, and Fredrik Gustafsson. The marginalized particle filter in practice. In *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, March 2006.
- [128] Debasis Sengupta and Steven M. Kay. Efficient estimation for non-Gaussian autoregressive processes. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(6):785–794, June 1989.
- [129] Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programming Language. The official guide to learning OpenGL, Version 2*. Addison-Wesley, 5 edition, 2005. ISBN 0-321-33573-2.
- [130] Gerald L. Smith, Stanley F. Schmidt, and Leonard A. McGee. Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle. Technical Report NASA TR R-135, National Aeronautics and Space Administration, 1962.
- [131] Harold W. Sorenson, editor. *Kalman Filtering: Theory and Applications*. IEEE Press, 1985.
- [132] Harold W. Sorenson and Daniel L. Alspach. Recursive Bayesian estimation using Gaussian sums. *Automatica*, 7(4):465–479, July 1971.
- [133] Ondřej Straka and Miroslav Šimandl. Using the Bhattacharyya distance in functional sampling density of particle filter. In *Proceedings of 16th Triennial IFAC World Congress*, Prague, Czech Republic, July 2005.

- [134] M. Sussman, W. Crutchfield, and M. Papakipos. Pseudorandom number generation on the GPU. In *Graphics Hardware. Eurographics Symposium Proceedings*, pages 87–94, Vienna, Austria, August 2006.
- [135] Martin Svensson. Implementation and evaluation of Bayesian terrain navigation. Master's thesis no LiTH-ISY-EX-2039, Department of Electrical Engineering, Linköpings universitet, Sweden, March 1999. In Swedish.
- [136] Peter Swerling. First order propagation in a stagewise smoothing procedure for satellite observations. *The Journal of the Astronautical Sciences*, 6:46–52, 1959.
- [137] Chih J. K. Tan. The PLFG parallel pseud-random number generator. *Future Generation Computer Systems*, 18:693–698, 2002.
- [138] James H. Taylor. The Cramér-Rao estimation lower bound computation for deterministic nonlinear systems. *IEEE Transactions on Automatic Control*, 24(2): 343–344, April 1979.
- [139] Robert M. Taylor, Jr, Brian P. Flanagan, and John A. Uber. Computing the recursive posterior Cramer-Rao bound for a nonlinear nonstationary system. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume VI, pages 673–676, Hong Kong, April 2003. The conference was canceled.
- [140] Petr Tichavský, Carlos H. Muravchik, and Arye Nehorai. Posterior Cramér-Rao discrete-time nonlinear filtering. *IEEE Transactions on Signal Processing*, 46(5): 1386–1396, May 1998.
- [141] David Törnqvist, Fredrik Gustafsson, and Inger Klein. GLR tests for fault detection over sliding data windows. In *Proceedings of 16th Triennial IFAC World Congress*, Prague, Czech Republic, July 2005.
- [142] Rudolph van der Merwe and Eric A. Wan. The square-root unscented Kalman filter for state and parameter-estimation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3461–3464, Salt Lake City, UT, USA, May 2001.
- [143] Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric Wan. The unscented particle filter. Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, August 2000.
- [144] Harry S. van Trees. *Part I. Detection, Estimation, and Modulation Theory*. Detection, Estimation, and Modulation Theory. John Wiley & Sons, Ltd, 1968. ISBN 0-471-89955-0.
- [145] Abraham Wald. Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical Society*, 54(3):426–482, November 1943.
- [146] Eric A. Wan and Rudolph van der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, Lake Louise, AB, Canada, October 2000.

- 
- [147] Yuanxin Wu, Dewen Hu, Meiping Wu, and Xiaoping Hu. Unscented Kalman filtering for additive noise case: Augmented versus nonaugmented. *IEEE Signal Processing Letters*, 12(5):357–360, May 2005.

**PhD Dissertations**  
**Division of Automatic Control**  
**Linköping University**

- M. Millnert:** Identification and control of systems subject to abrupt changes. Thesis No. 82, 1982. ISBN 91-7372-542-0.
- A. J. M. van Overbeek:** On-line structure selection for the identification of multivariable systems. Thesis No. 86, 1982. ISBN 91-7372-586-2.
- B. Bengtsson:** On some control problems for queues. Thesis No. 87, 1982. ISBN 91-7372-593-5.
- S. Ljung:** Fast algorithms for integral equations and least squares identification problems. Thesis No. 93, 1983. ISBN 91-7372-641-9.
- H. Jonson:** A Newton method for solving non-linear optimal control problems with general constraints. Thesis No. 104, 1983. ISBN 91-7372-718-0.
- E. Trulsson:** Adaptive control based on explicit criterion minimization. Thesis No. 106, 1983. ISBN 91-7372-728-8.
- K. Nordström:** Uncertainty, robustness and sensitivity reduction in the design of single input control systems. Thesis No. 162, 1987. ISBN 91-7870-170-8.
- B. Wahlberg:** On the identification and approximation of linear systems. Thesis No. 163, 1987. ISBN 91-7870-175-9.
- S. Gunnarsson:** Frequency domain aspects of modeling and control in adaptive systems. Thesis No. 194, 1988. ISBN 91-7870-380-8.
- A. Isaksson:** On system identification in one and two dimensions with signal processing applications. Thesis No. 196, 1988. ISBN 91-7870-383-2.
- M. Viberg:** Subspace fitting concepts in sensor array processing. Thesis No. 217, 1989. ISBN 91-7870-529-0.
- K. Forsman:** Constructive commutative algebra in nonlinear control theory. Thesis No. 261, 1991. ISBN 91-7870-827-3.
- F. Gustafsson:** Estimation of discrete parameters in linear systems. Thesis No. 271, 1992. ISBN 91-7870-876-1.
- P. Nagy:** Tools for knowledge-based signal processing with applications to system identification. Thesis No. 280, 1992. ISBN 91-7870-962-8.
- T. Svensson:** Mathematical tools and software for analysis and design of nonlinear control systems. Thesis No. 285, 1992. ISBN 91-7870-989-X.
- S. Andersson:** On dimension reduction in sensor array signal processing. Thesis No. 290, 1992. ISBN 91-7871-015-4.
- H. Hjalmarsson:** Aspects on incomplete modeling in system identification. Thesis No. 298, 1993. ISBN 91-7871-070-7.
- I. Klein:** Automatic synthesis of sequential control schemes. Thesis No. 305, 1993. ISBN 91-7871-090-1.
- J.-E. Strömberg:** A mode switching modelling philosophy. Thesis No. 353, 1994. ISBN 91-7871-430-3.
- K. Wang Chen:** Transformation and symbolic calculations in filtering and control. Thesis No. 361, 1994. ISBN 91-7871-467-2.
- T. McKelvey:** Identification of state-space models from time and frequency data. Thesis No. 380, 1995. ISBN 91-7871-531-8.
- J. Sjöberg:** Non-linear system identification with neural networks. Thesis No. 381, 1995. ISBN 91-7871-534-2.
- R. Germundsson:** Symbolic systems – theory, computation and applications. Thesis No. 389, 1995. ISBN 91-7871-578-4.
- P. Pucar:** Modeling and segmentation using multiple models. Thesis No. 405, 1995. ISBN 91-7871-627-6.
- H. Fortell:** Algebraic approaches to normal forms and zero dynamics. Thesis No. 407, 1995. ISBN 91-7871-629-2.

**A. Helmersson:** Methods for robust gain scheduling. Thesis No. 406, 1995. ISBN 91-7871-628-4.

**P. Lindskog:** Methods, algorithms and tools for system identification based on prior knowledge. Thesis No. 436, 1996. ISBN 91-7871-424-8.

**J. Gunnarsson:** Symbolic methods and tools for discrete event dynamic systems. Thesis No. 477, 1997. ISBN 91-7871-917-8.

**M. Jirstrand:** Constructive methods for inequality constraints in control. Thesis No. 527, 1998. ISBN 91-7219-187-2.

**U. Forssell:** Closed-loop identification: Methods, theory, and applications. Thesis No. 566, 1999. ISBN 91-7219-432-4.

**A. Stenman:** Model on demand: Algorithms, analysis and applications. Thesis No. 571, 1999. ISBN 91-7219-450-2.

**N. Bergman:** Recursive Bayesian estimation: Navigation and tracking applications. Thesis No. 579, 1999. ISBN 91-7219-473-1.

**K. Edström:** Switched bond graphs: Simulation and analysis. Thesis No. 586, 1999. ISBN 91-7219-493-6.

**M. Larsson:** Behavioral and structural model based approaches to discrete diagnosis. Thesis No. 608, 1999. ISBN 91-7219-615-5.

**F. Gunnarsson:** Power control in cellular radio systems: Analysis, design and estimation. Thesis No. 623, 2000. ISBN 91-7219-689-0.

**V. Einarsson:** Model checking methods for mode switching systems. Thesis No. 652, 2000. ISBN 91-7219-836-2.

**M. Norrlöf:** Iterative learning control: Analysis, design, and experiments. Thesis No. 653, 2000. ISBN 91-7219-837-0.

**F. Tjärnström:** Variance expressions and model reduction in system identification. Thesis No. 730, 2002. ISBN 91-7373-253-2.

**J. Löfberg:** Minimax approaches to robust model predictive control. Thesis No. 812, 2003. ISBN 91-7373-622-8.

**J. Roll:** Local and piecewise affine approaches to system identification. Thesis No. 802, 2003. ISBN 91-7373-608-2.

**J. Elbornsson:** Analysis, estimation and compensation of mismatch effects in A/D converters. Thesis No. 811, 2003. ISBN 91-7373-621-X.

**O. Härkegård:** Backstepping and control allocation with applications to flight control. Thesis No. 820, 2003. ISBN 91-7373-647-3.

**R. Wallin:** Optimization algorithms for system analysis and identification. Thesis No. 919, 2004. ISBN 91-85297-19-4.

**D. Lindgren:** Projection methods for classification and identification. Thesis No. 915, 2005. ISBN 91-85297-06-2.

**R. Karlsson:** Particle Filtering for Positioning and Tracking Applications. Thesis No. 924, 2005. ISBN 91-85297-34-8.

**J. Jansson:** Collision Avoidance Theory with Applications to Automotive Collision Mitigation. Thesis No. 950, 2005. ISBN 91-85299-45-6.

**E. Geijer Lundin:** Uplink Load in CDMA Cellular Radio Systems. Thesis No. 977, 2005. ISBN 91-85457-49-3.

**M. Enqvist:** Linear Models of Nonlinear Systems. Thesis No. 985, 2005. ISBN 91-85457-64-7.

**T. B. Schön:** Estimation of Nonlinear Dynamic Systems — Theory and Applications. Thesis No. 998, 2006. ISBN 91-85497-03-7.

**I. Lind:** Regressor and Structure Selection — Uses of ANOVA in System Identification. Thesis No. 1012, 2006. ISBN 91-85523-98-4.

**J. Gillberg:** Frequency Domain Identification of Continuous-Time Systems Reconstruction and Robustness. Thesis No. 1031, 2006. ISBN 91-85523-34-8.

**M. Gerdin:** Identification and Estimation for Models Described by Differential-Algebraic Equations. Thesis No. 1046, 2006. ISBN 91-85643-87-4.

**C. Grönwall:** Ground Object Recognition using Laser Radar Data – Geometric Fitting, Performance Analysis, and Applications. Thesis No. 1055, 2006. ISBN 91-85643-53-X.

**A. Eidehall:** Tracking and threat assessment for automotive collision avoidance. Thesis No. 1066, 2007. ISBN 91-85643-10-6.

**F. Eng:** Non-Uniform Sampling in Statistical Signal Processing. Thesis No. 1082, 2007. ISBN 978-91-85715-49-7.

**E. Wernholt:** Multivariable Frequency-Domain Identification of Industrial Robots. Thesis No. 1138, 2007. ISBN 978-91-85895-72-4.

**D. Axehill:** Integer Quadratic Programming for Control and Communication. Thesis No. 1158, 2008. ISBN 978-91-85523-03-0.