

Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models

Rudolph van der Merwe

M.Eng., Electrical Engineering, University of Stellenbosch, 1998
B.Eng., Electrical Engineering, University of Stellenbosch, 1995

A dissertation submitted to the faculty of the
OGI School of Science & Engineering at
Oregon Health & Science University
in partial fulfillment of the
requirements for the degree
Doctor of Philosophy
in
Electrical and Computer Engineering

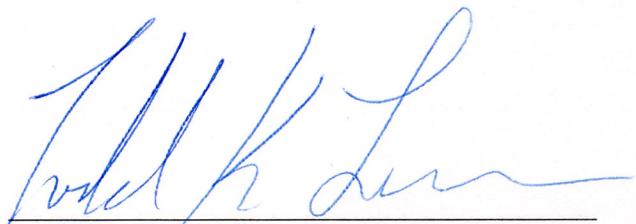
April 2004

© Copyright 2004 by Rudolph van der Merwe
All Rights Reserved

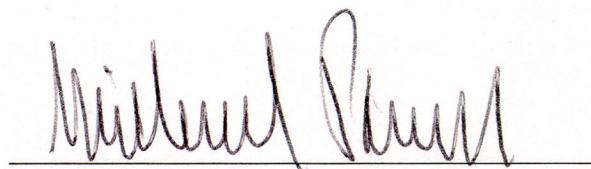
The dissertation "Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models" by Rudolph van der Merwe has been examined and approved by the following Examination Committee:



Dr. Eric A. Wan
Associate Professor
Dept. of Computer Science & Engineering
Thesis Research Adviser



Dr. Todd K. Leen
Professor
Dept. of Computer Science & Engineering



Dr. Misha Pavel
Professor
Dept. of Biomedical Engineering



Dr. Dieter Fox
Assistant Professor
Dept. of Computer Science & Engineering
University of Washington
External Committee Member

Dedication

To Zoë,

My companion in love and life
on this long and weirdly wonderful, arduous journey.
For not undertaking this journey we would not have met.
For not meeting, this journey surely would have failed.

Dankie vir alles!

Acknowledgments

I would like to thank the following people who all contributed in some form to the success of this endeavour:

My professors and colleagues at OGI for their intellectual support, and for providing a congenial and stimulating academic environment. In particular, I want to thank my advisor, Eric Wan, for his time, patience, insight and knowledge imparted to me over the last five years. Special thanks also go to Todd Leen for sharing his mathematical expertise and wisdom, as well as emphasizing the importance of rigor in research. Finally, I want to thank the remaining members of my thesis committee, Misha Pavel and Dieter Fox, for their input and suggestions along the way, as well as for taking the time to review this work.

My family for their love and support. Chris, Henriette, Mary, Christelle, Theron, Rykie & Gary: Thank you for always believing in me.

All my wonderful friends (of whom there are simply too many to mention by name¹): Thanks for your friendship and support. It kept me sane.

This work was supported in part by the following grants: NSF ECS-0083106, DARPA F33615-98-C-3516 & ONR N0014-02-C-0248.

¹One exception: Dr. Ashvin Goel [67], consider yourself cited.

Contents

Dedication	iv
Acknowledgments	v
Abstract	xviii
1 Introduction	1
1.1 Compact Overview	1
1.2 Reasoning under Uncertainty	1
1.3 Probabilistic Inference	2
1.4 Optimal Solution: Recursive Bayesian Estimation	6
1.5 Approximate Solutions	7
1.5.1 Gaussian Approximate Methods	9
1.5.2 Sequential Monte Carlo methods	14
1.6 Research Objectives and Work Overview	15
1.6.1 Summary of research objectives	16
1.7 Contributions of this Work	19
1.8 Thesis Outline	23
1.8.1 Main thesis chapters	23
1.8.2 Appendices	25
1.9 Publications	25
2 Gaussian Approximate Bayesian Estimation:	
The Kalman Filter Framework	27
2.1 Introduction	27
2.2 Optimal Gaussian Approximate Linear Bayesian Update	28

2.3	Nonlinear Transformations of Random Variables	31
2.4	The EKF and its Flaws	35
2.5	Demonstration of EKF Inaccuracy	37
2.5.1	Analytic scalar example	37
2.5.2	Arbitrary nonlinear transformation of Gaussian random variable . .	40
2.5.3	Conversion of polar to Cartesian coordinates	41
2.6	State, Parameter and Dual Estimation	43
2.6.1	State estimation	43
2.6.2	Parameter estimation	44
2.6.3	Dual estimation	45
2.7	Chapter Summary	47
3	Sigma-Point Kalman Filters:	
	Derivation, Algorithmic Implementation, Applications & Extensions .	49
3.1	Introduction	49
3.2	The Unscented Kalman Filter	50
3.2.1	The unscented transformation	51
3.2.2	The scaled unscented transformation	55
3.2.3	Implementing the unscented Kalman filter	59
3.3	The Central Difference Kalman Filter	62
3.3.1	Second order Sterling polynomial interpolation	62
3.3.2	Estimation of posterior mean and covariance by Sterling interpolation	65
3.3.3	Demonstration of Sterling interpolation for posterior statistics estimation	69
3.3.4	Implementing the central difference Kalman filter	71
3.4	Sigma-Point Kalman Filters and the Sigma-point Approach	74
3.4.1	The Sigma-point Approach	74
3.4.2	Sigma-point Kalman filters	79
3.5	SPKF Application to State, Parameter and Dual estimation	81
3.5.1	SPKF State Estimation	82

3.5.2	SPKF Parameter Estimation	89
3.5.3	SPKF Dual Estimation	102
3.6	SPKF Extensions & Implementation Variations	108
3.6.1	Additive noise forms	108
3.6.2	Square-root forms	111
3.6.3	SPKF Based Smoothing	124
3.7	Chapter Summary	126
4	Sigma-Point Kalman Filters:	
	Theoretical Analysis	128
4.1	Introduction	128
4.2	Alternate Interpretation of Sigma-Point Approach	129
4.2.1	Statistical linearization	130
4.2.2	Weighted statistical linear regression (WSLR)	132
4.2.3	Relationship between WSLR and the sigma-point approach	135
4.2.4	Demonstration of statistical linearization as implicitly used by the sigma-point approach	137
4.3	Accuracy of Sigma-point Approach	139
4.3.1	Posterior mean accuracy	140
4.3.2	Posterior covariance accuracy	142
4.3.3	Demonstration of accuracy of sigma-point approach	145
4.4	Relationship between Sigma-Point Approach and Gaussian Quadrature	146
4.5	Theoretical Analysis of SPKF based Parameter Estimation	148
4.5.1	MAP Parameter Estimation	149
4.5.2	The Gauss-Newton Method for Nonlinear Least Squares	156
4.5.3	The SPKF Parameter Estimation Measurement Update as an Online Gauss-Newton Method	158
4.5.4	Discussion and Experimental Demonstration	161
4.6	Summary of SPKF properties	166
4.7	Chapter Summary	167

5 SPKF Based UAV Autonomy	169
5.1 Introduction	169
5.2 Experimental Platform	171
5.2.1 Vehicle Airframe	172
5.2.2 Avionics System	174
5.2.3 Nonlinear Vehicle Models	176
5.2.4 Sensor Models	187
5.2.5 Simulation Systems	189
5.2.6 Control System	190
5.3 SPKF Based Estimation System	192
5.3.1 SPKF based Time Delayed Sensor Fusion	200
5.3.2 SPKF based quaternion unity norm constraint enforcement	206
5.4 State Estimation Experimental Results	210
5.4.1 Experiment SE0: Determining effect of IMU offset term in vehicle dynamics	210
5.4.2 Experiment SE1: EKF vs. SPKF (without latency compensation) vs. SPKF (with latency compensation)	212
5.4.3 Experiment SE2: Closed loop control and estimation performance	216
5.4.4 Experiment SE3: IMU degradation experiments	219
5.5 Parameter Estimation Experimental Results	226
5.5.1 Implementation Issues	227
5.5.2 Experiment PE1: Static system parameter identification	229
5.5.3 Experiment PE2: Dynamic system parameter identification and tracking	229
5.6 Dual Estimation Experimental Results	231
5.6.1 System Implementation	233
5.6.2 Experiment D1: Joint estimation of auxiliary states and dynamic tracking of vehicle mass	235
5.7 Chapter Summary	237

6 Non-Gaussian Bayesian Estimation:	
Sequential Monte Carlo / SPKF Hybrids	238
6.1 Introduction	238
6.2 Particle Filters: Monte Carlo Simulation and Sequential Importance Sampling	239
6.2.1 Perfect Monte Carlo Simulation	239
6.2.2 Bayesian Importance Sampling	240
6.2.3 Sequential Importance Sampling	242
6.2.4 Mitigating SIS Degeneracy : Resampling	247
6.2.5 The Particle Filter Algorithm	249
6.2.6 Demonstration of Nonlinear Non-Gaussian Inference	251
6.3 Improving Particle Filters: Designing Better Proposal Distributions	255
6.3.1 Sigma-Point Particle Filter	260
6.3.2 Gaussian Mixture Sigma-Point Particle Filters	264
6.3.3 Discussion about differences between SPPF and GMSPPF	274
6.4 Experimental Results	275
6.4.1 Experiment 1 : Scalar Nonlinear, Non-Gaussian Time-series Estimation	277
6.4.2 Experiment 2 : Econometrics - Pricing Financial Options	278
6.4.3 Experiment 3 : Comparison of PF, SPPF & GMSPPF on nonlinear, non-Gaussian state estimation problem	287
6.4.4 Experiment 4 : Human Face Tracking	290
6.4.5 Experiment 5 : Mobile Robot Localization	298
6.5 Chapter Summary	311
7 Conclusions and Future Work	313
7.1 Introduction	313
7.2 Concluding Summary	313
7.3 General Overview	315
7.4 Possible Extensions & Future Work	317
7.5 Derived & Related Work	324

Bibliography	327
A The Kalman Filter	346
A.1 Introduction	346
A.2 MMSE Derivation of Kalman Filter	346
A.3 Gaussian MAP Derivation of Kalman Filter	350
A.4 The Full Kalman Filter Algorithm	354
A.5 Alternative Forms	355
A.5.1 Joseph's Form of the Measurement Update	355
A.5.2 Schmidt-Kalman Filter	357
B Navigation Primer	359
B.1 Introduction	359
B.2 Reference Frames	359
B.3 Attitude Quaternion	362
B.4 Matrix Exponentiation of a Skew-Symmetric Matrix	365
C ReBEL Toolkit	369
C.1 A Recursive Bayesian Estimation Library for Matlab®	369
Biographical Note	376

List of Algorithms

1	The extended Kalman filter (EKF)	36
2	The unscented Kalman filter (UKF)	60
3	The central difference Kalman filter (CDKF)	71
4	The unscented Kalman filter (UKF) - SPKF formulation	79
5	The central difference Kalman filter (CDKF) - SPKF formulation	79
6	The unscented Kalman filter (UKF) - parameter estimation form	93
7	The central difference Kalman filter (CDKF) - parameter estimation form . .	94
8	The unscented Kalman filter (UDKF) - additive noise case	108
9	The central difference Kalman filter (CDKF) - additive noise case	110
10	The square-root UKF (SR-UKF) - general state estimation form	114
11	The square-root UKF (SR-UKF) - state estimation form, additive noise case	115
12	The square-root CDKF (SR-CDKF) - general state estimation form	116
13	The square-root CDKF (SR-CDKF) - state estimation form, additive noise case	118
14	The square-root UKF (SR-UKF) - parameter estimation	122
15	The square-root CDKF (SR-CDKF) - parameter estimation form	123
16	The particle filter (PF)	250
17	The sigma-point particle filter (SPPF)	261
18	The Gaussian mixture sigma-point particle filter (GMSPPF)	269
19	The Kalman filter (KF)	356

List of Tables

2.1	Calculating the statistics of a Gaussian random variable that undergoes a quadratic nonlinear transformation.	40
3.1	Estimation of Mackey-Glass time-series with the EKF, UKF and CDKF using a known model: Monte-Carlo averaged (200 runs) estimation error. . .	84
3.2	Inverted double pendulum parameter estimation.	98
3.3	Square-root SPKF state estimation results for Mackey-Glass time-series problem.	119
3.4	Comparison of smoother performance.	125
4.1	Calculating the statistics of a Gaussian random variable that undergoes a quadratic nonlinear transformation.	146
4.2	Gauss-Hermite quadrature abscissas and weights for small n	148
5.1	MIT-Draper-XCell-90 model state vector components	178
5.2	MIT-Draper-XCell-90 model parameters	179
5.3	Kinematic model state vector components.	181
5.4	State estimation results : EKF vs. SPKF (with and without GPS latency compensation).	215
6.1	Scalar nonlinear non-Gaussian state estimation experiment results.	278
6.2	Financial options pricing: One-step-ahead normalized square errors over 100 runs.	286
6.3	Non-stationary, nonlinear, non-Gaussian time series estimation experiment. .	290

List of Figures

1.1	Reasoning under uncertainty : Robot Localization	2
1.2	Probabilistic inference	4
1.3	Probabilistic dynamic state-space model	5
1.4	Grid based filters vs. sequential Monte Carlo methods.	10
1.5	Unmanned aerial vehicle (UAV) guidance, navigation and control (GNC) system.	18
2.1	Optimal vs. EKF approximate transformation of Gaussian random variables.	41
2.2	Optimal vs. EKF results on polar-to-Cartesian transformation experiment..	42
2.3	State estimation: block diagram of a discrete-time nonlinear DSSM	44
2.4	Dual estimation	46
3.1	Example of weighted sigma-point set for a 2D Gaussian RV.	52
3.2	Schematic diagram of the unscented transformation.	53
3.3	Demonstration of the SUT for mean and covariance propagation.	58
3.4	Optimal vs. linearized vs. Sterling vs. SUT approximation for estimating the statistic of a or polar-to-Cartesian coordinate transformation.	70
3.5	Estimation of Mackey-Glass time-series with the EKF and SPKF using a known model.	83
3.6	Estimation of Mackey-Glass time-series with the EKF and SPKF using a known model (detail).	85
3.7	Monte-Carlo analysis of estimation error for Mackey-Glass nonlinear (chaotic) time-series estimation problem.	86
3.8	State estimation for inverted double pendulum problem.	88
3.9	SPKF neural network training examples.	96

3.10	Neural network parameter estimation using different methods for noise es- timation.	97
3.11	Inverted double pendulum parameter estimation.	98
3.12	Singhal and Wu's <i>Four Region</i> classification problem.	99
3.13	Rossenbrock's "Banana" optimization problem.	101
3.14	Schematic diagrams of two main dual estimation filter implementations. . .	103
3.15	Dual nonlinear time-series estimation experiment.	106
3.16	Inverted double pendulum dual estimation and control experiment.	107
3.17	Square-root SPKF parameter estimation.	121
3.18	Forward/backward neural network prediction training for sigma-point Kalman smoother.	125
4.1	Stochastic linearization / weighted statistical linear regression	138
4.2	Optimization issues: local vs. average (expected) gradient of a nonlinear function.	164
5.1	Unmanned aerial vehicle (UAV) guidance, navigation and control (GNC) system.	170
5.2	Close-up of instrumented X-Cell-90 helicopter in flight.	172
5.3	Instrumented X-Cell-90 helicopter in flight.	173
5.4	Schematic diagram of the software based UAV simulation system.	190
5.5	System with a delayed measurement due to sensor latency.	196
5.6	IMU offset impact determination experiment.	211
5.7	Simulator flight plan for UAV estimation experiments.	213
5.8	State estimation results: EKF vs. SPKF (without GPS latency compensa- tion) vs. SPKF (with GPS latency compensation)	214
5.9	State estimation results (position and velocity error) : SPKF vs. EKF . .	217
5.10	State estimation results (Euler angle errors) : SPKF vs. EKF . . .	217
5.11	Comparing closed-loop estimation and control results between the EKF and SPKF.	218
5.12	Typical bias/drift-rate plots for IMU simulation.	222

5.13	IMU degradation experiment: additive noise	223
5.14	IMU degradation experiment: drift/bias	224
5.15	IMU degradation experiment: IMU update rate	225
5.16	Pure parameter estimation using full nonlinear model and known states. . .	230
5.17	Tracking a time-varying system parameter.	232
5.18	Joint estimation : Hidden system states	236
5.19	Joint estimation : tracking time-varying system mass	236
6.1	Moving particles to areas of high-likelihood	246
6.2	Resampling	249
6.3	Schematic diagram of a generic particle filter (SIR-PF)	252
6.4	Non-Gaussian (bimodal) state estimation demonstration	254
6.5	Schematic diagram of the Gaussian mixture sigma-point particle filter (GM-SPPF).	269
6.6	Scalar nonlinear non-Gaussian state estimation experiment results: state estimates	279
6.7	Scalar nonlinear non-Gaussian state estimation experiment results: covariance estimates	279
6.8	Volatility smile plots for options on the FTSE-100 index.	281
6.9	Probability smile plots for options on the FTSE-100 index.	282
6.10	SPPF one-step-ahead predictions on the call and put option's prices with confidence intervals.	283
6.11	Estimated interest rate and volatility.	284
6.12	Probability distribution of the implied interest rate.	285
6.13	Probability distribution of the implied volatility.	285
6.14	GMM approximation of heavy tailed, asymmetric Gamma distributed noise distribution.	288
6.15	Non-stationary, nonlinear, non-Gaussian time series estimation experiment. .	289
6.16	Human face tracking using the SPPF	291
6.17	SPPF based human face tracking	296

6.18	Mobile robot localization: map and physical setup	298
6.19	Mobile robot localization: PF vs GMSSPF results	304
6.20	Mobile robot localization: k=1 close-up	306
6.21	Mobile robot localization: k=2 close-up	307
6.22	Mobile robot localization: k=10 close-up	308
6.23	Mobile robot localization: k=20 close-up	309
6.24	Mobile robot localization: k=50 close-up	310
B.1	Reference frames used for terrestrial navigation	360
B.2	Vehicle body fixed reference frame.	361
B.3	Quaternion representation of an arbitrary rotation.	364
C.1	The ReBEL toolkit : general information	372
C.2	The ReBEL Toolkit : general, modular unified framework for probabilistic inference in generic DSSMs.	373
C.3	The ReBEL Toolkit : code example	374
C.4	The ReBEL Toolkit website	375

Abstract

Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models

Rudolph van der Merwe

Supervising Professor: Dr. Eric A. Wan

Probabilistic inference is the problem of estimating the hidden variables (states or parameters) of a system in an optimal and consistent fashion as a set of noisy or incomplete observations of the system becomes available online. The optimal solution to this problem is given by the recursive Bayesian estimation algorithm which recursively updates the posterior density of the system state as new observations arrive. This posterior density constitutes the complete solution to the probabilistic inference problem, and allows us to calculate any "optimal" estimate of the state. Unfortunately, for most real-world problems, the optimal Bayesian recursion is intractable and approximate solutions must be used. Within the space of approximate solutions, the extended Kalman filter (EKF) has become one of the most widely used algorithms with applications in state, parameter and dual estimation. Unfortunately, the EKF is based on a sub-optimal implementation of the recursive Bayesian estimation framework applied to Gaussian random variables. This can seriously affect the accuracy or even lead to divergence of any inference system that is based on the EKF or that uses the EKF as a component part. Recently a number of related novel, more accurate and theoretically better motivated algorithmic alternatives to the EKF have surfaced in the literature, with specific application to state estimation for automatic control. We have extended these algorithms, all based on derivativeless deterministic

sampling based approximations of the relevant Gaussian statistics, to a family of algorithms called *Sigma-Point Kalman Filters* (SPKF). Furthermore, we successfully expanded the use of this group of algorithms (SPKFs) within the general field of probabilistic inference and machine learning, both as stand-alone filters and as subcomponents of more powerful sequential Monte Carlo methods (particle filters). We have consistently shown that there are large performance benefits to be gained by applying Sigma-Point Kalman filters to areas where EKFs have been used as the de facto standard in the past, as well as in new areas where the use of the EKF is impossible.

Chapter 1

Introduction

1.1 Compact Overview

This chapter will first describe the broad problem domain addressed by the body of work presented in this thesis. After this, a compact literature overview of related work in the field is given. Finally, a summary of the specific contributions of the work presented in this dissertation as well as a overview of the thesis itself is provided.

1.2 Reasoning under Uncertainty

Thanks in part to the relentless accuracy of Moore’s Law [138] over the last 40 years, cybernetic systems have become more and more capable of tackling hard problems outside the confines of the research lab. We are currently exploring the solar system with robotic proxies capable of autonomous reasoning and action [141, 49]; Neural network based systems are used daily for automatic recognition and classification of handwritten text for purposes of large volume mail-sorting and check verification [113, 112]; Autonomous biometric systems (voice and face recognition, etc.) are becoming more prevalent in areas such as homeland security and counter-terrorism [206, 207]. These are but a few examples of such successful systems, but they do share a very important commonality: They all operate in the *real world* and therefor have to *deal with uncertainty*. Consider the situation shown in Figure 1.1: How does a learning machine (in this case a mobile robot), determine its own pose (position and orientation) relative to its environment using noisy (uncertain) sensor measurements? Furthermore, once it has calculated some form of optimal estimate of its pose, how much “trust” should the robot have in this inferred quantity? In essence,

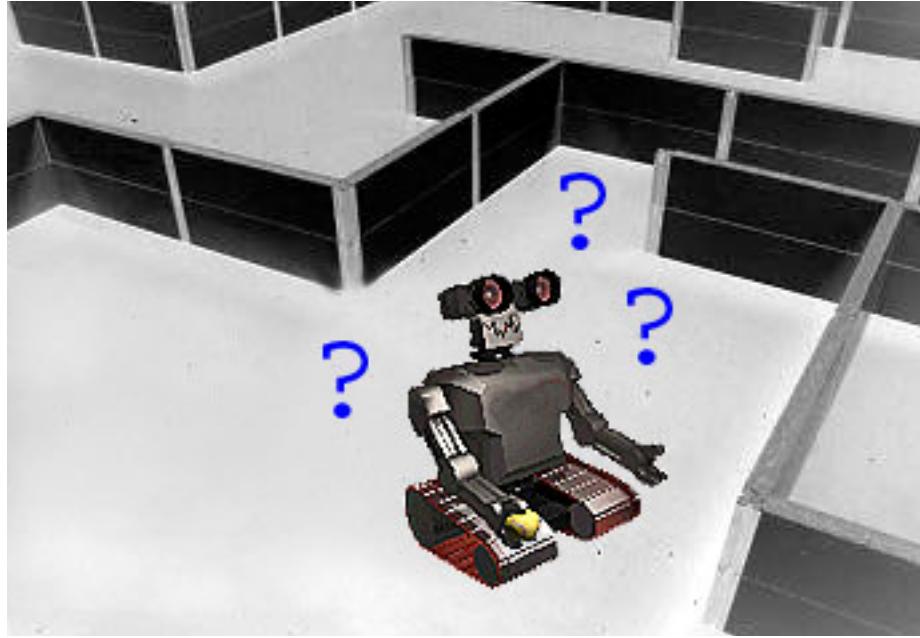


Figure 1.1: *Reasoning under uncertainty*: Using only noisy sensor measurements, how can the robot determine its pose (position and orientation) relative to its environment? Furthermore, how can this be accomplished in a manner that optimally uses all the available information and deal with the inherent uncertainty in a robust manner?

the estimate itself constitutes an *uncertain belief* of the underlying “reality” of the robots situation. In order to design such machines that *learn from*, *reason about*, and *act upon* the real world, we need to represent uncertainty. Probability theory provides a language for representing these uncertain beliefs and a calculus for manipulating these beliefs in a consistent manner [33, 89, 151]. In essence, probability theory allows us to *reason under uncertainty*. This process is called *probabilistic inference*.

1.3 Probabilistic Inference

Probabilistic inference is the problem of estimating the hidden variables (states or parameters) of a system in an optimal and consistent fashion (using probability theory) given noisy or incomplete observations. This general framework is depicted in Figure 1.2.

In particular, we will be addressing the *sequential (recursive) probabilistic inference* problem within discrete-time nonlinear dynamic systems that can be described by a *dynamic state-space model* (DSSM) as shown in Figure 1.3. The hidden system state \mathbf{x}_k ,

with initial probability density $p(\mathbf{x}_0)$, evolves over time (k is the discrete time index) as an indirect or partially observed first order Markov process according to the conditional probability density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. The observations \mathbf{y}_k are conditionally independent given the state and are generated according to the conditional probability density $p(\mathbf{y}_k|\mathbf{x}_k)$. The DSSM can also be written as a set of nonlinear system equations

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k; \mathbf{w}) \quad (1.1)$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k; \mathbf{w}) \quad (1.2)$$

where \mathbf{v}_k is the process noise that drives the dynamic system through the nonlinear state transition function \mathbf{f} , and \mathbf{n}_k is the observation or measurement noise corrupting the observation of the state through the nonlinear observation function \mathbf{h} . The state transition density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is fully specified by \mathbf{f} and the process noise distribution $p(\mathbf{v}_k)$, whereas \mathbf{h} and the observation noise distribution $p(\mathbf{n}_k)$ fully specify the observation likelihood $p(\mathbf{y}_k|\mathbf{x}_k)$. The exogenous input to the system, \mathbf{u}_k , is assumed known. Both \mathbf{f} and/or \mathbf{h} are parameterized via the parameter vector \mathbf{w} . The dynamic state-space model, together with the known statistics of the noise random variables as well as the prior distributions of the system states, defines a probabilistic generative model of how the system evolves over time and of how we (partially or inaccurately) observe this hidden state evolution. This process is depicted in Figure 1.3.

Dynamic state-space models (as defined in this thesis) can be interpreted as a special case of the more general framework of *Bayesian networks* [140] (also known as Bayes nets), which in turn is the combination of *Bayesian probability theory* [89] and *graphical models* [93]. Bayesian networks do not explicitly “hard code” the notion of *temporal* information flow, they rather directly model the probabilistic dependence between the different nodes in a graph. These dependences are indicated by connecting edges between the nodes, i.e., nodes that are not connected by an edge have no direct probabilistic dependence on each other. The edges are in general not directed, but directed edges (indicated by arrows) can be used to explicitly model conditional dependence and causal information flow. A DSSM

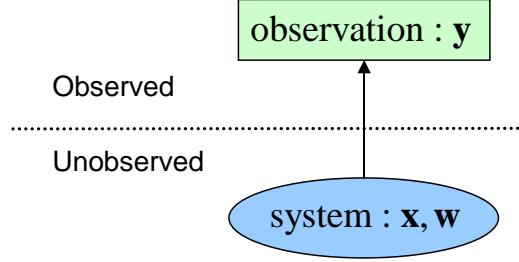


Figure 1.2: Schematic diagram of probabilistic inference: Given \mathbf{y} (noisy observation), what can we deduce/infer about \mathbf{x} (system state) and/or \mathbf{w} (system parameters) ?

can thus be interpreted as dynamic Bayesian network (DBN) with directed edges connecting the (hidden) states, where the edges directly model the temporal flow of information with the implied causality constraints. The first order Markov and conditional observation independence of the DSSMs we consider in this thesis, is modeled by the specific graphical structure and relationship of the directed edges and nodes as illustrated by the directed acyclic graph (DAG) in Figure 1.3. In general, the hidden state-space modeled by DSSMs are continuous: That is, even though the time-progression from one state to the next is discrete, the actual state variable evolves smoothly along some manifold (as determined by the process model) in state space. The number of unique state realizations are thus infinite. DSSMs can however also be used to model a finite number of uniquely indexed states. For this specific form of the state-space, the DSSM is known as a *hidden Markov model* (HMM) [14]. HMMs can further be divided into two groups: Those whose finite set of states are still continuously valued and those where the state variables are themselves discrete. The latter group are closely related to the notion of *vector quantization*, where the state variable values act as discrete indexes into state value look-up tables. For a good overview of the application of Bayesian networks to linear Gaussian systems, see the tutorial by Roweis and Ghahramani [167].

The problem statement of sequential probabilistic inference in the DSSM framework (as discussed above) can now be framed as follows: How do we optimally estimate the hidden system variables in a recursive fashion as incomplete and noisy observations becomes

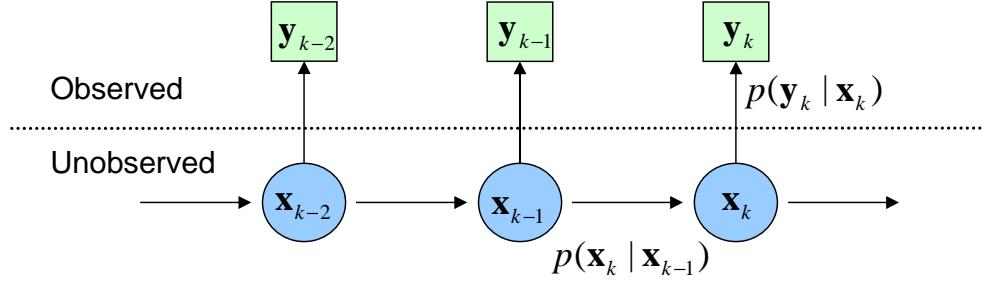


Figure 1.3: Graphical model of a probabilistic dynamic state-space model. This representation is also known as a *directed acyclic graph* (DAG) in the graph theory field.

available online? This issue lies at the heart of numerous real world applications¹ in a variety of fields such as engineering, bio-informatics, environmental modeling, statistics, finance & econometrics and machine learning to name but a few.

In a Bayesian framework, the posterior filtering density

$$p(\mathbf{x}_k | \mathbf{y}_{1:k}) \quad (1.3)$$

of the state given all the observations

$$\mathbf{y}_{1:k} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_k\} , \quad (1.4)$$

constitutes the complete solution to the sequential probabilistic inference problem, and allows us to calculate any "optimal" estimate of the state, such as the *conditional mean*

$$\hat{\mathbf{x}}_k = E[\mathbf{x}_k | \mathbf{y}_{1:k}] = \int \mathbf{x}_k p(\mathbf{x}_k | \mathbf{y}_{1:k}) d\mathbf{x}_k . \quad (1.5)$$

The problem statement can thus be reformulated as: *How do we recursively compute the posterior density as new observations arrive online?*

¹A few examples: Inertial navigation, bearings-only tracking, navigational map building, speech enhancement, global optimization of source-filter models for speech coding, financial time-series prediction, neural network training, environmental modeling with partial or missing data, etc.

1.4 Optimal Solution: Recursive Bayesian Estimation

The optimal method to recursively update the posterior density as new observations arrive is given by the *recursive Bayesian estimation* algorithm. By making use of Bayes rule and the dynamic state-space model of the system, the posterior density can be expanded and factored into the following recursive update form,

$$p(\mathbf{x}_k | \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_{1:k} | \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{y}_{1:k-1})} \quad (1.6)$$

$$= \frac{p(\mathbf{y}_k, \mathbf{y}_{1:k-1} | \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{y}_k, \mathbf{y}_{1:k-1})} \quad (1.7)$$

$$= \frac{p(\mathbf{y}_k | \mathbf{y}_{1:k-1}, \mathbf{x}_k) p(\mathbf{y}_{1:k-1} | \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{y}_k | \mathbf{y}_{1:k-1}) p(\mathbf{y}_{1:k-1})} \quad (1.8)$$

$$= \frac{p(\mathbf{y}_k | \mathbf{y}_{1:k-1}, \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{y}_{1:k-1}) p(\mathbf{y}_{1:k-1}) p(\mathbf{x}_k)}{p(\mathbf{y}_k | \mathbf{y}_{1:k-1}) p(\mathbf{y}_{1:k-1}) p(\mathbf{x}_k)} \quad (1.9)$$

$$= \frac{p(\mathbf{y}_k | \mathbf{y}_{1:k-1}, \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k | \mathbf{y}_{1:k-1})} \quad (1.10)$$

$$= \frac{p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k | \mathbf{y}_{1:k-1})} , \quad (1.11)$$

where we again made use of Bayes rule in going from Equation 1.8 to 1.9, and the conditional independence of the observation given the state in going from Equation 1.10 to 1.11.

In order to gain insight into the Bayesian recursion, let's look at the part-by-part deconstruction of Equation 1.11: The posterior at time $k - 1$, $p(\mathbf{x}_{k-1} | \mathbf{y}_{1:k-1})$, is first projected forward in time in order to calculate the *prior* at time k . This is done using the probabilistic process model, i.e.

$$p(\mathbf{x}_k | \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} | \mathbf{y}_{1:k-1}) d\mathbf{x}_{k-1} . \quad (1.12)$$

Next, the latest noisy measurement is incorporated using the observation likelihood to generate the updated posterior

$$p(\mathbf{x}_k | \mathbf{y}_{1:k}) = C p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{y}_{1:k-1}). \quad (1.13)$$

The normalizing factor is given by

$$C = \left(\int p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{y}_{1:k-1}) d\mathbf{x}_k \right)^{-1}, \quad (1.14)$$

and the *state transition prior* and *observation likelihood densities* are given by

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}) = \int \delta(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k; \mathbf{w})) p(\mathbf{v}_k) d\mathbf{v}_k \quad (1.15)$$

and

$$p(\mathbf{y}_k | \mathbf{x}_k) = \int \delta(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{n}_k; \mathbf{w})) p(\mathbf{n}_k) d\mathbf{n}_k, \quad (1.16)$$

where $\delta(\cdot)$ is the Dirac-delta function.

Although this is the optimal recursive solution, the multi-dimensional integrals (Equations 1.12, 1.14, 1.15 and 1.16) are usually only tractable for *linear, Gaussian* systems in which case the closed-form recursive solution is given by the well known *Kalman filter* [102]. For most general real-world (nonlinear, non-Gaussian) systems however, the multi-dimensional integrals are intractable and approximate solutions must be used.

1.5 Approximate Solutions

Numerous approximate solutions to the recursive Bayesian estimation problem have been proposed over the last couple of decades in a variety of fields. These methods can be loosely grouped into the following four main categories², some of which will be elaborated on in more detail later:

- **Gaussian approximate methods:** These methods all model the pertinent densities in the Bayesian recursion by Gaussian distributions, under the assumption that a consistent minimum variance estimator (of the posterior state density) can be realized through the recursive propagation and updating of only the first and second

²This list is not intended to be complete (exhaustive), but rather aims to reflect the broader categories and some of the representative or seminal works in those areas. Many excellent reviews and tutorials of the larger recursive Bayesian estimation field and optimal filtering can be found in the literature [62, 4, 46, 92, 93, 65]. These publications contain much more exhaustive taxonomies of the different approaches that fall into the broad categories presented here. For the sake of clarity and brevity, we do not intend to duplicate such a complete listing and exposition here.

order moments of the true densities.

- *Kalman filter* [102]: The celebrated Kalman filter (KF) is the optimal closed-form solution for linear, Gaussian DSSMs.
- *Extended Kalman filter* [90]: The EKF applies the Kalman filter framework to nonlinear Gaussian systems, by first linearizing the DSSM using a first-order truncated Taylor series expansion around the current estimates.
- *Gaussian sum filters* [3]: The GSF approximates the posterior by a finite linear mixture of Gaussian densities, i.e., a Gaussian mixture model (GMM) approximation. Each Gaussian component density of the GMM is propagated using a separate parallel running KF/EKF.
- **Direct numerical integration methods:** These methods, also known as *grid-based filters* (GBF), approximate the optimal Bayesian recursion integrals with large but finite sums over a uniform N-dimensional grid that tiles the complete state-space in the “area of interest”. For even moderately high dimensional state-spaces the computational complexity quickly becomes prohibitively large, which all but preclude any practical use of these filters. For more detail, see [155].
- **Sequential Monte-Carlo methods [45]:** SMC methods make no explicit assumption about the form of the posterior density. They can be used for inference and learning in any general, nonlinear non-Gaussian DSSMs. These methods, like the grid-based filters, approximates the Bayesian integrals, with finite sums. Unlike GBFs however, the summation is done with sequential importance sampling on an *adaptive* “stochastic grid”. This grid, as defined by a set of weighted samples drawn from a proposal distribution that approximates the true posterior, is concentrated in high likelihood areas of the state-space (See Figure 1.4). This approach results in huge computational savings, allowing for the implementation of practical algorithms called *particle filters* [44, 118, 68, 87]. See Section 1.5.2 for more detail.
- **Variational Bayesian methods [11, 92]:** Variational Bayesian methods approximates the true posterior distribution with a tractable approximate form. A lower

bound on the likelihood of the posterior is then maximized with respect to the free parameters of this tractable approximation, through the use of *Jensen's inequality* and variational calculus [65]. Recursive adaptation of this approach to DSSMs forms the basis of numerous inference algorithms, such as the *Bayesian mixture of factor analyzers* [64] to name but one. Variational Bayesian methods is a relatively young, but promising newcomer to the field of approximate Bayesian inference. Nonetheless, these methods fell outside the scope of the research questions addressed in this thesis.

All of the approximate methods discussed above, make some form of simplifying assumption regarding either the form of the probability densities in question, the nature of the underlying system dynamics and the desired form of the resulting estimator. This is done in order to allow for tractable and practically implementable algorithms. The accuracy and validity of the simplifying assumptions will strongly depend on the specific nature of the inference problem at hand, making certain approximate solutions more attractive than others. The work presented in this thesis, focused primarily on the most popular and widely used of these approaches, namely. *Gaussian approximate methods* and *sequential Monte Carlo methods* (SMC) as well as hybridizations of the two.

We will now give a brief introduction to these two areas, which will be covered in more detail in subsequent chapters. For brevity's sake, a certain amount of insightful but rather lengthy introductory material on both Gaussian approximations and sequential Monte Carlo methods are not presented here, but rather in Chapters 2 and 6 respectively. This includes an in-depth discussion of the extended Kalman filter (EKF) and why it is flawed, and important issue which is addressed in this thesis. We refer the reader to those sections for more background material if the following discussion (Sections 1.5.1 and 1.5.2) are to deemed too concise.

1.5.1 Gaussian Approximate Methods

Due in part to their relative ease of implementation and modest computational cost, the group of *Gaussian approximate solutions* has received most attention for over the past 40 years. Under the assumption that the underlying DSSM is linear and all of the probability

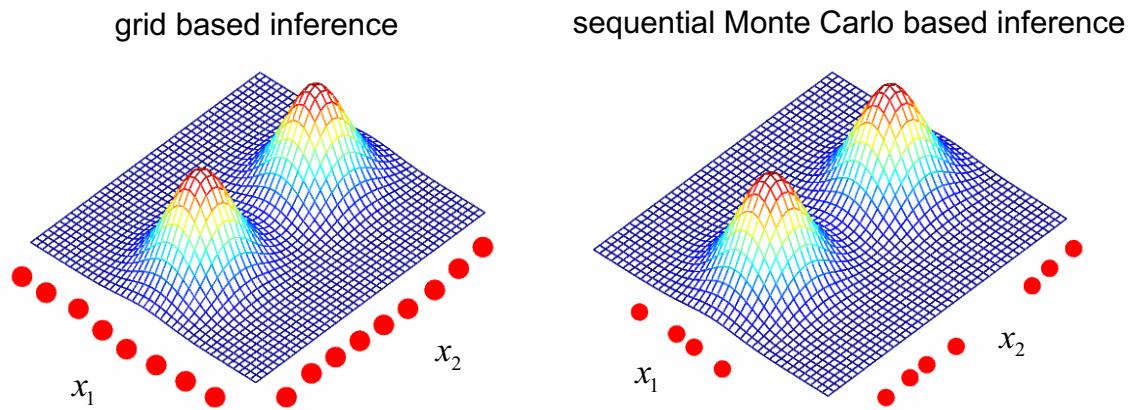


Figure 1.4: Difference between grid-based filters (left) and sequential Monte Carlo based filters (right). Both approaches approximates integrals over complex distributions by finite sums over a grid in the state-space. However, the grid-based approach uses an uniform grid that covers a large volume of the state-space, resulting in huge computational costs. The sequential Monte Carlo approach on the other hand, concentrates its grid in areas of increased likelihood, i.e., peaks in probability density, thereby significantly reducing the computational cost without sacrificing accuracy. This figure shows a 2D example with a non-Gaussian probability distribution. The x_1 and x_2 coordinates of the summation grid for both approaches are indicated by location of the red dots on their respective axes. Notice how the SMC approach concentrates the grid in areas of increased probability (peaks of distribution) that constitutes the bulk of the probability mass of the whole distribution.

densities are Gaussian, the celebrated *Kalman filter* [102] is the optimal (and exact) solution to the recursive Bayesian estimation problem. If these assumptions hold, the achieved solution is optimal in the minimum mean-square-error (MMSE) sense, the maximum likelihood (ML) sense, the maximum a posteriori (MAP) sense and asymptotically achieves the Cramer-Rao lower bound [104]. However, Kalman's original derivation of the Kalman filter (see Appendix A, Section A.2 for a full derivation) did not require the underlying system equations to be linear or the probability densities to be Gaussian. The only assumptions made are that consistent estimates of the system random variables (RVs) can be maintained by propagating only their *first and second order moments* (means and covariances), that the estimator itself be linear, and that predictions of the state and of the system observations can be calculated. These predictions are optimally calculated by taking the expected value of Equations 1.1 and 1.2. It turns out that these expectations can in general only be calculated exactly for *linear Gaussian* random variables. This does not however disallow the application of the Kalman framework to nonlinear systems. It just requires further *approximations* to be made. One such approximation is the linearization of the DSSM through the use of a first order truncated Taylor series expansion around the current estimate of the system state. This algorithm is known as the *extended Kalman filter* (EKF) [90, 4, 62].

Of all the approximate solutions listed in the previous section, the EKF has probably had the most widespread use in nonlinear estimation and inference over the last 30 years. It has been applied successfully³ to problems in all areas of probabilistic inference, including state estimation (SE), parameter estimation (PE) (also known as machine learning) and dual estimation (DE) [143, 76, 62]. Here are but a few real-world examples of where the EKF is used for these purposes:

- *State estimation*: The EKF has become a standard component part of most commercially sold *integrated navigation systems* (INS) which are used to generate navigational state solutions (position, attitude and velocity) for a variety of aircraft (manned and unmanned) [142, 84, 35, 185]. These units typically use an EKF to

³Some would say that the EKF has become the *de facto* standard for approximate nonlinear estimation.

fuse observational data coming from a variety of avionics sensors such as a GPS, IMU and magnetic compass into an estimate of the vehicles navigational state. Historically, one of the first successful real-world applications of the EKF for this purpose was in the navigation computer of the Apollo moon mission [80].

- *Parameter estimation:* Puskorius and Feldkamp from Ford Research Laboratories have applied the EKF algorithm to the problem of training of recurrent neural networks for real-world automotive engineering problems such as engine misfire detection, drop-out rejection, sensor catalyst modeling and on-vehicle idle speed control [160]. The result of this work was a custom VLSI ASIC based on a recurrent neural network framework [184].
- *Dual estimation:* Nelson successfully applied the EKF in a dual estimation framework to the problem of single channel speech enhancement [143]. For this application, the EKF is used to enhance the quality of human speech that has been corrupted by additive noise. This is done by estimating not only the underlying clean speech sequence, but also the parameters of the speech models that are used to clean up the speech. The only input to the system being the noisy speech itself [145, 144].

Even newer, more powerful inference frameworks such as *sequential Monte Carlo methods* (see Section 1.5.2) often use extended Kalman filters as subcomponents [44]. Unfortunately, the EKF is based on a suboptimal implementation of the recursive Bayesian estimation framework applied to Gaussian random variables. As we show in Chapter 2, the simple “first order Taylor series linearization” employed by the EKF, ignores the fact that the prior and predicted system state variables ($\hat{\mathbf{x}}_{k-1}$ and $\hat{\mathbf{x}}_k^-$) are, in fact, random variables themselves. This failure to account for the “probabilistic spread” of the state variables seriously affect the *accuracy* of the posterior predictions as well as the final state estimates generated by the filter. Furthermore, this often leads to *divergence* of the filter itself, where the filter fails to generate *consistent* estimates of the estimation error covariance, causing the filter to “trust” its own estimates more than is warranted by the true underlying state-space evolution and observation sequence. These two *issues* not only affect the EKF, but also any inference system that is based on the EKF or that uses the EKF as a component

part.

These shortcomings of the EKF have in recent years lead to the independent development of a number of closely related Gaussian approximate *derivativeless* filters, all based on novel deterministic sampling methods for the propagation of Gaussian random variables through nonlinear systems. In the mid to late nineties, Julier and Uhlmann [94, 99, 100, 96] introduced the *Unscented Kalman Filter* (UKF) in the general context of state estimation for automatic control. The UKF makes explicit use of the *scaled unscented transformation* (SUT) [97] in the calculation of the optimal terms in the Gaussian approximate Bayesian update (see Chapter 3), forgoing the need for an analytical Taylor series linearization as used in the EKF. The scaled unscented transformation only requires functional evaluations of the true nonlinear DSSM, which allows it to be used on non-smooth, non-analytical systems. The use of the EKF for such systems are impossible. Julier showed how, for the same computational cost, the UKF consistently outperforms the EKF in terms of state estimation accuracy and estimate consistency [95].

Independently from Julier and Uhlman's work, two different groups published a closely related algorithm in the late nineties. This filter, based on *Stirling's interpolations formula* [182], is closely related to the UKF through the use of a similar derivativeless deterministic sampling approach to propagate the Gaussian statistics (means and covariances) through nonlinear DSSMs. Ito and Xiong [88] called their version of this algorithm the *central difference filter* (CDF) and Norgaard et al [148] called theirs the *divided difference filter* (DDF). Although these two approaches were developed independently and published at the same time, they are essentially the same algorithm and we use the name *central difference Kalman filter* (CDKF) to refer to it. It turns out that the CDKF has, for all practical purposes, the same estimation performance and ease of implementation, i.e., no analytical derivatives, only functional evaluations, etc., as the UKF [147].

It turns out that the different derivativeless techniques used for Gaussian approximate random variable propagation, by both the UKF and CDKF algorithms, can be viewed as different implementational realizations of the same general deterministic sampling framework we call the *sigma-point approach*. This allows us to group all Gaussian approximate solutions that make use of this framework into a larger loosely knit family of algorithms

called *sigma-point Kalman filters (SPKFs)* [188]. Within this larger family of filters, we have since derived numerically more efficient and stable versions of both the CDKF and the UKF , called the *square-root CDKF* (SR-CDKF) and the *square-root UKF* (SR-UKF) [191, 192]. As we will show later, there are interesting parallels between the sigma-point approach (as used in all SPKFs) and a statistical technique called *weighted statistical linear regression* (also known as *stochastic linearization* [62, 114]). This partial alternative interpretation of the sigma-point approach allows for further useful unifying insights into why the SPKF is expected to perform better and more robustly than the EKF for Gaussian approximate probabilistic inference problems.

The original publications on the UKF and CDKF focused primarily on state-estimation for automatic control. We have since further expanded the use of SPKFs into the larger full domain of probabilistic inference, specifically with applications to *state estimation*, *parameter estimation* and *dual estimation*. These effort form one of the main focus areas of the work and original contribution presented in this thesis and is presented in full in Chapter 3.

1.5.2 Sequential Monte Carlo methods

More recently, another group of approximate solutions known as *Sequential Monte Carlo Methods* (SMC) or *particle filters*, have gained significant interest not only in the theoretical/research community, but also as viable solutions to real-world problems [45]. These methods, although more computationally expensive⁴ than Gaussian approximations, are the least restrictive with respect to the assumptions they make and can be effectively applied to general nonlinear, non-Gaussian problems. Particle filters represents the distribution of all pertinent random variables in the Bayesian recursion by empirical point mass approximations and then recursively update them using *sequential importance sampling and resampling* [46].

Many researchers in the statistical and signal processing communities have, almost simultaneously, proposed several variations of particle filtering algorithms. As pointed

⁴With the increasing availability of cheap but extremely powerful common-of-the-shelf (COTS) computational resources (general purpose CPUs, DSPs, etc) these methods are quickly becoming viable (practical) solutions for hard (nonlinear, non-Gaussian) real-world estimation problems.

out in [119], basic Monte Carlo methods, based on sequential importance sampling, has already been introduced in the physics and statistics literature in the fifties [69, 165]. These methods were also introduced in the automatic control field in the late sixties [71], and further explored during the seventies by various researchers [70, 208, 1]. However, all these earlier implementations were based on plain sequential importance sampling, which, as we will see later (Chapter 6), degenerates over time. The major contribution toward allowing this class of algorithm to be of any practical use was the inclusion of a resampling stage in the early nineties [68]. Since then many new improvements have been proposed [46]. One of these improvements uses an extended Kalman filter as a core subcomponent [44, 39]. Although this was done in an attempt to further mitigate the sample depletion problem experienced by all importance sampling based approaches, it made the resulting hybrid filter susceptible to the same well-known issues that plague the EKF. Since sigma-point Kalman filter attempts to improve on the EKF for application to Gaussian approximate nonlinear estimation, it stands to believe that they could be very useful as a subcomponent in hybrid SMC filters for nonlinear, non-Gaussian inference problems. It is in this area that another main focus of this thesis lies, i.e. *the extension (improvement) of the standard particle filter framework through the hybridization with Sigma-Point Kalman filters*. This is covered in full detail in Chapter 6.

1.6 Research Objectives and Work Overview

Over the last thirty years the extended Kalman filter has become a standard technique within the field of probabilistic inference, used in numerous diverse estimation algorithms and related applications. One of the reasons for its wide spread use has been the clear understanding of the underlying theory of the EKF and how that relates to different aspects of the inference problem. This insight has been enhanced by unifying studies toward the relationship between the EKF and other related algorithms [170, 12, 13], allowing for the improvement of numerous existing algorithms [90, 39, 44] and the development of new ones [157, 158, 159, 66, 76].

As discussed above, the unscented Kalman filter (UKF) and the central difference

Kalman filter (CDKF) were introduced as viable and more accurate alternatives to the EKF within the framework of state estimation. Like most new algorithms, these methods were not widely known or understood and their application has been limited. The goal of this dissertation research was an attempt to extend these differently motivated and derived algorithms into a common family of filters, called *sigma-point Kalman filters*, and expand their use to other areas of probabilistic inference, such as *parameter* and *dual estimation* as well as *sequential Monte Carlo methods*. In doing so we hope to extend the theoretical understanding of SPKF based techniques (as applied to probabilistic inference and machine learning), developed new novel algorithmic structures based on the SPKF and apply these methods to a variety of interesting and representative inference problems.

1.6.1 Summary of research objectives

In order to address the larger research goal, the following five main objectives were identified:

1. Combine different derivativeless, deterministic sampling based Gaussian approximate filters into a common family of algorithms called **sigma-point Kalman filters (SPKF)**.
2. Derive new **numerically efficient** and **stable square-root versions** of existing SPKFs.
3. Extend application of SPKFs to larger probabilistic inference and machine learning field:
 - (a) **State estimation**
 - Verification and extension of existing state estimation claims in literature.
 - Compare to EKF based state estimation.
 - (b) **Parameter estimation**
 - Derive efficient algorithmic SPKF forms for parameter estimation.

- Experimentally verify SPKF parameter estimation performance on benchmark problems (neural network training, parametric model fitting, etc.) and compare to EKF performance.
- Investigate the extension of SPKF parameter estimators to non-MSE cost functions.
- Relate (theoretically) SPKF based parameter estimation methods to other existing second order nonlinear optimization methods.

(c) **Dual estimation**

- Implement different algorithmic forms for SPKF based dual estimation: *dual SPKF, joint SPKF and SPKF based EM*.
- Comparative experimental verification of performance relative to EKF based methods.

4. SPKF application to real-world inference problem: *UAV Autonomy*

Two large ongoing research projects within our group at OGI are the DARPA sponsored “*Model-Relative Control of Autonomous Vehicles*” [38, 105] and the ONR sponsored “*Sigma-Point Kalman Filter Based Sensor Integration, Estimation and System Identification for Enhanced UAV Situational Awareness & Control*” [149, 203] projects. Both these overlapping programs focuses on the autonomous control of an *unmanned aerial vehicle* (UAV)⁵. The core component of such a system is a high-performance digital computer based *guidance, navigation & control* (GNC) unit as depicted in Figure 1.5. The main subsystems of this GNC unit is a control system and a guidance & navigation system (GNS). The GNS takes noisy avionics sensor (GPS, IMU, magnetic compass, altimeter, etc.) measurements as input, and then fuse them in a probabilistic sense with predictions from a vehicle dynamics model in order to calculate optimal vehicle navigation state solutions. These state estimates together with desired flight trajectories are then fed to the control system that computes some form of optimal control law to drive the flight surface actuators of the vehicle. The

⁵In our case the UAV is an autonomous rotorcraft (helicopter). See Chapter 5 for full detail.

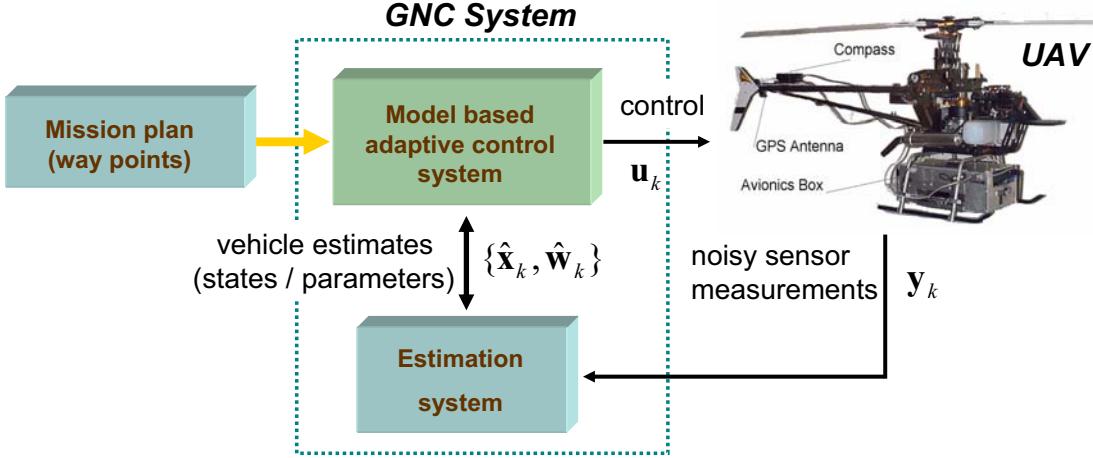


Figure 1.5: Unmanned aerial vehicle (UAV) guidance, navigation and control (GNC) system.

current state-of-the-art probabilistic inference system used for such UAV guidance and navigation systems are usually EKF based [56, 58]. Our aim is to replace the EKF in the UAV-GNC system by a SPKF with specific focus on:

- Improved six-degrees-of-freedom (6DOF) state estimation accuracy (relative to EKF).
- SPKF based compensation for GPS latency.
- Evaluation of improved control envelope due to use of better state estimator.
- SPKF based parameter estimation: Track certain UAV model parameters such as mass, moments of inertia, etc.
- SPKF based dual estimation: Simultaneously estimate 6DOF state, as well as secondary set of hidden dynamic model states (blade flapping angles, etc.) and attempt to track a subset of vehicle model parameters (mass, etc.)

This real-world application will bring together numerous theoretical and algorithmic aspects and serve as a proof of concept for integrating the different separately developed SPKF technologies.

5. Development of **hybrid SMC / SPKF algorithms** for improved general, nonlinear, non-Gaussian probabilistic inference:

As indicated in Section 1.5.2 and discussed in full detail in Chapter 6, we aimed to hybridize existing particle filters with sigma-point Kalman filters in an attempt to mitigate the well known *sample depletion problem* [46]. Previous attempts to accomplish the same goal through the use of EKF hybridization has had only partial success due to the inherent problems experienced with EKF inaccuracy, divergence, etc. We intend to improve/extend this approach through the replacement of the EKF in all such hybrid methods by the SPKF. Specific goals include:

- Hybrid SMC/SPKF algorithm development from well founded theoretical arguments.
- Application of new hybrid algorithm to nonlinear, non-Gaussian inference problems for experimental verification and comparison to existing competitive methods.

By addressing these five main objectives, we hope to contribute to taking the SPKF out of its infancy and making it better understood and more accessible to different areas within the probabilistic inference field.

1.7 Contributions of this Work

Most of the research objectives as stated in the previous section were successfully completed during the course of this thesis research. During the pursuit and completion of these objectives, a number of ancillary issues came to light. These were addressed as needed within the larger scope of the research. In summary, the following concrete and substantial contributions to the body of knowledge regarding SPKFs and their application to probabilistic inference and machine learning were made:

1. Sigma-point Kalman filters:

- Combined different derivativeless Kalman filters (UKF, CDF & DDF) into a single implementational/algorithmic framework : SPKFs
- Derived numerically efficient and stable square-root versions of all SPKF algorithms: SR-UKF & SR-CDKF

- Extended application of SPKFs to larger probabilistic inference and machine learning field:

– **State estimation:**

- * Verified literature claims of consistently better performance than EKF on a variety of nonlinear state estimation problems.
- * Successfully applied SPKFs to new state estimation problems such as nonlinear time-series prediction.
- * Implemented and verified a novel SPKF based smoother for nonlinear time-series estimation. This smoother also forms the core component of the *E-step* of a future SPKF based full EM (expectation maximization) algorithm.

– **Parameter estimation**

- * Derived efficient $\mathcal{O}(n^2)$ algorithmic SPKF forms for parameter estimation.
- * Experimentally verified SPKF parameter estimation performance on a variety of benchmark problems (neural network training, parametric model fitting, etc.) and compared to EKF performance.
- * Extended SPKF based parameter estimation framework to minimize general non-MSE cost functions.
- * We show the relationship between the SPKF and other second order nonlinear optimization methods. This extends the theoretical understanding of the SPKF for parameter estimation. The SPKF is cast into a general online (stochastic) *adaptive modified Newton method* framework that minimizes an instantaneous nonlinear least squares cost function.

– **Dual estimation**

- * Derived and implemented two different algorithmic forms for SPKF based dual estimation:
 - *dual SPKF* : iterative state/parameter estimation; less optimal; lower

computational cost

- *joint SPKF* : direct joint MAP estimation of states and parameters; closer to optimal solution; higher computational cost.
- * Experimentally verified performance and compared to EKF based methods on a number of dual estimation problems.

2. Real-world application of SPKF framework: UAV Autonomy

- A SPKF based UAV guidance & navigation system for sensor integration and probabilistic inference (state, parameter & dual estimation) was successfully implemented (first of its kind to authors knowledge).
- SPKF system incorporates a novel method to deal with sensor latency (in GPS). The SPKF framework allows for an elegant (using sigma-point approach) way of optimally fusing time-delayed measurements with smoothed state estimates. This allows for increased estimation performance of whole inference system.
- System was extensively tested against state-of-the-art EKF based guidance & navigation system. The SPKF system consistently outperforms EKF system in terms of: estimation accuracy and resulting control cost.
- We demonstrated the enhanced robustness of our SPKF based system, compared to similar EKF based systems, through synthetic IMU degradation experiments. This has direct implications on the cost/performance tradeoff for any SPKF based GNC system. Since the IMU is typically the most expensive component of any GNC system, our system allows for greatly enhanced performance at the same financial cost level, or conversely, we can achieve equal performance (to EKF systems) at a greatly reduced financial cost, i.e., through the use of a cheaper, lower quality IMU.
- Modular form of SPKF system for 6DOF state estimation allows for easy extension to other vehicles (UAVs, UUVs, UGVs, etc.) due to use of kinematic vehicle process model and high accuracy IMU sensor.

- Prototype dual estimation framework successfully demonstrated. Capable of tracking time-varying system parameters such as vehicle mass (due to fuel consumption and payload transients).

3. Sequential Monte Carlo methods : SPKF / particle filter hybrids

- Particle filter sample depletion problem address by derivation/implementation of two new hybrid algorithms that use an adaptive bank of SPKFs for *proposal distribution* generation.
 - *Sigma-point particle filter* (SPPF)
 - *Gaussian-mixture sigma-point particle filter* (GMSPPF)
- We show how these new algorithms closer match the requirements for the theoretically optimal proposal distribution that are guaranteed to reduce the variance of the resulting importance weights.
- Algorithms were experimentally verified on a number of representative non-linear, non-Gaussian inference problems, including *human face tracking*⁶, *non-Gaussian nonlinear time-series estimation*, *global robot localization* and *financial options pricing*.

4. Released public domain software library for recursive Bayesian estimation: *ReBEL*

- During the course of this dissertation research, the underlying software implementation of the different inference algorithms went through a number of evolutionary iterations. We eventually standardized our implementation on a modular framework that decouples the application (inference problem) layer from the estimation (inference filter) layer. This resulted in a Matlab toolkit, called ***ReBEL*** (Recursive Bayesian Estimation Library), that contains all of

⁶The actual implementation of our SPPF algorithm for human face tracking was done by Rui at Microsoft Research [169]. This serves as an external verification of our algorithm in a real-world inference system. Subsequently, our SPPF algorithm has received wide ranging interest in the literature [196, 117, 36] and has been applied successfully to numerous other real-world inference problems [86, 117].

the different existing as well as newly derived SPKF and SPKF/SMC hybrid algorithms. The toolkit allows for a high-level description of the problem domain through the definition of the relevant DSSM (process model, observation model, noise models, etc.) after which any of the different inference algorithms can then be applied for either state-, parameter- or dual estimation.

- The toolkit also include a large number of demonstration/example scripts that duplicate a large amount of the experimental results presented in this dissertation. This allows for easy external verification and duplication of our published results.
- The toolkit is freely available for academic research and has to date⁷ been downloaded by more than 1000 different researchers and research groups with a wide variety of intended uses.
- ReBEL has been licensed by a number of commercial companies through OHSU's Technology and Research Collaborations (TRC) office, resulting in the award of an OHSU *Commercialization Award* to the author.

1.8 Thesis Outline

The remainder of the dissertation is organized as follows:

1.8.1 Main thesis chapters

Chapter 2 gives an in-depth discussion of optimal Gaussian approximate recursive Bayesian estimation, introduces the extended Kalman filter and shows why the EKF is in fact a highly suboptimal (flawed) approximate solution. This chapter covers much of the introductory motivation of why a better solution than the EKF is needed for Gaussian approximate nonlinear estimation.

Chapter 3 is one of the core chapters in this dissertation and covers the algorithmic development of the *sigma-point Kalman filter* in detail. The two main SPKF variants, the

⁷The initial version of **ReBEL** was released in May 2002, followed by a number of bug-fix releases as well as major updates. By the time of this publication, January 2004, more than 1000 unique downloads of the toolkit for academic use has taken place.

UKF and CDKF, are introduced and shown to be different implementational variations of a common derivativeless Kalman filter framework, which in turn is based on a deterministic sampling technique called the sigma-point approach. We derive numerically efficient and stable square-root versions of the SPKF as well as inference type specific forms. After the different SPKF algorithms are introduced, we cover in detail (using numerous experimental examples) how they are applied to the three domains of probabilistic inference, specifically *state-*, *parameter-* and *dual-estimation*. These experiments also verify the superiority of the SPKF over the EKF for all classes of estimation problems.

In **Chapter 4** a number of theoretical aspects of the sigma-point Kalman filter are derived and analyzed in order to gain further insight and relate it to other algorithms. Specific attention is given to the *weighted statistical linear regression* interpretation of the sigma-point approach that is employed by all SPKF forms. We investigate the theoretical accuracy of the sigma-point approach (compared to the optimal solution) as well as the relationship between SPKF based parameter estimation and other 2nd order nonlinear optimization methods. At the end of this chapter the most salient (and important) characteristics of the SPKF are summarized and contrasted with those of the EKF.

Chapter 5 focuses specifically on the application of the SPKF to the *UAV Autonomy* problem. This large-scale application is covered in depth, giving detail about the vehicle itself (dynamic models, etc.), the experimental setup that was used (high-fidelity simulator) and the numerous experiments that was done. Results are given for a number of state estimation (open and closed loop) experiments as well as parameter and dual estimation experiments.

Chapter 6 focuses on the derivation, implementation and experimental verification of the two hybrid sequential Monte Carlo / SPKF algorithms we developed, the *sigma-point particle filter* and the *Gaussian-mixture sigma-point particle filter*. A thorough introduction to sequential Monte Carlo methods (particle filters) is first presented, showing the relationship to general nonlinear, non-Gaussian approximate Bayesian inference. After that the *sample depletion problem* is introduced which serves as motivation for the development of the above mentioned hybrid approaches. As with the chapter on the SPKF, we

verify these new algorithms on numerous representative inference problems including non-linear non-Gaussian time-series prediction, financial options pricing, human face tracking and global robot localization.

In **Chapter 7**, the major results of this work is discussed and summarized, conclusions are drawn and possible directions for future research are indicated.

1.8.2 Appendices

Some of the material that forms part of the secondary contributions made by this work is relegated to the Appendices section in order to maintain the presentational flow of this document. These include **Appendix C** which covers the development and use of the ReBEL Toolkit.

1.9 Publications

A large majority of the work contained in this thesis has already been published in the peer-reviewed literature and presented at numerous conferences and workshops [199, 200, 204, 190, 77, 189, 191, 192, 193, 188]. Here is a list of those publications:

- VAN DER MERWE, R. Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic-State Space Models. In *Proc. of Workshop on Advances in Machine Learning* (Montreal, June 2003).
- VAN DER MERWE, R., AND WAN, E. A. Gaussian Mixture Sigma-Point Particle Filters for Sequential Probabilistic Inference in Dynamic State-Space Models. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Hong Kong, April 2003), IEEE.
- VAN DER MERWE, R., AND WAN, E. A. The Square-Root Unscented Kalman Filter for State- and Parameter-Estimation. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Salt Lake City, May 2001), IEEE.
- VAN DER MERWE, R., AND WAN, E. A. Efficient Derivative-Free Kalman Filters for Online Learning. In *Proc. of ESANN* (Bruges, Belgium, April 2001).
- WAN, E. A., AND VAN DER MERWE, R. *Kalman Filtering and Neural Networks*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. Wiley, 2001, ch. 7 - The Unscented Kalman Filter, pp. 221-280.

- VAN DER MERWE, R., DE FREITAS, N., DOUCET, A., AND WAN, E. A. The Unscented Particle Filter. In *Advances in Neural Information Processing Systems (NIPS) 13* (Denver, Nov 2001), pp. 584-590.
- VAN DER MERWE, R., DE FREITAS, N., DOUCET, A., AND WAN, E. A. The Unscented Particle Filter. Tech. Rep. CUED/F-INFENG/TR 380, Cambridge University Engineering Department, August 2000.
- WAN, E. A., AND VAN DER MERWE, R. The Unscented Kalman Filter for Nonlinear Estimation. In *Proc. of IEEE Symposium 2000 (AS-SPCC)* (Lake Louise, Oct 2000), IEEE.
- WAN, E. A., VAN DER MERWE, R., AND NELSON, A. T. Dual Estimation and the Unscented Transformation. In Neural Information Processing Systems (NIPS) 12 (Denver, Nov 2000), pp. 666-672.
- WAN, E. A., AND VAN DER MERWE, R. Noise-Regularized Adaptive Filtering for Speech Enhancement. In Proc. of EuroSpeech (Sep 1999).

The work presented in this thesis and the publications above, have already impacted a number of external related research efforts, resulting in further applications, refinements and publications. We briefly summarize some of the more important and significant of these in Chapter 7.

Chapter 2

Gaussian Approximate Bayesian Estimation:

The Kalman Filter Framework

2.1 Introduction

As we showed in Section 1.4, the optimal recursive Bayesian solution to the probabilistic inference problem requires the propagation of the full posterior state probability density function. This optimal solution is general enough to deal with any form of posterior density, including aspects such as multimodalities, asymmetries and discontinuities. However, since this solution does not put any restrictions on the form of the posterior density, it cannot in general be described by a finite number of parameters. Therefore, as already pointed out earlier, any practically implementable estimator must make some form of approximating assumption with regard to the shape of the posterior density and the form of the Bayesian recursion as summarized by Equations 1.6 through 1.16. This chapter, and the first half of this thesis, focuses on the group of *Gaussian approximate solutions* which are all based around the *Kalman filter framework* [102]. The most well known application of the Kalman filter framework to nonlinear inference problems is the extended Kalman filter (EKF). As already alluded to in Section 1.5, even though the EKF is one of the most widely used approximate solution for nonlinear estimation and filtering, it has serious limitations. This chapter will investigate this claim further, first by showing what the optimal Gaussian approximate linear Bayesian update, which forms the core of the Kalman filter framework, looks like, and then contrasting that with the suboptimal implementation of this approach

as used in the EKF. This exposition will serve as a justification for the need to develop a better solution, the *sigma-point Kalman filter*, (SPKF) which is covered in detail in Chapter 3.

In this chapter we will also introduce the three different *algorithmic forms* of probabilistic inference, i.e., *state-estimation*, *parameter estimation* and *dual estimation*, and show how the EKF (and hence subsequently the SPKF) is used to perform these functions for general nonlinear models. The actual experimental results of applying the SPKF to these “three legs” of the inference problem is reported in Chapter 3.

2.2 Optimal Gaussian Approximate Linear Bayesian Update

A common misconception about the Kalman framework is that it *requires* the underlying state space to be linear as well as all the probability densities to be Gaussian. This is in fact overly strict and incorrect. Kalman’s original derivation of the Kalman filter (see Appendix A, Section A.2) did not assume these conditions. The only assumptions he made were the following:

1. Consistent minimum variance estimates of the system random variables (RVs) and hence the posterior state distribution (pdf) can be calculated, by maintaining (recursively propagating and updating) only their *first and second order moments* (means and covariances).
2. The estimator (measurement update) itself is a *linear* function of the prior knowledge of the system, summarized by $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$, and the new observed information, summarized by $p(\mathbf{y}_k|\mathbf{x}_k)$. In other words, it is assumed that Equation 1.11 of the optimal Bayesian recursion can be accurately approximated by a linear function.
3. Accurate predictions of the state (using process model) and of the system observations (using observation model) can be calculated. These predictions are needed to approximate the first and second order moments of $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ and $p(\mathbf{y}_k|\mathbf{x}_k)$.

Consistency in Assumption 1 implies that the estimates of the mean and covariance of the posterior state density, i.e., $\hat{\mathbf{x}}_k$ and $\mathbf{P}_{\mathbf{x}_k}$, satisfies the following:

$$\text{trace} \left[\mathbf{P}_{\mathbf{x}_k} - E \left[(\mathbf{x}_k - \hat{\mathbf{x}}_k) (\mathbf{x}_k - \hat{\mathbf{x}}_k)^T \right] \right] \geq \mathbf{0}, \quad (2.1)$$

where $(\mathbf{x}_k - \hat{\mathbf{x}}_k)$ is called the estimation error¹. The fact that only means and covariances are maintained (per Assumption 1) is why these methods are (somewhat misleadingly) called *Gaussian approximate solutions*. In other words, the densities are not required to be Gaussian, we simply only maintain the Gaussian components (mean and covariance) of these densities in the estimator. If the assumed consistency of the estimator holds and the mean estimates are unbiased, then the resulting Gaussian approximate posterior will have significant support overlap with the true posterior.

The form of the measurement update is chosen to be linear in Assumption 2, in order to allow for a practically realizable and computationally feasible estimator that can be implemented using numerous powerful and computationally efficient linear algebra components. This assumed linear form of the measurement update, implies that the resulting Gaussian approximate estimator will be the optimal (as defined by the minimum variance criteria of Assumption 1) *linear estimator*, if the remaining assumptions are exactly satisfied. This in turn implies that we need to calculate the *predictions* in Assumption 3 in an optimal fashion.

Based on these assumptions, Kalman derived [102] the following recursive form of the optimal Gaussian approximate linear Bayesian update (Kalman update for short) of the conditional mean of the state RV, $\hat{\mathbf{x}}_k = E[\mathbf{x}_k | \mathbf{y}_{1:k}]$ and its covariance, $\mathbf{P}_{\mathbf{x}_k}$:

$$\hat{\mathbf{x}}_k = (\text{prediction of } \mathbf{x}_k) + \mathbf{K}_k (\mathbf{y}_k - (\text{prediction of } \mathbf{y}_k)) \quad (2.2)$$

$$= \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (2.3)$$

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k} \mathbf{K}_k^T. \quad (2.4)$$

¹An *efficient* estimator minimizes the magnitude of the LHS of Equation 2.1 while still satisfying the \geq constraint. A *conservative* estimate replaces the inequality with a strictly *greater than* ($>$) relationship.

While this is a *linear* recursion, we have not assumed linearity of the model. The optimal terms in this recursion are given by

$$\hat{\mathbf{x}}_k^- = E[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}, \mathbf{u}_k)] \quad (2.5)$$

$$\hat{\mathbf{y}}_k^- = E[\mathbf{h}(\mathbf{x}_k^-, \mathbf{n}_k)] \quad (2.6)$$

$$\mathbf{K}_k = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{y}_k - \hat{\mathbf{y}}_k^-)^T] E[(\mathbf{y}_k - \hat{\mathbf{y}}_k^-)(\mathbf{y}_k - \hat{\mathbf{y}}_k^-)^T]^{-1} \quad (2.7)$$

$$= \mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k}^{-1} \quad (2.8)$$

where the optimal prediction (prior mean at time k) of \mathbf{x}_k is written as $\hat{\mathbf{x}}_k^-$, and corresponds to the *expectation* (taken over the posterior distribution of the state at time $k-1$) of a *nonlinear function of the random variables* \mathbf{x}_{k-1} and \mathbf{v}_{k-1} (similar interpretation for the optimal prediction $\hat{\mathbf{y}}_k^-$, except the expectation is taken over the prior distribution of the state at time k). The optimal gain term \mathbf{K}_k is expressed as a function of the expected cross-correlation matrix (covariance matrix) of the state prediction error and the observation prediction error, and the expected auto-correlation matrix of the observation prediction error². Note that evaluation of the covariance terms also require taking *expectations of a nonlinear function of the prior state variable*.

It turns out that these expectations can in general only be calculated exactly (in an analytical sense) for a *linear* DSSM and *Gaussian* random variables. Under these (linear, Gaussian) conditions, the Kalman filter framework are in fact an exact solution of the optimal Bayesian recursion (see Appendix A, Section A.3). This is probably why the Kalman filter framework is often misconstrued as only applicable to such linear, Gaussian systems. This does not, however, disallow the application of the Kalman framework to nonlinear systems. As mentioned above, even for nonlinear, non-Gaussian systems, the Kalman filter framework is still the (minimum variance) optimal Gaussian approximate linear estimator, if the rest of the assumptions hold. This does however require further *approximations* to be made in order to practically apply this framework to nonlinear systems. Specifically, these approximations directly address how the optimal terms in the

²The error between the true observation and the predicted observation, $\tilde{\mathbf{y}}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k^-$ is called the *innovation*.

Kalman update (Equations 2.5 through 2.8) are calculated.

2.3 Nonlinear Transformations of Random Variables

As noted above, at the core of the optimal Kalman update³ we need to accurately calculate the expected mean and covariance of a random variable that undergoes a nonlinear transformation. Consider the following problem: a random variable \mathbf{x} with mean $\bar{\mathbf{x}}$ and covariance \mathbf{P}_x undergoes an arbitrary nonlinear transformation. This can be written as

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) \quad (2.9)$$

$$= \mathbf{g}(\bar{\mathbf{x}} + \boldsymbol{\delta}_{\mathbf{x}}), \quad (2.10)$$

where $\boldsymbol{\delta}_{\mathbf{x}}$ is a *zero-mean* random variable with the same covariance $\mathbf{P}_{\boldsymbol{\delta}_{\mathbf{x}}}$ as \mathbf{x} , and $\bar{\mathbf{x}}$ is the mean value of \mathbf{x} . Given this formulation, what is the mean and covariance of \mathbf{y} ?

In order to analytically calculate these quantities, we first expand $\mathbf{g}(\cdot)$ using a multi-dimensional Taylor series⁴ expansion around $\bar{\mathbf{x}}$,

$$\mathbf{y} = \mathbf{g}(\bar{\mathbf{x}}) + \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}} \mathbf{g} + \frac{1}{2!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^2 \mathbf{g} + \frac{1}{3!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^3 \mathbf{g} + \frac{1}{4!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^4 \mathbf{g} + \dots, \quad (2.11)$$

where $\mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}} \mathbf{g}$ is an operator that evaluates the total differential of $\mathbf{g}(\cdot)$ when perturbed around a nominal value $\bar{\mathbf{x}}$ by $\boldsymbol{\delta}_{\mathbf{x}}$, i.e.,

$$\mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}} \mathbf{g} \doteq \left[(\boldsymbol{\delta}_{\mathbf{x}}^T \nabla) \mathbf{g}(\mathbf{x}) \right]^T \Big|_{\mathbf{x}=\bar{\mathbf{x}}}. \quad (2.12)$$

This operator can be written and interpreted as the scalar operator

$$\mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}} = \sum_{j=1}^{N_x} \delta_{x_j} \frac{\partial}{\partial x_j} \quad (2.13)$$

which acts on $\mathbf{g}(\cdot)$ on a component-by-component basis. Using this definition, the i th term

³We use the term *optimal Kalman update* as shorthand for “optimal Gaussian approximate linear Bayesian update”.

⁴For the purpose of this analysis, we assume that \mathbf{g} is analytic across the domain of all possible values of \mathbf{x} . This requirement does not need to be true in general, but it does simplify the analysis we present here.

($i = 0, 1, 2, \dots, \infty$) in the Taylor series for $\mathbf{y} = \mathbf{g}(\mathbf{x})$ is thus given by

$$\frac{1}{i!} \mathbf{D}_{\delta_{\mathbf{x}}}^i \mathbf{g} = \frac{1}{i!} \left[\sum_{j=1}^{N_x} \delta_{\mathbf{x}_j} \frac{\partial}{\partial x_j} \right]^i \mathbf{g}(\mathbf{x}) \Big|_{\mathbf{x}=\bar{\mathbf{x}}} \quad (2.14)$$

where $\delta_{\mathbf{x}_j}$ is the j th component of $\delta_{\mathbf{x}}$, $\frac{\partial}{\partial x_j}$ is the normal partial derivative operator with respect to x_j (the j th component of \mathbf{x}), and $N_{\mathbf{x}}$ is the dimension of \mathbf{x} (i.e. $\mathbf{x} \in \mathbb{R}^{N_{\mathbf{x}}}$).

The **expected value** (mean) of \mathbf{y} , $\bar{\mathbf{y}}$, is given by the expected value of Equation 2.11, i.e.,

$$\begin{aligned} \bar{\mathbf{y}} &= E[\mathbf{g}(\bar{\mathbf{x}} + \delta_{\mathbf{x}})] \\ &= \mathbf{g}(\bar{\mathbf{x}}) + E \left[\mathbf{D}_{\delta_{\mathbf{x}}} \mathbf{g} + \frac{1}{2!} \mathbf{D}_{\delta_{\mathbf{x}}}^2 \mathbf{g} + \frac{1}{3!} \mathbf{D}_{\delta_{\mathbf{x}}}^3 \mathbf{g} + \frac{1}{4!} \mathbf{D}_{\delta_{\mathbf{x}}}^4 \mathbf{g} + \dots \right] \end{aligned} \quad (2.15)$$

where the i th term in the series is given by

$$\begin{aligned} E \left[\frac{1}{i!} \mathbf{D}_{\delta_{\mathbf{x}}}^i \mathbf{g} \right] &= \frac{1}{i!} E \left[\left(\sum_{j=1}^{N_x} \delta_{\mathbf{x}_j} \frac{\partial}{\partial x_j} \right)^i \mathbf{g}(\mathbf{x}) \Big|_{\mathbf{x}=\bar{\mathbf{x}}} \right] \\ &= \frac{1}{i!} \sum_{k=1}^{(N_x)^i} \left[m_{s(\delta_{\mathbf{x}}, k, N_{\mathbf{x}}, i)} \frac{\partial^i \mathbf{g}(\mathbf{x})}{\partial x_k^i} \Big|_{\mathbf{x}=\bar{\mathbf{x}}} \right]. \end{aligned} \quad (2.16)$$

Here $s(a, k, b, i)$ is an argument constructor function defined as

$$s(a, k, b, i) \doteq \text{pick the } k\text{th term in the full expansion of : } \left(\sum_{j=1}^b a_j \right)^i,$$

e.g.,

$$\begin{aligned} s((\partial x), 3, 2, 2) &= \text{pick the 3rd term of : } (\partial x_1 + \partial x_2)^2 \\ &= \text{pick the 3rd term of : } (\partial x_1 \partial x_1 + \partial x_1 \partial x_2 + \partial x_2 \partial x_1 + \partial x_2 \partial x_2) \\ &= \partial x_2 \partial x_1 , \end{aligned}$$

and $m_{\delta_{\mathbf{x}_1}\delta_{\mathbf{x}_2}\dots\delta_{\mathbf{x}_n}}$ is the n th order central moment of $\delta_{\mathbf{x}}$, i.e.,

$$\begin{aligned} m_{\delta_{\mathbf{x}_1}\delta_{\mathbf{x}_2}\dots\delta_{\mathbf{x}_n}} &= E[\delta_{\mathbf{x}_1}\delta_{\mathbf{x}_2}\dots\delta_{\mathbf{x}_n}] \\ &= \int (\delta_{\mathbf{x}_1}\delta_{\mathbf{x}_2}\dots\delta_{\mathbf{x}_n}) p(\mathbf{x}) d\mathbf{x}. \end{aligned}$$

It is clear from Equation 2.16 that in order to calculate the mean of \mathbf{y} accurately to the m th order (of the expanded Taylor series), we need to know all the moments of $\delta_{\mathbf{x}}$ (and hence \mathbf{x}) and the derivatives of \mathbf{g} , up to and including the m th order.

For further analysis and comparison to the EKF and subsequent sigma-point Kalman filter formulations, it is insightful to write Equation 2.15 as

$$\bar{\mathbf{y}} = \mathbf{g}(\bar{\mathbf{x}}) + \frac{1}{2}E[\mathbf{D}_{\delta_{\mathbf{x}}}^2 \mathbf{g}] + E\left[\frac{1}{4!}\mathbf{D}_{\delta_{\mathbf{x}}}^4 \mathbf{g} + \frac{1}{6!}\mathbf{D}_{\delta_{\mathbf{x}}}^6 \mathbf{g} + \dots\right] \quad (2.17)$$

where we assumed that the distribution of $\delta_{\mathbf{x}}$ (and hence \mathbf{x}) is symmetrical⁵, such that all odd-order moments are zeros. The expectation in the second order term can be written as

$$\begin{aligned} E[\mathbf{D}_{\delta_{\mathbf{x}}}^2 \mathbf{g}] &= E[\mathbf{D}_{\delta_{\mathbf{x}}}(\mathbf{D}_{\delta_{\mathbf{x}}} \mathbf{g})^T] \\ &= E[(\nabla^T \delta_{\mathbf{x}} \delta_{\mathbf{x}}^T \nabla) \mathbf{g}(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}}] . \end{aligned} \quad (2.18)$$

Using the interpretation of the $\mathbf{D}_{\delta_{\mathbf{x}}}$ operator and noting that $E[\delta_{\mathbf{x}} \delta_{\mathbf{x}}^T] = \mathbf{P}_{\mathbf{x}}$, we can rewrite Equation 2.18 as

$$E[\mathbf{D}_{\delta_{\mathbf{x}}}^2 \mathbf{g}] = (\nabla^T \mathbf{P}_{\mathbf{x}} \nabla) \mathbf{g}(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}} . \quad (2.19)$$

Substituting this result back into Equation 2.17 gives us the following convenient form of the mean expansion which we will use again in Chapter 4:

$$\bar{\mathbf{y}} = \mathbf{g}(\bar{\mathbf{x}}) + \frac{1}{2} [(\nabla^T \mathbf{P}_{\mathbf{x}} \nabla) \mathbf{g}(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}}] + E\left[\frac{1}{4!}\mathbf{D}_{\delta_{\mathbf{x}}}^4 \mathbf{g} + \frac{1}{6!}\mathbf{D}_{\delta_{\mathbf{x}}}^6 \mathbf{g} + \dots\right] \quad (2.20)$$

⁵Symmetry is a less strict requirement than Gaussianity, but does include multi-dimensional Gaussian distributions as a special case. This assumption holds for a very large group of 'real-world' occurring distributions such as student-T, Cauchy, the exponential family, etc, which are then often modeled as Gaussian distributions or finite mixtures of Gaussians.

In a similar fashion as the exposition shown above, we can calculate the true **covariance** of the posterior random variable \mathbf{y} . First, by definition the covariance of \mathbf{y} is given by

$$\mathbf{P}_{\mathbf{y}} = E \left[(\mathbf{y} - \bar{\mathbf{y}}) (\mathbf{y} - \bar{\mathbf{y}})^T \right]. \quad (2.21)$$

We can calculate $\mathbf{y} - \bar{\mathbf{y}}$ by substituting from Equations 2.11 and 2.17,

$$\begin{aligned} \mathbf{y} - \bar{\mathbf{y}} &= \mathbf{g}(\bar{\mathbf{x}} + \boldsymbol{\delta}_{\mathbf{x}}) - E[\mathbf{g}(\bar{\mathbf{x}} + \boldsymbol{\delta}_{\mathbf{x}})] \\ &= \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}} \mathbf{g} + \frac{1}{2!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^2 \mathbf{g} + \frac{1}{3!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^3 \mathbf{g} + \frac{1}{4!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^4 \mathbf{g} + \dots \\ &\quad - E \left[\frac{1}{2!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^2 \mathbf{g} + \frac{1}{4!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^4 \mathbf{g} + \frac{1}{6!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^6 \mathbf{g} + \dots \right]. \end{aligned} \quad (2.22)$$

Taking outer products and expectations (again exploiting the fact that if we assume $\boldsymbol{\delta}_{\mathbf{x}}$ to be symmetrically distributed, the odd moment terms equate to zero), we can write⁶ the covariance of \mathbf{y} as

$$\begin{aligned} \mathbf{P}_{\mathbf{y}} &= \mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T - \frac{1}{4} E \left[\mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^2 \mathbf{g} \right] E \left[\mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^2 \mathbf{g} \right]^T \\ &\quad + E \left[\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{i!j!} \mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^i \mathbf{g} \left(\mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^j \mathbf{g} \right)^T}_{\forall i,j: \text{ such that } ij>1} \right] \\ &\quad - \left(\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{(2i)!(2j)!} E \left[\mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^{2i} \mathbf{g} \right] E \left[\mathbf{D}_{\boldsymbol{\delta}_{\mathbf{x}}}^{2j} \mathbf{g} \right]^T}_{\forall i,j: \text{ such that } ij>1} \right). \end{aligned} \quad (2.23)$$

Here $\mathbf{G}_{\bar{\mathbf{x}}} = \nabla_{\mathbf{x}} \mathbf{g}(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}}$ is the Jacobian matrix of $\mathbf{g}(\mathbf{x})$ evaluated at $\mathbf{x} = \bar{\mathbf{x}}$. From this is clear that in general the m th order term in the covariance series expansion can only be calculated accurately if we know all of the derivatives of \mathbf{g} and the moments of $\boldsymbol{\delta}_{\mathbf{x}}$ up to that order. In Section 2.5.1 we give an analytic example of the above analysis applied to a simple scalar problem.

⁶See [99] for a complete derivation.

2.4 The EKF and its Flaws

As stated earlier, the Kalman filter [102] calculates the optimal terms in Equations 2.5, 2.6 and 2.8 exactly for linear DSSMs. This is a well known result for linear Gaussian systems, i.e, the linear transformation of a Gaussian random variable stays Gaussian, and is implicitly due to the fact that all of the higher order (> 1) derivatives of \mathbf{g} in Equations 2.15 and 2.23 equate to zero.

For nonlinear DSSMs however, the *extended Kalman filter* (EKF) first linearizes the system around the current state estimate using a first-order truncation of the multi-dimensional Taylor series expansion. In other words, Equations 2.11, 2.15 and 2.23 are approximated by,

$$\mathbf{y} \approx \mathbf{y}^{lin} = \mathbf{g}(\bar{\mathbf{x}}) + \mathbf{D}_{\delta_{\mathbf{x}}} \mathbf{g} \quad (2.24)$$

$$\bar{\mathbf{y}} \approx \bar{\mathbf{y}}^{lin} = \mathbf{g}(\bar{\mathbf{x}}) \quad (2.25)$$

$$\mathbf{P}_{\mathbf{y}} \approx \mathbf{P}_{\mathbf{y}}^{lin} = \mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T. \quad (2.26)$$

Applying this result to Equations 2.5, 2.6 and 2.8, we obtain:

$$\hat{\mathbf{x}}_k^- \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}}, \mathbf{u}_k) \quad (2.27)$$

$$\hat{\mathbf{y}}_k^- \approx \mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}}) \quad (2.28)$$

$$\mathcal{K}_k \approx \hat{\mathbf{P}}_{\mathbf{x}_k \mathbf{y}_k}^{lin} \left(\hat{\mathbf{P}}_{\mathbf{y}_k}^{lin} \right)^{-1}, \quad (2.29)$$

which forms the core of the EKF (the full recursive algorithm is given in Algorithm 1). Clearly these approximations will only be valid (and the resulting linearly calculated posterior statistics accurate), if all the higher order derivatives of the nonlinear functions are effectively zero over the “uncertainty region” of \mathbf{x} , as summarized by the support of its prior distribution. In other words, it requires the zeroth and first order terms of Equation 2.11 to dominate the remaining terms, over the region of the state-space defined by the prior distribution of \mathbf{x} . It is important to note here, that although this probabilistic spread of \mathbf{x} (or equivalently that of $\delta_{\mathbf{x}}$ around $\bar{\mathbf{x}}$), as captured by the covariance $\mathbf{P}_{\mathbf{x}}$, plays an important role in the validity of the EKFs “first order linearization”, it is completely *ignored*

Algorithm 1 The extended Kalman filter (EKF).

Initialization:

$$\hat{\mathbf{x}}_0 = E[\mathbf{x}_0] \quad (2.30)$$

$$\mathbf{P}_{\mathbf{x}_0} = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T] \quad (2.31)$$

$$\mathbf{R}_{\mathbf{v}} = E[(\mathbf{v} - \bar{\mathbf{v}})(\mathbf{v} - \bar{\mathbf{v}})^T] \quad (2.32)$$

$$\mathbf{R}_{\mathbf{n}} = E[(\mathbf{n} - \bar{\mathbf{n}})(\mathbf{n} - \bar{\mathbf{n}})^T] \quad (2.33)$$

For $k = 1, 2, \dots, \infty$:

1. *Prediction step*:

- Compute the process model Jacobians:

$$\mathbf{F}_{\mathbf{x}_k} = \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}, \bar{\mathbf{v}}, \mathbf{u}_k) |_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}} \quad (2.34)$$

$$\mathbf{G}_{\mathbf{v}} = \nabla_{\mathbf{v}} \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}, \mathbf{u}_k) |_{\mathbf{v}=\bar{\mathbf{v}}} \quad (2.35)$$

- Compute the predicted state mean and covariance (*time-update*)

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}}, \mathbf{u}_k) \quad (2.36)$$

$$\mathbf{P}_{\mathbf{x}_k}^- = \mathbf{F}_{\mathbf{x}_k} \mathbf{P}_{\mathbf{x}_k} \mathbf{F}_{\mathbf{x}_k}^T + \mathbf{G}_{\mathbf{v}} \mathbf{R}_{\mathbf{v}} \mathbf{G}_{\mathbf{v}}^T \quad (2.37)$$

2. *Correction step*:

- Compute the observation model Jacobians:

$$\mathbf{H}_{\mathbf{x}_k} = \nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x}, \bar{\mathbf{n}}) |_{\mathbf{x}=\hat{\mathbf{x}}_k^-} \quad (2.38)$$

$$\mathbf{D}_{\mathbf{n}} = \nabla_{\mathbf{n}} \mathbf{h}(\hat{\mathbf{x}}_k^-, \mathbf{n}) |_{\mathbf{n}=\bar{\mathbf{n}}} \quad (2.39)$$

- Update estimates with latest observation (*measurement update*)

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k}^- \mathbf{H}_{\mathbf{x}_k}^T (\mathbf{H}_{\mathbf{x}_k} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{H}_{\mathbf{x}_k}^T + \mathbf{D}_{\mathbf{n}} \mathbf{R}_{\mathbf{n}} \mathbf{D}_{\mathbf{n}}^T)^{-1} \quad (2.40)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k [\mathbf{y}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}})] \quad (2.41)$$

$$\mathbf{P}_{\mathbf{x}_k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_{\mathbf{x}_k}) \mathbf{P}_{\mathbf{x}_k}^- . \quad (2.42)$$

during the linearization process. Alternatively put, the linearization method employed by the EKF does not take into account the fact that \mathbf{x} is a random variable with inherent uncertainty. In effect, through the linearization around a *single point* (the current state estimate), the crucial expectation operator of Equations 2.5, 2.6 and 2.8 is omitted⁷. This has large implications for the accuracy and consistency of the resulting EKF algorithm. These approximations often introduce large errors in the EKF calculated posterior mean and covariance of the transformed (Gaussian) random variable, which may lead to suboptimal performance and sometimes divergence of the filter [77, 200, 204]. Three examples of this are given in the next section.

2.5 Demonstration of EKF Inaccuracy

2.5.1 Analytic scalar example

In this example we propagate a scalar random variable x through a simple nonlinear function to generate y , i.e.,

$$y = g(x) = x^2 , \quad (2.43)$$

where $x \sim \mathcal{N}(x; \bar{x}, \sigma_x^2)$ is drawn from a Gaussian distribution with mean \bar{x} and covariance σ_x^2 . We will next analytically calculate the mean and covariance of y , i.e., \bar{y} and σ_y^2 . using the scalar versions of Equations 2.17 and 2.23. Using the scalar version of Equations 2.17, the mean is calculated as

$$\bar{y} = g(\bar{x}) + E \left[(x - \bar{x}) \frac{dg(\bar{x})}{dx} + \frac{1}{2} (x - \bar{x})^2 \frac{d^2g(\bar{x})}{dx^2} + \dots \right] \quad (2.44)$$

$$= \bar{x}^2 + E [2(x - \bar{x})\bar{x} + (x - \bar{x})^2] \quad (2.45)$$

$$= \bar{x}^2 + E [2\bar{x}x - 2\bar{x}^2 + (x - \bar{x})^2] \quad (2.46)$$

$$= \bar{x}^2 + 2\bar{x}^2 - 2\bar{x}^2 + E [(x - \bar{x})^2] \quad (2.47)$$

$$= \bar{x}^2 + \sigma_x^2 , \quad (2.48)$$

⁷One can also interpret this as taking the expectation, but assuming the prior random variable is “peaked up” around its mean, i.e. $\mathbf{P}_{\mathbf{x}} \rightarrow 0$.

where $\frac{d^n g(\bar{x})}{dx^n}$ is the n th derivative of $g(x)$ evaluated at $x = \bar{x}$. We made use of the fact that all the derivatives of $g(x) = x^2$ above the second order are zero in Equation 2.45, and that $E[(x - \bar{x})^2] = \sigma_x^2$ (by definition) in Equation 2.48. Using the scalar version of Equations 2.23, the covariance is given by

$$\sigma_y^2 = \left(\frac{dg(\bar{x})}{dx} \right) \sigma_x^2 \left(\frac{dg(\bar{x})}{dx} \right) - \frac{1}{4} E \left[(x - \bar{x})^2 \frac{d^2 g(\bar{x})}{dx^2} \right]^2 + \quad (2.49)$$

$$= +E \left[\frac{1}{2} (x - \bar{x}) \frac{dg(\bar{x})}{dx} (x - \bar{x})^2 \frac{d^2 g(\bar{x})}{dx^2} + \frac{1}{2} (x - \bar{x})^2 \frac{d^2 g(\bar{x})}{dx^2} (x - \bar{x}) \frac{d^2 g(\bar{x})}{dx^2} + \right. \\ \left. + \frac{1}{4} (x - \bar{x})^2 \frac{d^2 g(\bar{x})}{dx^2} (x - \bar{x})^2 \frac{d^2 g(\bar{x})}{dx^2} \right] \quad (2.50)$$

$$= (2\bar{x})^2 \sigma_x^2 - \frac{1}{4} E [2(x - \bar{x})^2]^2 + E [(x - \bar{x})^4 + 4\bar{x}(x - \bar{x})^3] \quad (2.51)$$

$$= 4\bar{x}^2 \sigma_x^2 - (\sigma_x^2)^2 + E [(x - \bar{x})^4] + 4\bar{x} E [(x - \bar{x})^3] . \quad (2.52)$$

The third term in Equation 2.52, $E[(x - \bar{x})^4]$, is known as the *kurtosis* of x . For a Gaussian random variable, the kurtosis can be expressed in terms of the variance [152], i.e.,

$$E[(x - \bar{x})^4] = 3(\sigma_x^2)^2 . \quad (2.53)$$

The second part of the last term in Equation 2.52, $E[(x - \bar{x})^3]$, is known as the *skewness* of x . For a Gaussian random variable, this odd order central moment is zero. Substituting these results back into Equation 2.52, gives us the final value of the covariance of y in terms of the mean and covariance of x :

$$\begin{aligned} \sigma_y^2 &= 4\bar{x}^2 \sigma_x^2 - (\sigma_x^2)^2 + 3(\sigma_x^2)^2 \\ &= 2(\sigma_x^2)^2 + 4\bar{x}^2 \sigma_x^2 . \end{aligned} \quad (2.54)$$

In comparison to these exact results, the EKF calculates the mean and covariance of y using the first order Taylor series approximations given by Equations 2.25 and 2.26. For

the specific model used here (Equation 2.43), the resulting approximations are:

$$\bar{y}^{lin} = g(\bar{x}) = \bar{x}^2 \quad (2.55)$$

$$\begin{aligned} (\sigma_y^2)^{lin} &= \left(\frac{dg(\bar{x})}{dx} \right) \sigma_x^2 \left(\frac{dg(\bar{x})}{dx} \right) \\ &= 4\bar{x}^2 \sigma_x^2. \end{aligned} \quad (2.56)$$

Comparing Equation 2.55 with Equation 2.48 (mean), and Equation 2.56 with Equation 2.54 (covariance), we see that the EKF approximations has errors in both terms, given by

$$e_{\bar{y}} = \bar{y} - \bar{y}^{lin} = (\bar{x}^2 + \sigma_x^2) - \bar{x}^2 = \sigma_x^2 \quad (2.57)$$

$$e_{\sigma_y^2} = \sigma_y^2 - (\sigma_y^2)^{lin} = [2(\sigma_x^2)^2 + 4\bar{x}^2 \sigma_x^2] - 4\bar{x}^2 \sigma_x^2 = 2(\sigma_x^2)^2. \quad (2.58)$$

The magnitude of these errors are in both cases proportional to the variance of the prior random variable σ_x^2 . Clearly, the more “peaked” up the prior random variable x is around its mean, the more accurate the crude linearization of the EKF will be. For any significant probabilistic spread in x , the EKF calculated posterior mean will be biased and the posterior mean will be under estimated (inconsistent).

Finally, to experimentally verify the analytical results derived above, we performed three Monte Carlo simulations of the model described in Equation 2.43. We generated 100,000 samples for each experiment, drawn from a Gaussian distribution with mean $\bar{x} = 1$ and variance $\sigma_{x,j}^2 = \{\sigma_{x,1}^2, \sigma_{x,2}^2, \sigma_{x,3}^2\} = \{0.1, 1, 10\}$, i.e., $x_i \sim \mathcal{N}(x; 1, \sigma_{x,j}^2)$ for $i = 1, 2, \dots, N$, $N = 100,000$. These samples were then propagated trough the quadratic nonlinearity to generate a set of 100,000 posterior samples, i.e., $y_i = x_i^2$ for $i = 1, 2, \dots, N$. The posterior mean and covariance of y was then calculated using the standard empirical unbiased maximum-likelihood estimators, i.e.,

$$\hat{y} = \frac{1}{N} \sum_{i=1}^N y_i \quad \text{and} \quad \hat{\sigma}_y^2 = \frac{1}{N-1} \sum_{i=1}^N (y_i - \hat{y})^2. \quad (2.59)$$

Table 2.1 summarizes and compares the analytical mean and covariance (true and EKF linearized) as well as the experimental empirical verifications thereof. Clearly the empirical

Table 2.1: Mean and covariance of a nonlinear transformed scalar Gaussian random variable. The table compares the true analytically calculated posterior mean and covariance of $y = g(x) = x^2$, i.e., \bar{y} and σ_y^2 , with Monte Carlo (100,000 samples) calculated verifications thereof. The first order linearization approximations (as employed by the EKF) of these values are also shown. The experiment was repeated three times, the only difference being the covariance of the prior random variable x , i.e., x is a Gaussian random variable with mean $\bar{x} = 1$ and covariance $\sigma_x^2 = \{0.1, 1, 10\}$ for each experiment.

$\bar{x} = 1$	$\sigma_x^2 = 0.1$		$\sigma_x^2 = 1$		$\sigma_x^2 = 10$	
	\bar{y}	σ_y^2	\bar{y}	σ_y^2	\bar{y}	σ_y^2
True statistics (analytical)	1.1	0.42	2	6	11	240
True statistics (Monte Carlo)	1.101	0.419	2.000	5.998	10.99	240.27
Approximated statistics (EKF)	1	0.4	1	4.0	1	40

Monte Carlo results concur with our analytically calculated results. Furthermore, the inaccuracy of the first-order linearization approach of the EKF is clearly illustrated. The approximation error of the EKF for both the mean and covariance also grows (as expected) in relation with the magnitude of the prior covariance of x .

2.5.2 Arbitrary nonlinear transformation of Gaussian random variable

This demonstration contrasts the differences between the optimal and EKF approximated transformation of Gaussian random variables (GRVs). This experiment propagates a two dimensional GRV through an arbitrary nonlinear transformation and then compares the optimally calculated first and second order posterior statistics of the transformed RV with those calculated by the EKF. The optimal statistics were calculated with a Monte Carlo approach using 100,000 samples drawn from the prior distribution and then propagated through the full nonlinear mapping. For this experiment the arbitrary nonlinear mapping was generated by a randomly initialized multi-layer-perceptron (MLP) neural network with 2 input units, 10 hidden units and 2 output units. The neural network weights and biases were randomly drawn from a Gaussian distribution with a large enough covariance such that the resulting mapping was sufficiently nonlinear. The results of this experiment is shown in Figure 2.1. On the top-left the prior GRV (indicated by its mean, covariance-ellipse and representative sample distribution) is shown. The resulting posterior sample distribution (after the nonlinear transformation) with its true mean and covariance are

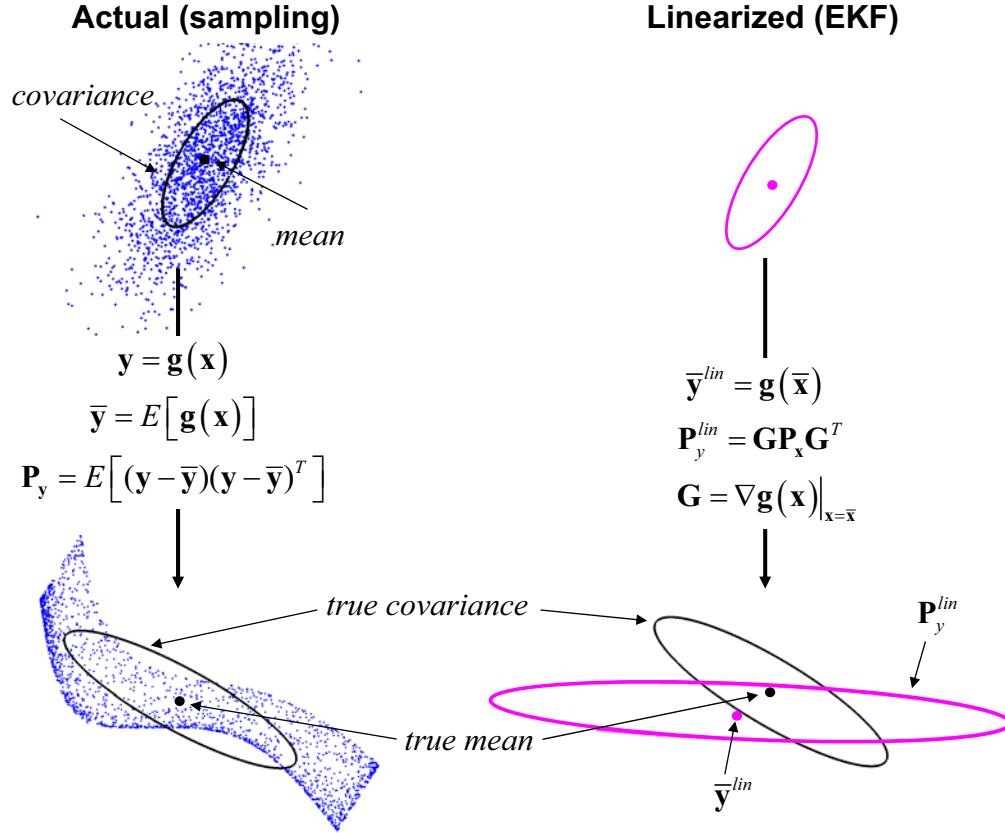


Figure 2.1: Optimal vs. EKF approximate transformation of Gaussian random variable (GRV). On the top-left a GRV (indicated by its mean, covariance-ellipse and representative sample distribution) undergoes an arbitrary highly nonlinear transformation. The resulting posterior sample distribution with its true mean and covariance are indicated at the bottom-left. On the right the GRV are analytically propagated through a “first-order linearized” (as used by the EKF) system. The posterior mean estimate is clearly biased and the posterior covariance estimate is highly inaccurate and inconsistent.

indicated at the bottom-left. On the right the GRV are analytically propagated through a “first-order linearized” (as used by the EKF) system. The posterior mean estimate is clearly biased and the posterior covariance estimate is highly inaccurate and inconsistent.

2.5.3 Conversion of polar to Cartesian coordinates

The third example illustrating the potential inaccuracies and inconsistencies of the EKF when applied to nonlinear estimation problems, is that of *polar to Cartesian coordinate transformation*. This ubiquitous transformation lies at the heart of the observation models

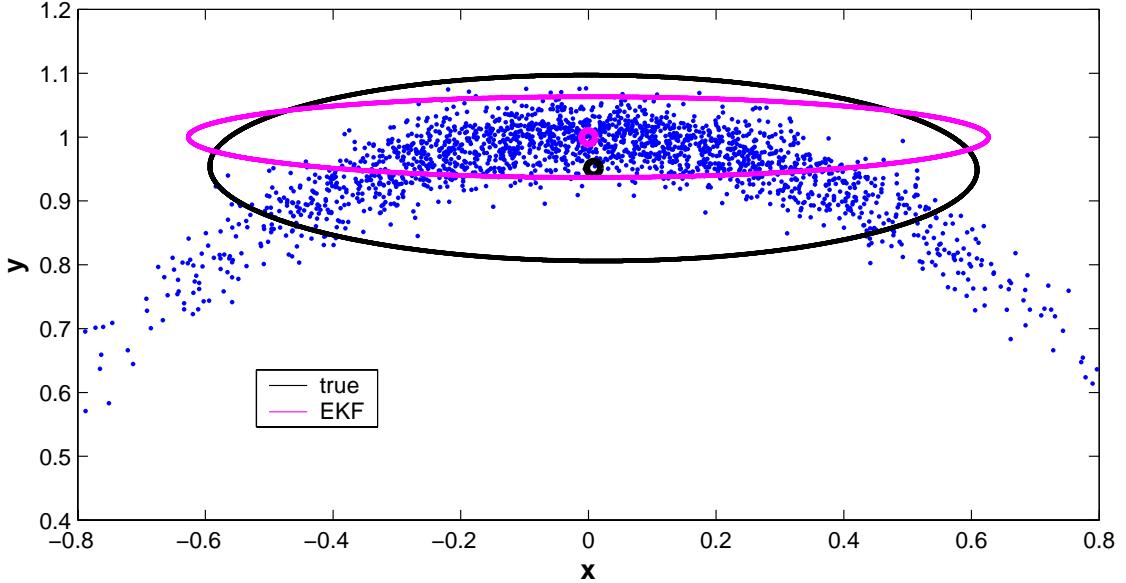


Figure 2.2: Optimal vs. EKF results on polar-to-Cartesian transformation experiment. The blue dots indicate the noisy polar coordinate measurements coming from a sensor such as a laser or sonar range finder. The true covariance and mean of this distribution is indicated by the black dot and ellipse. The EKF calculated (linearized approximation) mean and covariance is indicated in magenta.

for many real-world sensors such as sonar, radar and laser range finders to name but a few. A sensor returns polar measurement information that are related through a nonlinear transformation with the underlying Cartesian position of a target, i.e,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r \cos \theta \\ r \sin \theta \end{bmatrix},$$

where (r, θ) are the range and bearing angle of the target in the polar coordinate frame and (x, y) are the normal 2-dimensional Cartesian coordinates of the target. Linearizing this nonlinear observation model using a first order Taylor series truncation, quickly becomes very inaccurate for any significant bearing error. An example of this can be found in a range-optimized sonar sensor that can provide reasonably accurate range measurements (0.02m standard deviation) but extremely poor bearing measurements (standard deviation of up to 15°) [115]. The results of this experiment is shown in Figure 2.2. The true position of the target is at $(0, 1)$ in the Cartesian plane. Several thousand measurement samples were generated by taking the true range and bearing (r, θ) value of the target location and adding

zero-mean Gaussian noise to both the r and the θ components, and then converting back to the Cartesian plane. The resulting distribution (in the Cartesian plane) has a characteristic “banana”-shape, clearly indicating the better accuracy in the range direction than the bearing direction. The true mean and covariance of this distribution is indicated by the black dot and covariance ellipse. The mean and covariance calculated by the EKF (using Equations 2.25 and 2.26) are indicated by the magenta dot and ellipse. The EKF solution is clearly inaccurate. There is a large bias in the mean estimate in the range direction and the covariance is also underestimated in this direction. The underestimation results in an estimated posterior distribution that is too peaked (compared to the true posterior) in the range direction, resulting in “over confidence” in the resulting mean position estimate. This is called an *inconsistent estimate*, which, in more formal terms implies that $\text{trace}(\hat{\mathbf{P}}) < \text{trace}(\mathbf{P})$, where $\hat{\mathbf{P}}$ is the estimated covariance and \mathbf{P} is the true covariance. If the EKF filter now recurses on this result the filter can (and often does) diverge.

2.6 State, Parameter and Dual Estimation

In presenting the SPKF in Chapter 3 as a better alternative to the EKF, we shall cover a number of application areas of nonlinear estimation (probabilistic inference) in which the EKF has previously been applied. General application areas may be divided into *state estimation*, *parameter estimation* (machine learning, e.g., learning the weights of a neural network), and *dual estimation* (e.g., the expectation-maximization (EM) algorithm). Each of these areas place specific requirements on the SPKF or EKF, and will be developed in turn. An overview of the framework for these areas is briefly reviewed in this section.

2.6.1 State estimation

The basic framework for the EKF involves estimation of the state of a discrete-time non-linear DSSM,

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k) \quad (2.60)$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k), \quad (2.61)$$

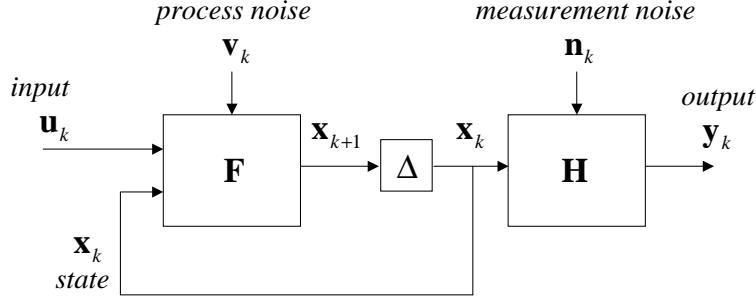


Figure 2.3: Block diagram of a discrete-time nonlinear DSSM used for general state estimation.

where \mathbf{x}_k represents the unobserved state of the system, \mathbf{u}_k is a known exogenous input, and \mathbf{y}_k is the observed measurement signal. The *process noise* \mathbf{v}_k drives the dynamic system, and the *observation noise* is given by \mathbf{n}_k . Note that we are not assuming additivity of the noise sources. The system dynamic model \mathbf{f} and \mathbf{h} are assumed known and parameterized by a set of known (given) parameters. A simple block diagram of this system is shown in Figure 2.3. In state estimation, the EKF is the standard Gaussian approximate method of choice to achieve a recursive (approximate) maximum-likelihood estimation of the state \mathbf{x}_k .

2.6.2 Parameter estimation

Parameter estimation, sometimes referred to as system identification or machine learning, involves determining a nonlinear mapping⁸

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k; \mathbf{w}) , \quad (2.62)$$

where \mathbf{x}_k is the input, \mathbf{y}_k is the output, and the nonlinear map $\mathbf{g}(\cdot)$ is parameterized by the vector \mathbf{w} . The nonlinear map, for example, may be a feed-forward or recurrent neural network (\mathbf{w} constitute the weights and biases), with numerous applications in regression, classification, and dynamic modeling [74, 76]. Learning corresponds to estimating the parameters \mathbf{w} in some “optimal” fashion. Typically, a training set is provided with sample pairs consisting of known input and desired outputs, $\{\mathbf{x}_k, \mathbf{d}_k\}$. The error of the machine

⁸For parameter estimation the index k does not necessarily indicate time, it can simply be an enumerating index.

is defined as

$$\mathbf{e}_k = \mathbf{d}_k - \mathbf{g}(\mathbf{x}_k; \mathbf{w}) , \quad (2.63)$$

and the goal of learning involves solving for the parameters \mathbf{w} in order to minimize the expectation of some given function of the error. While a number of optimization approaches exist (e.g., gradient descent using backpropagation), the EKF may be used to estimate the parameters by writing a new state-space representation

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{r}_k \quad (2.64)$$

$$\mathbf{d}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{w}_k) + \mathbf{e}_k , \quad (2.65)$$

where the parameters \mathbf{w}_k correspond to a stationary process with identity state transition matrix, driven by “artificial” process noise \mathbf{r}_k (the choice of variance determines convergence and tracking performance and will be discussed in further detail in Section 3.5.2). The output \mathbf{d}_k corresponds to a nonlinear observation on \mathbf{w}_k . The EKF can then be applied directly as an efficient “second-order” technique for learning the parameters [13]. The use of the EKF for training neural networks has been developed by Singhal and Wu [178] and Puskorius and Feldkamp [157, 158, 159], and is covered in detail in [143]. The use of the SPKF in this role is developed in Section 3.5.2.

2.6.3 Dual estimation

A special case of machine learning arises when the input \mathbf{x}_k is unobserved, and requires coupling both state estimation and parameter estimation. For these *dual estimation* problems, we again consider a discrete-time nonlinear dynamic system,

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k; \mathbf{w}) \quad (2.66)$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k; \mathbf{w}) , \quad (2.67)$$

where both the system state \mathbf{x}_k and the set of model parameters \mathbf{w} for the dynamic system must be simultaneously estimated from only the observed noisy signal \mathbf{y}_k . Example applications include adaptive nonlinear control, noise reduction (e.g., speech or image

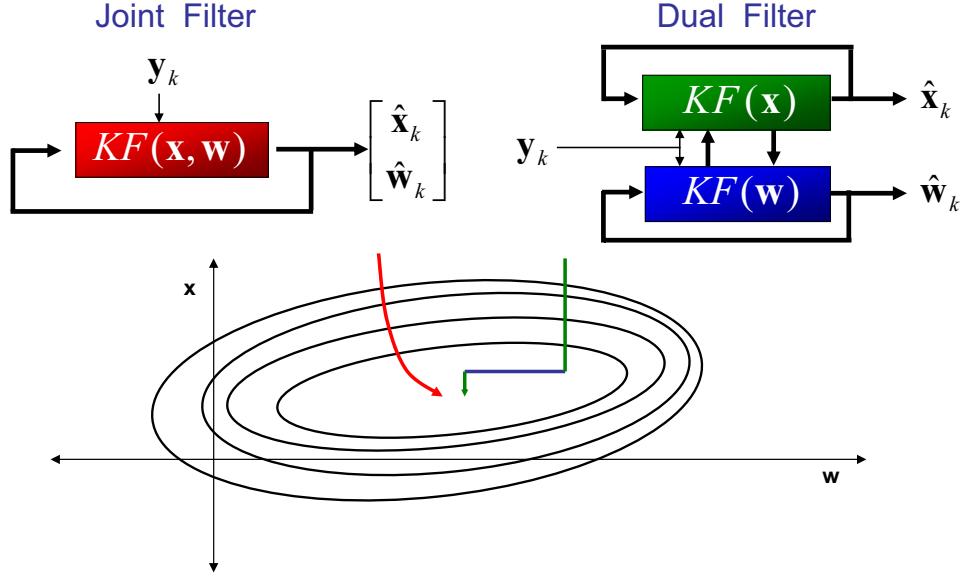


Figure 2.4: Dual estimation

enhancement), determining the underlying price of financial time-series, etc. A general theoretical and algorithmic framework for dual Kalman based estimation is presented by Nelson in [143] and leveraged in this work. This framework encompasses two main approaches, namely *joint filtering* and *dual filtering*. This is indicated in Figure 2.4.

In the *dual filtering approach*, a separate state-space representation is used for the state and the parameters. Two Kalman filters (EKF) are run simultaneously (in an iterative fashion) for state and parameter estimation. At every time step, the current estimate of the parameters $\hat{\mathbf{w}}_k$ is used in the state filter as a given (known) input, and likewise the current estimate of the state $\hat{\mathbf{x}}_k$ is used in the parameter filter. This results in a step-wise optimization within the combined state-parameter space as indicated on the right-hand side of Figure 2.4.

In the *joint filtering approach*, the unknown system state and parameters are concatenated into a single higher-dimensional *joint state vector*, $\tilde{\mathbf{x}}_k$, i.e.,

$$\tilde{\mathbf{x}}_k = \begin{bmatrix} \mathbf{x}_k^T & \mathbf{w}_k^T \end{bmatrix}^T$$

and the state space model is reformulated as

$$\begin{aligned}\tilde{\mathbf{x}}_{k+1} &= \tilde{\mathbf{f}}(\tilde{\mathbf{x}}_k, \mathbf{u}_k, \tilde{\mathbf{v}}_k) \\ \mathbf{y}_k &= \tilde{\mathbf{h}}(\tilde{\mathbf{x}}_k, \mathbf{n}_k),\end{aligned}$$

which can be expanded to

$$\begin{aligned}\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{w}_{k+1} \end{bmatrix} &= \begin{bmatrix} \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k; \mathbf{w}_k) \\ \mathbf{w}_k \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_k \end{bmatrix} \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k; \mathbf{w}_k) ,\end{aligned}$$

where $\tilde{\mathbf{v}}_k = [\mathbf{v}_k^T \quad \mathbf{r}_k^T]^T$. A single EKF (or SPKF) is now run on the joint state space to produce simultaneous estimates of the states \mathbf{x}_k and the parameters \mathbf{w} . The left-hand part of Figure 2.4 illustrates the joint estimation process: At the top of the image a block diagram of the joint estimation framework is shown. At the bottom, a stylized representation is shown indicating how the joint approach attempts direct optimization of the combined state-parameter space.

Approaches to dual estimation utilizing the SPKF are developed in Section 3.5.3.

2.7 Chapter Summary

In this chapter the optimal recursive Gaussian approximate Bayesian estimation algorithm was introduced, analyzed and compared to the flawed approximation employed by the EKF. It was indicated (and stressed) how the calculation of the statistics of a Gaussian random variable that undergoes a nonlinear transformation forms the core of the optimal Gaussian update. Utilizing a multi-dimensional Taylor series analysis, we showed how the EKF only achieves first order accuracy in the calculation of both the posterior mean and covariance of the transformed random variables. This limited accuracy of the EKF was demonstrated in two experimental examples that clearly indicated the nature of this problem as well as how it can lead to filter divergence. As discussed, one of the reasons for the EKFs inaccuracies is the failure to take into account the inherent “uncertainty” in the prior random variable during the linearization process. This insight will be revisited and investigated further in

the next chapter, where we introduce the family of sigma-point Kalman filters that attempt to address the short-comings of the EKF.

Chapter 3

Sigma-Point Kalman Filters: Derivation, Algorithmic Implementation, Applications & Extensions

3.1 Introduction

In order to improve the accuracy, consistency (robustness) and efficiency¹ of Gaussian approximate inference algorithms applied to general nonlinear DSSMs, the two major shortcomings of the EKF need to be addressed. These are: 1) Disregard for the “probabilistic uncertainty” of the underlying system state and noise RVs during the linearization of the system equations, and 2) Limited *first-order accuracy* of propagated means and covariances resulting from a first-order truncated Taylor-series linearization method. As discussed in Chapter 1, two related algorithms, the *unscented Kalman filter* [94] and the *central difference Kalman filter* [148, 88], have recently been proposed in an attempt to address these issues. This is done through the use of novel deterministic sampling approaches to approximate the optimal gain and prediction terms in the Gaussian approximate linear Bayesian update of the Kalman filter framework (Equations 2.5 - 2.8). These derivativeless², deterministic sampling based Kalman filters consistently outperform the EKF not only in terms of estimation accuracy, but also in filter robustness and easy of implementation, for

¹The *accuracy* of an estimator is an indication of the average magnitude (taken over the distribution of the data) of the estimation error. An estimator is *consistent* if $\text{trace}[\hat{\mathbf{P}}_{\mathbf{x}}] \geq \text{trace}[\mathbf{P}_{\mathbf{x}}]$ and it is *efficient* if that lower bound on the state error covariance is tight.

²Unlike the EKF, no analytical or perturbation based derivatives of the system equations need to be calculated.

no added computational cost [204, 200, 77, 191, 192, 94, 99, 100, 96, 148, 88].

In this chapter, we will first introduce both the unscented Kalman filter (UKF) and the central difference Kalman filter (CDKF), show how they are motivated and derived, and completely specify their algorithmic implementations. We will then show how the derivativeless Gaussian random variable propagation techniques that lie at the core of these algorithms, can be viewed as different implementations of a general deterministic sampling framework for the calculation of the posterior mean and covariance of the pertinent Gaussian approximate densities in the Kalman framework recursion. We call this general framework the *sigma-point approach*. This in turn allows us to group all Kalman filters which are implicitly based on the sigma-point approach into a larger family of algorithms called ***sigma-point Kalman filters* (SPKF)**.

We continue to show how we extended the SPKF family of algorithms by deriving numerically efficient and stable square-root forms of both the UKF and the CDKF. As mentioned in Section 1.7, one of the main contributions of this thesis is the extension of the SPKF framework to the larger probabilistic inference domain, with specific focus on nonlinear parameter and dual estimation. The derivation of these different algorithmic forms of the SPKF is covered in detail in this chapter along with their application to numerous inference problems. In most cases the experiments will not only verify the correctness and utility of the SPKF algorithm, but also contrast its performance relative to that of the EKF on the same problem.

3.2 The Unscented Kalman Filter

The unscented Kalman filter (UKF) is a recursive MMSE estimator based on the optimal Gaussian approximate Kalman filter framework, that addresses some of the approximation issues of the EKF [100]. Because the EKF only uses the first order terms of the Taylor series expansion of the nonlinear functions, it often introduces large errors in the estimated statistics of the posterior distributions of the states. This is especially evident when the models are highly nonlinear and the local linearity assumption breaks down, i.e., the effects of the higher order terms of the Taylor series expansion becomes significant. Unlike the

EKF, the UKF does not explicitly approximate the nonlinear process and observation models, it uses the *true* nonlinear models and rather approximates the distribution of the state random variable. In the UKF the state distribution is still represented by a Gaussian random variable (GRV), but it is specified using a minimal set of deterministically chosen sample points. These sample points completely capture the true mean and covariance of the GRV, and when propagated through the true nonlinear system, captures the posterior mean and covariance accurately to the 2nd order for any nonlinearity, with errors only introduced in the 3rd and higher orders. To elaborate on this, we start by first explaining the deterministic sampling approach called the *unscented transformation*³. After this, the *scaled unscented transformation* (SUT) is introduced and discussed. The scaled unscented transformation is a generalizing extension of the unscented transformation and forms the algorithmic core of the unscented Kalman filter.

3.2.1 The unscented transformation

The unscented transformation (UT) is a method for calculating the statistics of a random variable which undergoes a nonlinear transformation and builds on the principle that it is easier to approximate a probability distribution than an arbitrary nonlinear function [99]. As in Chapter 2, we consider the propagation of a L dimensional random variable \mathbf{x} through an arbitrary nonlinear function,

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) . \quad (3.1)$$

Assume \mathbf{x} has mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_{\mathbf{x}}$. To calculate the statistics (first two moments) of \mathbf{y} using the UT, we proceed as follows: First, a set of $2L + 1$ weighted samples, called *sigma-points*, $\mathcal{S}_i = \{w_i, \mathbf{x}_i\}$ are deterministically chosen so that they completely capture the true mean and covariance of the prior random variable \mathbf{x} . A selection scheme that

³The reason why this transformation is named “*unscented*” is wrapped in mystery. The author has tried in vain to coax the true implied meaning of this term from the inventor, but to this day, it remains a riddle [98].

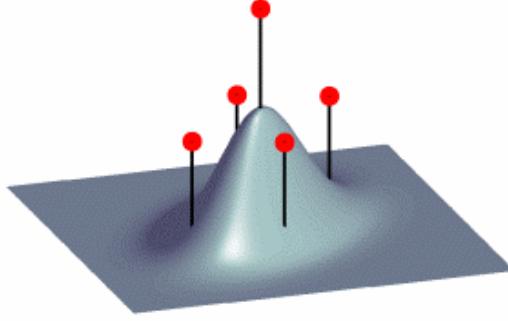


Figure 3.1: Weighted sigma-points for a 2 dimensional Gaussian random variable (RV). These sigma-points lie along the major eigen-axes of the RV's covariance matrix and complete captures the first and second order statistics of the RV. The height of each sigma-point indicates its relative weight. For this example, $\kappa = 1$ was used.

satisfies this requirement is:

$$\begin{aligned} \mathbf{\mathcal{X}}_0 &= \bar{\mathbf{x}} & w_0 &= \frac{\kappa}{L+\kappa} & i &= 0 \\ \mathbf{\mathcal{X}}_i &= \bar{\mathbf{x}} + \left(\sqrt{(L+\kappa)\mathbf{P}_{\mathbf{x}}} \right)_i & w_i &= \frac{1}{2(L+\kappa)} & i &= 1, \dots, L \\ \mathbf{\mathcal{X}}_i &= \bar{\mathbf{x}} - \left(\sqrt{(L+\kappa)\mathbf{P}_{\mathbf{x}}} \right)_i & w_i &= \frac{1}{2(L+\kappa)} & i &= L+1, \dots, 2L \end{aligned} \quad (3.2)$$

where w_i is the weight associated with the i th sigma-point such that $\sum_{i=0}^{2L} w_i = 1$. κ is a scaling parameter and $\left(\sqrt{(L+\kappa)\mathbf{P}_{\mathbf{x}}} \right)_i$ is the i th column (or row) of the matrix square root of the weighted covariance matrix, $(L+\kappa)\mathbf{P}_{\mathbf{x}}$. The numerically efficient *Cholesky factorization* method [156] is typically used to calculate the matrix square root. Since the matrix square-root of a positive-definite matrix is not unique, any ortho-normal rotation of the sigma-point set is again a valid set. Furthermore, if desired one can derive a selection scheme that captures higher order moment information such as skew or kurtosis [96]. This will in general require a larger set of sigma-points however. Figure 3.1 shows the location and weight (indicated by height) of a typical sigma-point set generated for a two dimensional Gaussian random variable.

Each sigma point is now propagated through the nonlinear function

$$\mathbf{y}_i = \mathbf{g}(\mathbf{\mathcal{X}}_i) \quad i = 0, \dots, 2L \quad (3.3)$$

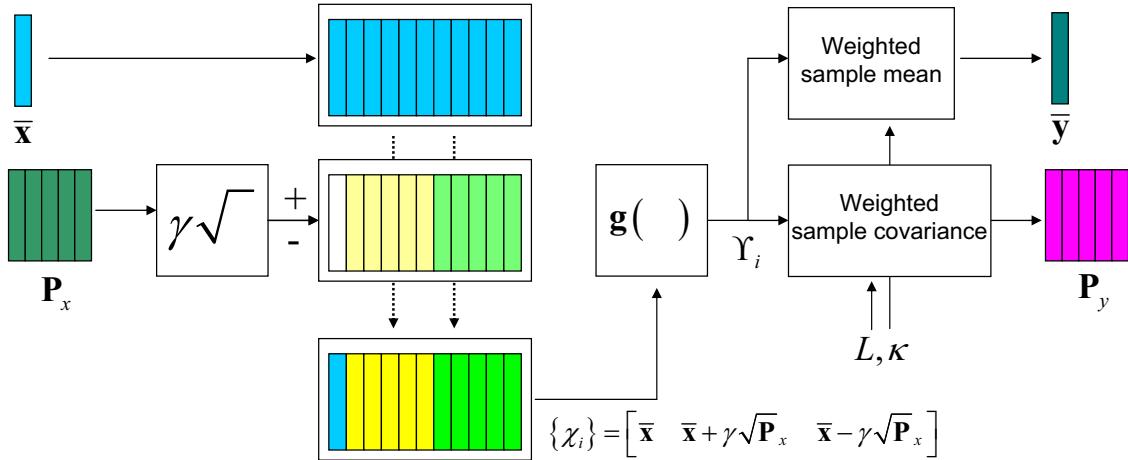


Figure 3.2: Schematic diagram of the unscented transformation.

and the approximated mean, covariance and cross-covariance of \mathbf{y} are computed as follows:

$$\bar{\mathbf{y}} \approx \sum_{i=0}^{2L} w_i \mathbf{y}_i \quad (3.4)$$

$$\mathbf{P}_y \approx \sum_{i=0}^{2L} w_i (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^T \quad (3.5)$$

$$\mathbf{P}_{xy} \approx \sum_{i=0}^{2L} w_i (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{y}_i - \bar{\mathbf{y}})^T . \quad (3.6)$$

These estimates of the mean and covariance are accurate to the second order (third order for true Gaussian priors) of the Taylor series expansion of $\mathbf{g}(\mathbf{x})$ for *any* nonlinear function⁴. Errors are introduced in the third and higher order moments but are scaled by the choice of the parameter κ . In comparison (as shown in Section 2.4), the EKF only calculates the posterior mean and covariance accurately to the first order with all higher order moments truncated. A comparison of the performance of the UT versus that of the linearization approach used in the EKF is shown in Figure 3.3. Note that the approximations of the posterior statistics as calculated by the UT using Equations 3.4 - 3.6 is *exact* for linear systems, i.e., if $\mathbf{g}(\mathbf{x}) = \mathbf{Ax} + \mathbf{b}$. Figure 3.2 provides a schematic representation of the unscented transformation.

⁴A derivation of the accuracy of the unscented transformation is given in Section 4.3 of Chapter 4.

Julier's original derivation of the unscented transformation was based on the following reasoning[98]: Assuming that we can calculate a set of weighted samples that accurately capture the relevant prior statistics of a random variable, what should the value of these weights and sample locations be, such that we can accurately approximate the posterior mean and covariance using only weighted sample mean and covariance estimators (Equations 3.4 - 3.6) operating on functional evaluations of these samples (Equation 3.3)? Specifically, he wanted to accurately capture the posterior mean and covariance up to and including the second order terms in the Taylor series expansion of the true quantities (see Equations 2.20 and 2.23). The solution to this problem was found by comparing the Taylor series expansions of the estimators⁵ (Equations 3.4 - 3.6) with the true quantities and choosing the weights and sigma-point locations such that the first and second order terms matched exactly. Under this scheme, errors are only introduced in the higher (> 2) order terms. For more detail, see [101].

The sigma-point selection scheme used in the UT has the property that as the dimension of the state-space (L) increases, the radius of the sphere that bounds all the sigma points increases as well. Even though the mean and covariance of the prior distribution are still captured correctly, it does so at the cost of possibly sampling non-local effects. If the nonlinearities in question are very severe, this can lead to significant difficulties. In order to address this problem, the sigma points can be scaled towards or away from the mean of the prior distribution by a proper choice of κ . The distance of the i th sigma point from $\bar{\mathbf{x}}$, $|\mathbf{x}_i - \bar{\mathbf{x}}|$, is proportional to $\sqrt{L + \kappa}$. When $\kappa = 0$, the distance is proportional to \sqrt{L} . When $\kappa > 0$ the points are scaled further from $\bar{\mathbf{x}}$ and when $\kappa < 0$ the points are scaled *towards* $\bar{\mathbf{x}}$. For the special case of $\kappa = 3 - L$, the desired dimensional scaling invariance is achieved by canceling the effect of L . However, when $\kappa = 3 - L < 0$ the weight $w_0 < 0$ and the calculated covariance can become *non-positive semidefinite*. The *scaled unscented transformation* was developed to address this problem [97].

⁵See Section 4.3 for a derivation of the accuracy of the unscented transformation based on a Taylor series expansion analysis.

3.2.2 The scaled unscented transformation

The scaled unscented transformation (SUT) replaces the original set of sigma-points with a transformed set given by

$$\boldsymbol{\chi}'_i = \boldsymbol{\chi}_0 + \alpha (\boldsymbol{\chi}_i - \boldsymbol{\chi}_0) \quad i = 0 \dots 2L , \quad (3.7)$$

where α is a positive scaling parameter which can be made arbitrarily small ($0 < \alpha < 1$) to minimize possible higher order effects. This formulation gives an extra degree of freedom to control the scaling of the sigma-points without causing the resulting covariance to possibly become non-positive semidefinite. This is achieved by applying the UT to an *auxiliary random variable* propagation problem which is related to the original nonlinear model (Equation 3.1) by

$$\mathbf{z} = \mathbf{g}'(\mathbf{x}) = \frac{\mathbf{g}(\bar{\mathbf{x}} + \alpha(\mathbf{x} - \bar{\mathbf{x}})) - \mathbf{g}(\bar{\mathbf{x}})}{\alpha^2} + \mathbf{g}(\bar{\mathbf{x}}) . \quad (3.8)$$

The Taylor series expansion of $\bar{\mathbf{z}}$ and $\mathbf{P}_{\mathbf{z}}$ agrees with that of $\bar{\mathbf{y}}$ and \mathbf{P}_y exactly up to the second order, with the higher order terms scaling geometrically with a common ratio of α [97]. The same second order accuracy of the normal UT is thus retained with a controllable scaling of the higher order errors by a proper choice of α . The auxiliary random variable formulation of the SUT is identical to applying the original UT on a *pre-scaled* set of sigma-points [97]. A set of sigma-points $\mathcal{S} = \{w_i, \boldsymbol{\chi}_i ; i = 0, \dots, 2L\}$ is calculated using Equation 3.2 and then transformed into the scaled set $\mathcal{S}' = \{w'_i, \boldsymbol{\chi}'_i ; i = 0, \dots, 2L\}$ by

$$\boldsymbol{\chi}'_i = \boldsymbol{\chi}_0 + \alpha (\boldsymbol{\chi}_i - \boldsymbol{\chi}_0) \quad (3.9)$$

$$w'_i = \begin{cases} w_0/\alpha^2 + (1 - 1/\alpha^2) & i = 0 \\ w_i/\alpha^2 & i = 1, \dots, 2L \end{cases} \quad (3.10)$$

where α is the new sigma-point *scaling* parameter. The sigma point selection and scaling can also be combined into a single step (thereby reducing the number of calculations) by setting

$$\lambda = \alpha^2(L + \kappa) - L , \quad (3.11)$$

and selecting the sigma-point set by:

$$\begin{aligned}\mathbf{\mathcal{X}}_0 &= \bar{\mathbf{x}} & w_0^{(m)} &= \frac{\lambda}{L+\lambda} & i=0 \\ \mathbf{\mathcal{X}}_i &= \bar{\mathbf{x}} + \left(\sqrt{(L+\lambda)\mathbf{P}_{\mathbf{x}}} \right)_i & i=1,\dots,L & w_0^{(c)} &= \frac{\lambda}{L+\lambda} + (1-\alpha^2+\beta) & i=0 \\ \mathbf{\mathcal{X}}_i &= \bar{\mathbf{x}} - \left(\sqrt{(L+\lambda)\mathbf{P}_{\mathbf{x}}} \right)_i & i=L+1,\dots,2L & w_i^{(m)} = w_i^{(c)} &= \frac{1}{2(L+\lambda)} & i=1,\dots,2L\end{aligned}\quad (3.12)$$

The weighting on the zeroth sigma-point directly affects the magnitude of the errors in the fourth and higher order terms for symmetric prior distributions [97]. A third parameter, β , is thus introduced which affects the weighting of the zeroth sigma-point for the calculation of the covariance. This allows for the minimization of higher order errors if prior knowledge (e.g. kurtosis, etc.) of the distribution of \mathbf{x} is available. The complete scaled unscented transformation is thus given by the following:

1. Choose the parameters κ , α and β . Choose $\kappa \geq 0$ to guarantee positive semi-definiteness of the covariance matrix. The specific value of κ is not critical though, so a good default choice is $\kappa = 0$. Choose $0 \leq \alpha \leq 1$ and $\beta \geq 0$. α controls the “size” of the sigma-point distribution and should ideally be a small number to avoid sampling non-local effects when the nonlinearities are strong. Here “locality” is defined in terms on the probabilistic spread of \mathbf{x} as summarized by its covariance. β is a non-negative weighting term which can be used to incorporate knowledge of the higher order moments of the distribution. For a Gaussian prior the optimal choice is $\beta = 2$ [97]. This parameter can also be used to control the error in the kurtosis which affects the ‘heaviness’ of the tails of the posterior distribution.
2. Calculate the set of $2L + 1$ scaled sigma-points and weights $\mathcal{S} = \{w_i, \mathbf{\mathcal{X}}_i ; i = 0, \dots, 2L\}$ by setting $\lambda = \alpha^2(L + \kappa) - L$ and using the combined selection/scaling scheme of Equation 3.12. As mentioned earlier, L is the dimension of \mathbf{x} .
3. Propagate each sigma point through the nonlinear transformation:

$$\mathbf{\mathcal{Y}}_i = \mathbf{g}(\mathbf{\mathcal{X}}_i) \quad i = 0, \dots, 2L \quad (3.13)$$

4. The mean $\bar{\mathbf{y}}$, covariance \mathbf{P}_y and cross-covariance \mathbf{P}_{xy} are computed as follows:

$$\bar{\mathbf{y}} \approx \sum_{i=0}^{2L} w_i^{(m)} \mathbf{y}_i \quad (3.14)$$

$$\mathbf{P}_y \approx \sum_{i=0}^{2L} w_i^{(c)} (\mathbf{y}_i - \bar{\mathbf{y}}) (\mathbf{y}_i - \bar{\mathbf{y}})^T \quad (3.15)$$

$$\mathbf{P}_{xy} \approx \sum_{i=0}^{2L} w_i^{(c)} (\mathbf{x}_i - \bar{\mathbf{x}}) (\mathbf{y}_i - \bar{\mathbf{y}})^T . \quad (3.16)$$

Note: Not much research effort has yet been spent on determining if there is a *global optimal* setting of the SUT scaling parameters (α , β and κ) and it is not one of the research questions that this dissertation aimed to answer. That said, our empirical evidence seems to indicate that such an optimal set is *problem specific* which thwart the effort to find such an global optimum. However, it also seems that the algorithm is not very sensitive to the exact values chosen for these parameters as long as they result in a numerically well behaved set of sigma-points and weights.

Demonstration of SUT

In order to demonstrate the accuracy of the scaled unscented transformation (SUT), we revisit the nonlinear mapping problem first presented in Section 2.5.

In this experiment we propagate a two dimensional GRV through an arbitrary nonlinear transformation and then compare the optimally calculated first and second order posterior statistics of the transformed RV with those calculated through normal linearization (EKF), and by the scaled unscented transformation (SUT). The optimal statistics were calculated with a Monte Carlo approach using 100,000 samples drawn from the prior distribution and then propagated through the full nonlinear mapping. For this experiment the arbitrary nonlinear mapping was generated by a randomly initialized multi-layer-perceptron (MLP) neural network with 2 input units, 10 hidden units and 2 output units. The neural network weights and biases were randomly drawn from a Gaussian distribution with a large enough covariance such that the resulting mapping was sufficiently nonlinear. The results of this experiment is shown in Figure 3.3. On the top-left the prior GRV (indicated by its

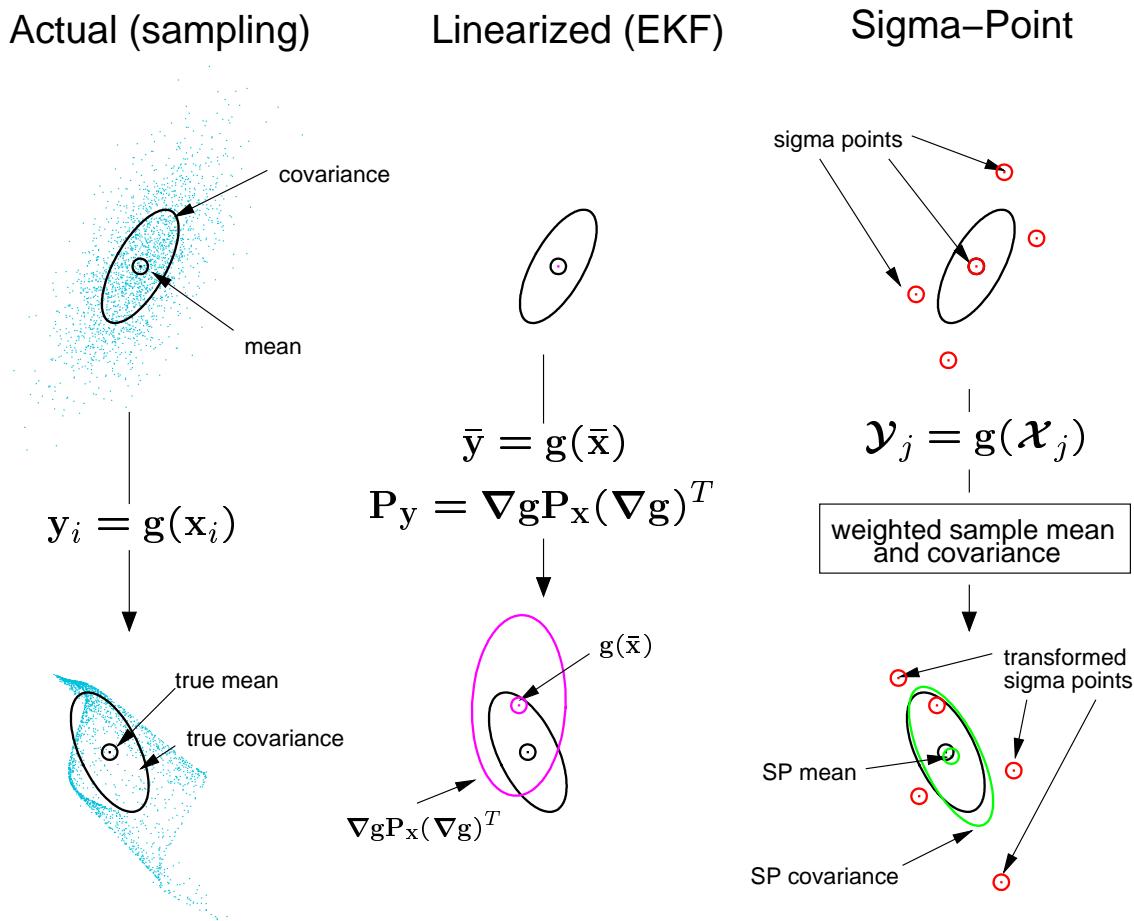


Figure 3.3: Demonstration of the accuracy of the scaled unscented transformation for mean and covariance propagation. a) actual, b) first-order linearization (EKF), c) SUT (sigma-point) A cloud of 5000 samples drawn from a Gaussian prior is propagated through an arbitrary highly nonlinear function and the true posterior sample mean and covariance are calculated. This reflects the truth as calculated by a Monte Carlo approach and is shown in the left plot. Next, the posterior random variable's statistics are calculated by a linearization approach as used in the EKF. The middle plot shows these results. The errors in both the mean and covariance as calculated by this “first-order” approximation is clearly visible. The plot on the right shows the results of the estimates calculated by the scaled unscented transformation. There is almost no bias error in the estimate of the mean and the estimated covariance is also much closer to the true covariance. The superior performance of the SUT approach is clearly evident.

mean, covariance-ellipse and representative sample distribution) is shown. The resulting posterior sample distribution (after the nonlinear transformation) with its true mean and covariance are indicated at the bottom-left. In the center plot, the GRV is analytically propagated through a “first-order linearized” (as used by the EKF) system. The posterior mean estimate is clearly biased and the posterior covariance estimate is highly inaccurate. The plot on the right shows the results using the SUT (note only 5 sigma-points are used compared to the thousands needed by the pure Monte Carlo method on the left). The bottom-right plot of Figure 3.3 indicates the posterior statistics of \mathbf{y} as calculated by the SUT (note also the posterior location of the propagated sigma-points). The superior performance of the sigma-point approach of the SUT over the normal linearization method is clear and closely matches the optimal result. Also note that the SUT calculated posterior covariance seems to be consistent and efficient. The consistency stems from the fact the SUT’s green posterior covariance ellipse is larger than the true posterior covariance ellipse (indicated in black), but since it is still “tight” around it, it seems efficient as well.

3.2.3 Implementing the unscented Kalman filter

The unscented Kalman filter (UKF) is a straightforward application of the scaled unscented transformation to the recursive Kalman filter framework as presented in Section 2.2 of the previous chapter. Specifically, the SUT is used to approximate the optimal terms in Equations 2.5-2.8 of the optimal Gaussian approximate linear Bayesian update, where the state random variable (RV) is redefined as the concatenation of the original state and the process and observation noise random variables:

$$\mathbf{x}_k^a = \begin{bmatrix} \mathbf{x}_k^x \\ \mathbf{x}_k^v \\ \mathbf{x}_k^n \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{v}_k \\ \mathbf{n}_k \end{bmatrix}. \quad (3.17)$$

The effective dimension of this augmented state RV is now $L = L_x + L_v + L_n$, i.e., $\mathbf{x}^a \in \mathbb{R}^{L_x+L_v+L_n}$, where L_x is the original state dimension, L_v is the process noise dimension and L_n is the observation noise dimension. In a similar manner the augmented state covariance

matrix is built up from the individual covariances matrices of \mathbf{x} , \mathbf{v} and \mathbf{n} :

$$\mathbf{P}^a = \begin{bmatrix} \mathbf{P}_{\mathbf{x}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\mathbf{v}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{\mathbf{n}} \end{bmatrix}, \quad (3.18)$$

where $\mathbf{R}_{\mathbf{v}}$ and $\mathbf{R}_{\mathbf{n}}$ are the process and observation noise covariances. The SUT sigma-point selection scheme is applied to this new augmented state RV to calculate the corresponding augmented sigma-point set \mathcal{X}_k^a . By augmenting the state random variable with the noise random variables, we take the uncertainty in the noise RVs into account in the same manner as we do for the state during the sigma-point propagation. This allows for the effect of the noise on the system dynamics and observations to be captured with the same level of accuracy as with which we treat the state. In contrast, the EKF simply models the noise RV's using their expected values, which for zero-mean Gaussian noise is (not surprisingly) equal to $\mathbf{0}$.

The complete UKF algorithm that updates the mean $\hat{\mathbf{x}}_k$ and covariance $\mathbf{P}_{\mathbf{x}_k}$ of the Gaussian approximation to the posterior distribution of the states will now be presented:

Algorithm 2 : The Unscented Kalman Filter (UKF)

- *Initialization:* $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0] \quad , \quad \mathbf{P}_{\mathbf{x}_0} = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$
- $\hat{\mathbf{x}}_0^a = E[\mathbf{x}^a] = E[\hat{\mathbf{x}}_0 \quad \mathbf{0} \quad \mathbf{0}]^T \quad , \quad \mathbf{P}_0^a = E[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T] = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\mathbf{v}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{\mathbf{n}} \end{bmatrix}$
- *For* $k = 1, \dots, \infty :$

1. Calculate sigma-points:

$$\mathcal{X}_{k-1}^a = \left[\hat{\mathbf{x}}_{k-1}^a \quad \hat{\mathbf{x}}_{k-1}^a + \gamma \sqrt{\mathbf{P}_{k-1}^a} \quad \hat{\mathbf{x}}_{k-1}^a - \gamma \sqrt{\mathbf{P}_{k-1}^a} \right] \quad (3.19)$$

2. Time-update equations:

$$\mathcal{X}_{k|k-1}^x = \mathbf{f}(\mathcal{X}_{k-1}^x, \mathcal{X}_{k-1}^v, \mathbf{u}_{k-1}) \quad (3.20)$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \mathcal{X}_{i,k|k-1}^x \quad (3.21)$$

$$\mathbf{P}_{\mathbf{x}_k}^- = \sum_{i=0}^{2L} w_i^{(c)} (\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-) (\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-)^T \quad (3.22)$$

3. Measurement-update equations:

$$\mathcal{Y}_{k|k-1} = \mathbf{h}(\mathcal{X}_{k|k-1}^x, \mathcal{X}_{k|k-1}^n) \quad (3.23)$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \mathcal{Y}_{i,k|k-1} \quad (3.24)$$

$$\mathbf{P}_{\tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} w_i^{(c)} (\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-) (\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T \quad (3.25)$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sum_{i=0}^{2L} w_i^{(c)} (\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-) (\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T \quad (3.26)$$

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k}^{-1} \quad (3.27)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (3.28)$$

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k} \mathbf{K}_k^T \quad (3.29)$$

- *Parameters:* $\mathbf{x}^a = [\mathbf{x}^T \quad \mathbf{v}^T \quad \mathbf{n}^T]^T$, $\mathcal{X}^a = [(\mathcal{X}^x)^T \quad (\mathcal{X}^v)^T \quad (\mathcal{X}^n)^T]^T$, $\gamma = \sqrt{L + \lambda}$: γ is a composite scaling parameter and λ is given by Eq. 3.11, L is the dimension of the augmented states, \mathbf{R}_v is the process-noise covariance, \mathbf{R}_n is the observation-noise covariance, and w_i are the weights as calculated in Eq. 3.12.

3.3 The Central Difference Kalman Filter

Separately from the development of the UKF, two different groups [88, 148] proposed another derivativeless Kalman filter for nonlinear estimation, based on *Sterling's polynomial interpolation formula* [182]. This formulation was the basis of Norgaard's [147, 148] derivation of the *divided difference filter* as well as Ito and Xiong's [88] *central difference filter*. These two filters are essentially identical and will henceforth be referred to simply as the *central difference Kalman filter* (CDKF). In order to show how this filter was derived, we will first briefly discuss Sterling's polynomial interpolation method of approximating nonlinear functions. Specifically, we will focus on the 2nd order Sterling approximation which forms the core of the CDKF.

3.3.1 Second order Sterling polynomial interpolation

As shown in Section 2.3, the Taylor series expansion of a nonlinear function of a random variable x around some point, say \bar{x} (its mean), is given by the following (for the scalar case):

$$g(x) = g(\bar{x}) + D_{\delta_x}g + \frac{1}{2!}D_{\delta_x}^2g + \dots \quad (3.30)$$

$$= g(\bar{x}) + (x - \bar{x})\frac{dg(\bar{x})}{dx} + \frac{1}{2!}(x - \bar{x})^2\frac{d^2g(\bar{x})}{dx^2} \quad (3.31)$$

where we limited the expansion to the second order term. Another way to approximate a nonlinear function over a certain interval is to make use of an interpolation formula, that uses a finite number of functional evaluations instead of analytical derivatives. One particular type of interpolation formula that uses central divided differences, is *Sterling's polynomial interpolation formula* [57, 37], which for the scalar 2nd order case is given by:

$$g(x) = g(\bar{x}) + \tilde{D}_{\Delta_x}g + \frac{1}{2!}\tilde{D}_{\Delta_x}^2g \quad (3.32)$$

where $\tilde{D}_{\Delta_x}g$ and $\tilde{D}_{\Delta_x}^2g$ are the first and second order central divided difference operators acting on $g(x)$. For the scalar case, these are given by

$$\tilde{D}_{\Delta_x}g = (x - \bar{x}) \frac{g(\bar{x} + h) - g(\bar{x} - h)}{2h} \quad (3.33)$$

$$\tilde{D}_{\Delta_x}^2g = (x - \bar{x})^2 \frac{g(\bar{x} + h) + g(\bar{x} - h) - 2g(\bar{x})}{h^2}, \quad (3.34)$$

where h is the interval length or central difference step size and \bar{x} is the prior mean of x around which the expansion is done. One can thus interpret the Sterling interpolation formula as a Taylor series where the analytical derivatives are replaced by central divided differences. Extending this formulation to the multi-dimensional case, $\mathbf{g}(\mathbf{x})$, is achieved by first *stochastically decoupling* the prior random variable \mathbf{x} by the following linear transformation [171]

$$\mathbf{z} = \mathbf{S}_x^{-1}\mathbf{x} \quad (3.35)$$

$$\tilde{\mathbf{g}}(\mathbf{z}) \doteq \mathbf{g}(\mathbf{S}_x\mathbf{z}) = \mathbf{g}(\mathbf{x}) \quad (3.36)$$

where \mathbf{S}_x is the Cholesky factor of the covariance matrix of \mathbf{x} , \mathbf{P}_x , such that

$$\mathbf{P}_x = \mathbf{S}_x \mathbf{S}_x^T. \quad (3.37)$$

Here we assume that the random variable \mathbf{x} has mean $\bar{\mathbf{x}} = E[\mathbf{x}]$ and covariance $\mathbf{P}_x = E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T]$. Note that the Taylor series expansion of $\mathbf{g}(\cdot)$ and $\tilde{\mathbf{g}}(\cdot)$ will be identical. This transformation stochastically decouples the variables in \mathbf{x} in that the individual components of \mathbf{z} becomes mutually uncorrelated (with unity variance), i.e.,

$$\mathbf{P}_z = E[(\mathbf{z} - \bar{\mathbf{z}})(\mathbf{z} - \bar{\mathbf{z}})^T] = \mathbf{I}. \quad (3.38)$$

This allows for the application of the scalar central differencing operations (Equations 3.33 and 3.34) independently to the components of $\tilde{\mathbf{g}}(\mathbf{z})$ in order to obtain the following multi-dimensional central difference operators:

$$\tilde{\mathbf{D}}_{\Delta_{\mathbf{z}}} \tilde{\mathbf{g}} = \left(\sum_{i=1}^L \Delta_{\mathbf{z}_i} \mathbf{m}_i \mathbf{d}_i \right) \tilde{\mathbf{g}}(\bar{\mathbf{z}}) \quad (3.39)$$

$$\tilde{\mathbf{D}}_{\Delta_{\mathbf{z}}}^2 \tilde{\mathbf{g}} = \left(\sum_{i=1}^L \Delta_{\mathbf{z}_i}^2 \mathbf{d}_i^2 + \sum_{j=1}^L \sum_{\substack{q=1 \\ q \neq j}}^L \Delta_{\mathbf{z}_j} \Delta_{\mathbf{z}_q} (\mathbf{m}_j \mathbf{d}_j) (\mathbf{m}_q \mathbf{d}_q) \right) \tilde{\mathbf{g}}(\bar{\mathbf{z}}), \quad (3.40)$$

where $\Delta_{\mathbf{z}_i} = (\mathbf{z} - \bar{\mathbf{z}})_i$ is the i th component of $\mathbf{z} - \bar{\mathbf{z}}$ ($i = 1, \dots, L$), L is the dimension of \mathbf{x} (and thus \mathbf{z}), and \mathbf{d}_i , \mathbf{m}_i and \mathbf{d}_i^2 are the “partial” first order difference and mean operators and “partial” second order difference operator resp. defined as:

$$\mathbf{d}_i \tilde{\mathbf{g}}(\bar{\mathbf{z}}) = \frac{1}{2h} [\tilde{\mathbf{g}}(\bar{\mathbf{z}} + h\mathbf{e}_i) - \tilde{\mathbf{g}}(\bar{\mathbf{z}} - h\mathbf{e}_i)] \quad (3.41)$$

$$\mathbf{m}_i \tilde{\mathbf{g}}(\bar{\mathbf{z}}) = \frac{1}{2} [\tilde{\mathbf{g}}(\bar{\mathbf{z}} + h\mathbf{e}_i) + \tilde{\mathbf{g}}(\bar{\mathbf{z}} - h\mathbf{e}_i)] \quad (3.42)$$

$$\mathbf{d}_i^2 \tilde{\mathbf{g}}(\bar{\mathbf{z}}) = \frac{1}{2h^2} [\tilde{\mathbf{g}}(\bar{\mathbf{z}} + h\mathbf{e}_i) + \tilde{\mathbf{g}}(\bar{\mathbf{z}} - h\mathbf{e}_i) - 2\tilde{\mathbf{g}}(\bar{\mathbf{z}})] \quad (3.43)$$

where \mathbf{e}_i is the i th unit vector.

As an aside, using Equation 3.35 and 3.36, we can show that

$$\tilde{\mathbf{g}}(\bar{\mathbf{z}} \pm h\mathbf{e}_i) = \mathbf{g}(\mathbf{S}_{\mathbf{x}}[\bar{\mathbf{z}} \pm h\mathbf{e}_i]) \quad (3.44)$$

$$= \mathbf{g}(\mathbf{S}_{\mathbf{x}}\bar{\mathbf{z}} \pm h\mathbf{S}_{\mathbf{x}}\mathbf{e}_i) \quad (3.45)$$

$$= \mathbf{g}(\bar{\mathbf{x}} \pm h\mathbf{s}_{\mathbf{x}_i}) \quad (3.46)$$

where $\mathbf{s}_{\mathbf{x}_i}$ is the i th column of the square Cholesky factor of the covariance matrix of \mathbf{x} , i.e.,

$$\mathbf{s}_{\mathbf{x}_i} = \mathbf{S}_{\mathbf{x}}\mathbf{e}_i = (\mathbf{S}_{\mathbf{x}})_i = \left(\sqrt{\mathbf{P}_{\mathbf{x}}} \right)_i. \quad (3.47)$$

It is important to note here for future reference, that the set of vectors defined in Equation 3.46 by $\bar{\mathbf{x}} \pm h\mathbf{s}_{\mathbf{x}_i}$, is equivalent in form to the way that the UKF generates its set of sigma-points in Equation 3.2, with the only difference being the value of the weighting term.

3.3.2 Estimation of posterior mean and covariance by Sterling interpolation

As before, we consider the nonlinear transformation of a L dimensional random variable \mathbf{x} with mean $\bar{\mathbf{x}}$ and covariance \mathbf{P}_x through an arbitrary nonlinear function $\mathbf{g}(\cdot)$, i.e.,

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) . \quad (3.48)$$

Using the 2nd order multi-dimensional Sterling interpolation expansion of Equation 3.48,

$$\begin{aligned} \mathbf{y} &= \mathbf{g}(\mathbf{x}) \\ &= \tilde{\mathbf{g}}(\mathbf{z}) \\ &\approx \tilde{\mathbf{g}}(\bar{\mathbf{z}}) + \tilde{\mathbf{D}}_{\Delta z} \tilde{\mathbf{g}} + \frac{1}{2} \tilde{\mathbf{D}}_{\Delta z}^2 \tilde{\mathbf{g}} , \end{aligned} \quad (3.49)$$

where $\bar{\mathbf{z}} = \mathbf{S}_x \bar{\mathbf{x}}$, we will now proceed to approximate the posterior mean, covariance and cross-covariance of \mathbf{y} ,

$$\bar{\mathbf{y}} = E[\mathbf{y}] \quad (3.50)$$

$$\mathbf{P}_y = E[(\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^T] \quad (3.51)$$

$$\mathbf{P}_{xy} = E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{y} - \bar{\mathbf{y}})^T] \quad (3.52)$$

in terms of the prior statistics of \mathbf{x} , $\bar{\mathbf{x}}$ and \mathbf{P}_x . This is done by replacing \mathbf{y} in Equations 3.50 - 3.52 by its 2nd order Sterling approximation (Equation 3.49) and carrying out the expectation operations as necessary.

The **mean** approximation is given by

$$\bar{\mathbf{y}} \approx E \left[\tilde{\mathbf{g}}(\bar{\mathbf{z}}) + \tilde{\mathbf{D}}_{\Delta_z} \tilde{\mathbf{g}} + \frac{1}{2} \tilde{\mathbf{D}}_{\Delta_z}^2 \tilde{\mathbf{g}} \right] \quad (3.53)$$

$$= \tilde{\mathbf{g}}(\bar{\mathbf{z}}) + E \left[\frac{1}{2} \tilde{\mathbf{D}}_{\Delta_z}^2 \tilde{\mathbf{g}} \right] \quad (3.54)$$

$$= \tilde{\mathbf{g}}(\bar{\mathbf{z}}) + E \left[\frac{1}{2h^2} \left(\sum_{i=1}^L \Delta_{z_i}^2 \mathbf{d}_i^2 \right) \tilde{\mathbf{g}}(\bar{\mathbf{z}}) \right] \quad (3.55)$$

$$= \tilde{\mathbf{g}}(\bar{\mathbf{z}}) + \frac{1}{2h^2} \sum_{i=1}^L [\tilde{\mathbf{g}}(\bar{\mathbf{z}} + h\mathbf{e}_i) + \tilde{\mathbf{g}}(\bar{\mathbf{z}} - h\mathbf{e}_i) - 2\tilde{\mathbf{g}}(\bar{\mathbf{z}})] \quad (3.56)$$

$$= \frac{h^2 - L}{h^2} \tilde{\mathbf{g}}(\bar{\mathbf{z}}) + \frac{1}{2h^2} \sum_{i=1}^L [\tilde{\mathbf{g}}(\bar{\mathbf{z}} + h\mathbf{e}_i) + \tilde{\mathbf{g}}(\bar{\mathbf{z}} - h\mathbf{e}_i)] , \quad (3.57)$$

where we assumed $\Delta_z = \mathbf{z} - \bar{\mathbf{z}}$ is a zero-mean unity variance symmetric random variable as defined in Equation 3.35. Substituting Equation 3.46 into Equation 3.57, we can rewrite the posterior mean approximation in terms of the prior statistics of \mathbf{x} :

$$\bar{\mathbf{y}} \approx \frac{h^2 - L}{h^2} \mathbf{g}(\bar{\mathbf{x}}) + \frac{1}{2h^2} \sum_{i=1}^L [\mathbf{g}(\bar{\mathbf{x}} + h\mathbf{s}_{x_i}) + \mathbf{g}(\bar{\mathbf{x}} - h\mathbf{s}_{x_i})] . \quad (3.58)$$

For the calculation of the **covariance** approximation, the following expansion of Equation 3.51 is used:

$$\mathbf{P}_y = E \left[(\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^T \right] \quad (3.59)$$

$$= E \left[(\mathbf{y} - \mathbf{g}(\bar{\mathbf{x}}))(\mathbf{y} - \mathbf{g}(\bar{\mathbf{x}}))^T \right] - E[\mathbf{y} - \mathbf{g}(\bar{\mathbf{x}})] E[\mathbf{y} - \mathbf{g}(\bar{\mathbf{x}})]^T \quad (3.60)$$

$$= E \left[(\mathbf{y} - \tilde{\mathbf{g}}(\bar{\mathbf{z}}))(\mathbf{y} - \tilde{\mathbf{g}}(\bar{\mathbf{z}}))^T \right] - E[\mathbf{y} - \tilde{\mathbf{g}}(\bar{\mathbf{z}})] E[\mathbf{y} - \tilde{\mathbf{g}}(\bar{\mathbf{z}})]^T \quad (3.61)$$

where we used the identity

$$\begin{aligned} \bar{\mathbf{y}} &= E[\mathbf{y}] \\ &= E[\mathbf{y}] + \mathbf{g}(\bar{\mathbf{x}}) - \mathbf{g}(\bar{\mathbf{x}}) \\ &= E[\mathbf{y}] + \mathbf{g}(\bar{\mathbf{x}}) - E[\mathbf{g}(\bar{\mathbf{x}})] \\ &= \mathbf{g}(\bar{\mathbf{x}}) + E[\mathbf{y} - \mathbf{g}(\bar{\mathbf{x}})] , \end{aligned} \quad (3.62)$$

in the expansion of Equation 3.59. From Equation 3.49 we have

$$\mathbf{y} - \tilde{\mathbf{g}}(\bar{\mathbf{z}}) = \tilde{\mathbf{D}}_{\Delta_z} \tilde{\mathbf{g}} + \frac{1}{2} \tilde{\mathbf{D}}_{\Delta_z}^2 \tilde{\mathbf{g}}, \quad (3.63)$$

which is the 2nd order Stirling interpolation approximation of $\mathbf{y} - \tilde{\mathbf{g}}(\bar{\mathbf{z}})$. Substituting this result into Equation 3.61 gives us the following approximation:

$$\begin{aligned} \mathbf{P}_{\mathbf{y}} &\approx E \left[\left(\tilde{\mathbf{D}}_{\Delta_z} \tilde{\mathbf{g}} + \frac{1}{2} \tilde{\mathbf{D}}_{\Delta_z}^2 \tilde{\mathbf{g}} \right) \left(\tilde{\mathbf{D}}_{\Delta_z} \tilde{\mathbf{g}} + \frac{1}{2} \tilde{\mathbf{D}}_{\Delta_z}^2 \tilde{\mathbf{g}} \right)^T \right] \\ &\quad - E \left[\tilde{\mathbf{D}}_{\Delta_z} \tilde{\mathbf{g}} + \frac{1}{2} \tilde{\mathbf{D}}_{\Delta_z}^2 \tilde{\mathbf{g}} \right] E \left[\tilde{\mathbf{D}}_{\Delta_z} \tilde{\mathbf{g}} + \frac{1}{2} \tilde{\mathbf{D}}_{\Delta_z}^2 \tilde{\mathbf{g}} \right]^T. \end{aligned} \quad (3.64)$$

As for the derivation of the mean, we assume that $\Delta_z = \mathbf{z} - \bar{\mathbf{z}}$ is a zero-mean unity variance symmetric random variable as defined in Equation 3.35. Due to the symmetry, all resulting odd-order expected moments can be equated to zero. Furthermore, in order to keep the results computationally tractable, we will discard all components of the resulting fourth order term, $E \left[\frac{1}{4} \tilde{\mathbf{D}}_{\Delta_z}^2 \tilde{\mathbf{g}} \left(\tilde{\mathbf{D}}_{\Delta_z}^2 \tilde{\mathbf{g}} \right)^T \right]$, that contains cross-differences in the expansion of Equation 3.64. This is done because inclusion of these terms leads to an excessive increase in the number of computations as the number of such terms grows rapidly with the dimension of \mathbf{z} . The reason for not considering the extra effort worthwhile is that we are unable to capture all fourth order moments anyway [147, 148]. Carrying out the expansion and expectation operations⁶ on Equation 3.64 under the assumptions just stated, results in the following approximations of the covariance:

$$\begin{aligned} \mathbf{P}_{\mathbf{y}} &\approx \frac{1}{4h^2} \sum_{i=1}^L [\mathbf{g}(\bar{\mathbf{x}} + h\mathbf{s}_{\mathbf{x}_i}) - \mathbf{g}(\bar{\mathbf{x}} - h\mathbf{s}_{\mathbf{x}_i})] [\mathbf{g}(\bar{\mathbf{x}} + h\mathbf{s}_{\mathbf{x}_i}) - \mathbf{g}(\bar{\mathbf{x}} - h\mathbf{s}_{\mathbf{x}_i})]^T \\ &\quad + \frac{h^2 - 1}{4h^4} \sum_{i=1}^L [\mathbf{g}(\bar{\mathbf{x}} + h\mathbf{s}_{\mathbf{x}_i}) + \mathbf{g}(\bar{\mathbf{x}} - h\mathbf{s}_{\mathbf{x}_i}) - 2\mathbf{g}(\bar{\mathbf{x}})] \times \\ &\quad \times [\mathbf{g}(\bar{\mathbf{x}} + h\mathbf{s}_{\mathbf{x}_i}) + \mathbf{g}(\bar{\mathbf{x}} - h\mathbf{s}_{\mathbf{x}_i}) - 2\mathbf{g}(\bar{\mathbf{x}})]^T \end{aligned} \quad (3.65)$$

⁶The complete derivation, which from this point on consists of a couple of pages of tedious algebra and careful bookkeeping, is not presented here for the sake of brevity. The interested reader is referred to [148] for a complete exposition.

In a similar fashion, the cross-covariance approximation is given by:

$$\mathbf{P}_{\mathbf{x}\mathbf{y}} = E \left[(\mathbf{x} - \bar{\mathbf{x}}) (\mathbf{y} - \bar{\mathbf{y}})^T \right] \quad (3.66)$$

$$\approx E \left[(\mathbf{S}_x (\mathbf{z} - \bar{\mathbf{z}})) \left(\tilde{\mathbf{D}}_{\Delta z} \tilde{\mathbf{g}} + \frac{1}{2} \tilde{\mathbf{D}}_{\Delta z}^2 \tilde{\mathbf{g}} - E \left[\frac{1}{2} \tilde{\mathbf{D}}_{\Delta z}^2 \tilde{\mathbf{g}} \right] \right)^T \right] \quad (3.67)$$

$$\begin{aligned} &= E \left[(\mathbf{S}_x \Delta_z) \left(\tilde{\mathbf{D}}_{\Delta z} \tilde{\mathbf{g}} \right)^T \right] + \frac{1}{2} E \left[(\mathbf{S}_x \Delta_z) \left(\tilde{\mathbf{D}}_{\Delta z}^2 \tilde{\mathbf{g}} \right)^T \right] - \\ &\quad - \frac{1}{2} E [(\mathbf{S}_x \Delta_z)] E \left[\frac{1}{2} \tilde{\mathbf{D}}_{\Delta z}^2 \tilde{\mathbf{g}} \right]^2 \end{aligned} \quad (3.68)$$

$$= E \left[(\mathbf{S}_x \Delta_z) \left(\tilde{\mathbf{D}}_{\Delta z} \tilde{\mathbf{g}} \right)^T \right] \quad (3.69)$$

$$= \frac{1}{2h} \sum_{i=1}^L \mathbf{s}_{x_i} [\tilde{\mathbf{g}}(\bar{\mathbf{z}} + h\mathbf{e}_i) - \tilde{\mathbf{g}}(\bar{\mathbf{z}} - h\mathbf{e}_i)]^T \quad (3.70)$$

$$= \frac{1}{2h} \sum_{i=1}^L \mathbf{s}_{x_i} [\mathbf{g}(\bar{\mathbf{x}} + h\mathbf{s}_{x_i}) - \mathbf{g}(\bar{\mathbf{x}} - h\mathbf{s}_{x_i})]^T , \quad (3.71)$$

where the odd-order moment terms of Equation 3.68 equate to zero due to the symmetry of Δ_z .

The optimal setting of the central difference interval parameter, h , is dictated by the prior distribution of $\mathbf{z} = \mathbf{S}_x^{-1}\mathbf{x}$. It turns out that h^2 should equal the kurtosis of \mathbf{z} to minimize errors between the Taylor series expansion of the true mean and covariance and those of the estimates [147]. For Gaussian priors, the optimal value of h is thus $h = \sqrt{3}$. (Note: At this point one might ask what the differences and similarities are between Sterling's approximation method and the UKF's unscented transformation? This will be covered in detail in Sections 3.4 and 4.2.)

Now that we can approximate the posterior statistics of a nonlinearly transformed random variable using the Sterling interpolation method, we are ready to show how this technique is used inside the actual algorithmic implementation of the central difference Kalman filter (CDKF). Before that though, we will present a brief demonstration of the Sterling interpolation method for posterior statistics estimation.

3.3.3 Demonstration of Sterling interpolation for posterior statistics estimation

For this example we revisit the *polar to Cartesian coordinate transformation* experiment presented in Chapter 2. As described in Section 2.5.3, this ubiquitous transformation lies at the heart of the observation models for many real-world sensors such as sonar, radar and laser range finders. A sensor returns polar measurement information that are related through a nonlinear transformation with the underlying Cartesian position of a target, i.e

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r \cos \theta \\ r \sin \theta \end{bmatrix},$$

where (r, θ) are the range and bearing angle of the target in the polar coordinate frame and (x, y) are the normal 2-dimensional Cartesian coordinates of the target. Linearizing this nonlinear observation model using a first order Taylor series truncation, quickly becomes very inaccurate for any significant bearing error. The results of this experiment is shown in Figure 3.4. The true position of the target is at $(0, 1)$ in the Cartesian plane. Several thousand measurement samples were generated by taking the true range and bearing (r, θ) value of the target location and adding zero-mean Gaussian noise to both the r and the θ components, and then converting back to the Cartesian plane. The resulting distribution (in the Cartesian plane) has a characteristic “banana”-shape, clearly indicating the better accuracy in the range direction than the bearing direction. The true mean and covariance of this distribution is indicated by the black dot and covariance ellipse. The mean and covariance calculated by a simple first order linearization approach (as used in the EKF, Equations 2.25 and 2.26) are indicated by the magenta dot and ellipse. The Sterling approximation calculated mean and covariance (using Equations 3.58 and 3.65) are indicated in green. The linearization based solution is clearly inaccurate. There is a large bias in the mean estimate in the range direction and the covariance is also underestimated in this direction. The underestimation results in an estimated posterior distribution that is too peaked (compared to the true posterior) in the range direction, resulting in “over confidence” in the resulting mean position estimate. In comparison, the sigma-point estimate

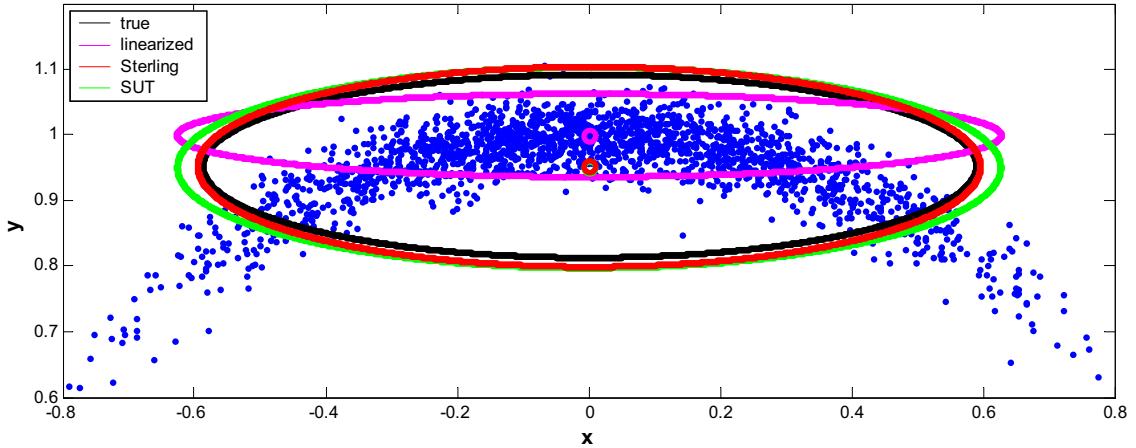


Figure 3.4: Optimal vs. linearized (EKF) vs. Sterling approximation (CDKF) vs. SUT (UKF) approximation for estimating the statistics of a Gaussian random variable that undergoes a polar-to-Cartesian coordinate transformation. The blue dots indicate the noisy polar coordinate measurements coming from a sensor such as a laser or sonar range finder. The true covariance and mean of this distribution is indicated by the black dot and ellipse. The magenta dots and ellipse indicate the results using normal linearization. Red and green indicate the results using Sterling approximation and SUT calculated means and covariances respectively.

has almost no bias error in the mean and the covariance estimate seems to be more accurate, consistent and efficient. Clearly, a recursive navigation filter based on the Sterling approximation approach will outperform a comparable EKF solution not only in accuracy but also in robustness. Since the Sterling approximation calculated covariance tends to be more consistent than the linearization result, the resulting CDKF will have a much smaller tendency (if at all) to diverge. Figure 3.4 also indicates the results obtained using the SUT. The mean estimates resulting from Sterling approximation and the SUT are indistinguishable. The SUT's covariance estimate seems to be slightly more conservative than the Sterling result. Both the SUT and Sterling approximation clearly outperforms the simple linearization approach however. Repeating the 2D nonlinear mapping example of Section 3.2.2 gives almost identical results for the SUT and Sterling approximation. In Section 3.5.1 we present a direct comparative experiment between the UKF and the CDKF which will again show that for all practical purposes there are no difference in estimation performance between the UKF (SUT) and the CDKF (Sterling approximation).

3.3.4 Implementing the central difference Kalman filter

The central difference Kalman filter (CDKF) is a straightforward application of Sterling interpolation for posterior statistics approximation, to the recursive Kalman filter framework as presented in Section 2.2 of the previous chapter. Specifically, Equations 3.58, 3.65 and 3.71 are used to approximate the optimal terms in Equations 2.5-2.8 of the optimal Gaussian approximate linear Bayesian update. The complete CDKF algorithm that updates the mean $\hat{\mathbf{x}}_k$ and covariance $\mathbf{P}_{\mathbf{x}_k}$ of the Gaussian approximation to the posterior distribution of the states will now be presented:

Algorithm 3 : The Central Difference Kalman Filter (CDKF)

- *Initialization:*

$$\begin{aligned}\hat{\mathbf{x}}_0 &= E[\mathbf{x}_0] & \mathbf{P}_{\mathbf{x}_0} &= E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T] \\ \bar{\mathbf{v}} &= E[\mathbf{v}] & \mathbf{R}_{\mathbf{v}} &= E[(\mathbf{v} - \bar{\mathbf{v}})(\mathbf{v} - \bar{\mathbf{v}})^T] \\ \bar{\mathbf{n}} &= E[\mathbf{n}] & \mathbf{R}_{\mathbf{n}} &= E[(\mathbf{n} - \bar{\mathbf{n}})(\mathbf{n} - \bar{\mathbf{n}})^T]\end{aligned}\quad (3.72)$$

- *For* $k = 1, \dots, \infty$:

1. Calculate covariance square-root column vectors for time-update:

$$\mathbf{s}_{k-1}^{x,i} = h \left(\sqrt{\mathbf{P}_{\mathbf{x}_{k-1}}} \right)_i \quad i = 1, \dots, L_x \quad (3.73)$$

$$\mathbf{s}_{k-1}^{v,i} = h \left(\sqrt{\mathbf{R}_{\mathbf{v}}} \right)_i \quad i = 1, \dots, L_v \quad (3.74)$$

2. Time-update equations:

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= \frac{h^2 - L_x - L_v}{h^2} \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}}, \mathbf{u}_{k-1}) \\ &\quad + \frac{1}{2h^2} \sum_{i=1}^{L_x} \left[\mathbf{f} \left(\hat{\mathbf{x}}_{k-1} + \mathbf{s}_{k-1}^{x,i}, \bar{\mathbf{v}}, \mathbf{u}_{k-1} \right) + \mathbf{f} \left(\hat{\mathbf{x}}_{k-1} - \mathbf{s}_{k-1}^{x,i}, \bar{\mathbf{v}}, \mathbf{u}_{k-1} \right) \right] \\ &\quad + \frac{1}{2h^2} \sum_{i=1}^{L_v} \left[\mathbf{f} \left(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}} + \mathbf{s}_{k-1}^{v,i}, \mathbf{u}_{k-1} \right) + \mathbf{f} \left(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}} - \mathbf{s}_{k-1}^{v,i}, \mathbf{u}_{k-1} \right) \right]\end{aligned}\quad (3.75)$$

$$\begin{aligned}
\mathbf{P}_{\mathbf{x}_k}^- &= \frac{1}{4h^2} \sum_{i=1}^{L_x} \left[\mathbf{f}(\hat{\mathbf{x}}_{k-1} + \mathbf{s}_{k-1}^{x,i}, \bar{\mathbf{v}}, \mathbf{u}_{k-1}) - \mathbf{f}(\hat{\mathbf{x}}_{k-1} - \mathbf{s}_{k-1}^{x,i}, \bar{\mathbf{v}}, \mathbf{u}_{k-1}) \right]^2 \\
&\quad + \frac{1}{4h^2} \sum_{i=1}^{L_v} \left[\mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}} + \mathbf{s}_{k-1}^{v,i}, \mathbf{u}_{k-1}) - \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}} - \mathbf{s}_{k-1}^{v,i}, \mathbf{u}_{k-1}) \right]^2 \\
&\quad + \frac{h^2 - 1}{4h^4} \sum_{i=1}^{L_x} \left[\mathbf{f}(\hat{\mathbf{x}}_{k-1} + \mathbf{s}_{k-1}^{x,i}, \bar{\mathbf{v}}, \mathbf{u}_{k-1}) + \mathbf{f}(\hat{\mathbf{x}}_{k-1} - \mathbf{s}_{k-1}^{x,i}, \bar{\mathbf{v}}, \mathbf{u}_{k-1}) \right. \\
&\quad \quad \left. - 2\mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}}, \mathbf{u}_{k-1}) \right]^2 \\
&\quad + \frac{h^2 - 1}{4h^4} \sum_{i=1}^{L_v} \left[\mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}} + \mathbf{s}_{k-1}^{v,i}, \mathbf{u}_{k-1}) + \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}} - \mathbf{s}_{k-1}^{v,i}, \mathbf{u}_{k-1}) \right. \\
&\quad \quad \left. - 2\mathbf{f}(\hat{\mathbf{x}}_{k-1}, \bar{\mathbf{v}}, \mathbf{u}_{k-1}) \right]^2
\end{aligned} \tag{3.76}$$

3. Calculate covariance square-root column vectors for measurement-update:

$$\mathbf{s}_k^{x,i} = h \left(\sqrt{\mathbf{P}_{\mathbf{x}_k}^-} \right)_i \quad i = 1, \dots, L_x \tag{3.77}$$

$$\mathbf{s}_k^{n,i} = h \left(\sqrt{\mathbf{R}_n} \right)_i \quad i = 1, \dots, L_n \tag{3.78}$$

4. Measurement-update equations:

$$\begin{aligned}
\hat{\mathbf{y}}_k^- &= \frac{h^2 - L_x - L_n}{h^2} \mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}}) \\
&\quad + \frac{1}{2h^2} \sum_{i=1}^{L_x} \left[\mathbf{h}(\hat{\mathbf{x}}_k^- + \mathbf{s}_k^{x,i}, \bar{\mathbf{n}}) + \mathbf{h}(\hat{\mathbf{x}}_k^- - \mathbf{s}_k^{x,i}, \bar{\mathbf{n}}) \right] \\
&\quad + \frac{1}{2h^2} \sum_{i=1}^{L_n} \left[\mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}} + \mathbf{s}_k^{n,i}) + \mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}} - \mathbf{s}_k^{n,i}) \right]
\end{aligned} \tag{3.79}$$

$$\mathbf{P}_{\tilde{\mathbf{y}}_k} = \frac{1}{4h^2} \sum_{i=1}^{L_x} \left[\mathbf{h}(\hat{\mathbf{x}}_k^- + \mathbf{s}_k^{x,i}, \bar{\mathbf{n}}) - \mathbf{h}(\hat{\mathbf{x}}_k^- - \mathbf{s}_k^{x,i}, \bar{\mathbf{n}}) \right]^2 \tag{3.80}$$

$$+ \frac{1}{4h^2} \sum_{i=1}^{L_n} \left[\mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}} + \mathbf{s}_k^{n,i}) - \mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}} - \mathbf{s}_k^{n,i}) \right]^2$$

$$+ \frac{h^2 - 1}{4h^4} \sum_{i=1}^{L_x} \left[\mathbf{h}(\hat{\mathbf{x}}_k^- + \mathbf{s}_k^{x,i}, \bar{\mathbf{n}}) + \mathbf{h}(\hat{\mathbf{x}}_k^- - \mathbf{s}_k^{x,i}, \bar{\mathbf{n}}) - 2\mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}}) \right]^2$$

$$+ \frac{h^2 - 1}{4h^4} \sum_{i=1}^{L_n} \left[\mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}} + \mathbf{s}_k^{n,i}) + \mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}} - \mathbf{s}_k^{n,i}) - 2\mathbf{h}(\hat{\mathbf{x}}_k^-, \bar{\mathbf{n}}) \right]^2$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \frac{1}{2h^2} \sum_{i=1}^{L_x} \mathbf{s}_k^{x,i} \left[\mathbf{h}(\hat{\mathbf{x}}_k^- + \mathbf{s}_k^{x,i}, \bar{\mathbf{n}}) - \mathbf{h}(\hat{\mathbf{x}}_k^- - \mathbf{s}_k^{x,i}, \bar{\mathbf{n}}) \right]^T \tag{3.81}$$

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\hat{\mathbf{y}}_k}^{-1} \quad (3.82)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (3.83)$$

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{P}_{\hat{\mathbf{y}}_k} \mathbf{K}_k^T \quad (3.84)$$

- *Parameters:* $h \geq 1$ is the scalar central difference interval size. For Gaussian \mathbf{x} , the optimal value is $h = \sqrt{3}$. L_x , L_v and L_n are the dimensions of the state, process noise and observation noise RVs. \mathbf{R}_v and \mathbf{R}_n are the process and observation noise covariances. $(\cdot)^2$ is shorthand for the vector outer product, *i.e.* $\mathbf{a}^2 \doteq \mathbf{a}\mathbf{a}^T$, and $(\sqrt{\mathbf{P}})_i$ is the i th column of the matrix square root of the square-symmetric matrix \mathbf{P} .
-

3.4 Sigma-Point Kalman Filters and the Sigma-point Approach

Now that we have presented the UKF and the CDKF, it is convenient to cast them into a single framework that allows us to group them (and other derived algorithms) into a single family of derivativeless Kalman filters for Gaussian approximate nonlinear estimation. We call this family of filters *sigma-point Kalman filters* (SPKF) and the underlying common framework the *sigma-point approach* for estimating the statistics of nonlinearly transformed Gaussian approximate random variables.

We will first describe the sigma-point approach and then show how both the UKF and the CDKF make use of it under a different set of implementational assumptions. After this we rewrite the algorithmic specification of the CDKF (as presented in Section 3.3.4) using the sigma-point approach in order to present both the UKF and the CDKF algorithms in the common sigma-point Kalman filter (SPKF) notation. Note, in Chapter 4 (Section 4.2) we provide an alternative interpretation of the sigma-point approach, based on weighted statistical linear regression (stochastic/statistical linearization), that provides further unifying insight into the the different SPKF filters.

3.4.1 The Sigma-point Approach

Notice how the arguments of all the nonlinear functional evaluations in Equations 3.58, 3.65 and 3.71, the core of the Stirling approximation approach, are all of the form, $\bar{\mathbf{x}} \pm h\mathbf{s}_{\mathbf{x}_i}$, for $i = 1, \dots, L$ (L is the dimension of \mathbf{x} , $\bar{\mathbf{x}}$ is the mean of \mathbf{x} and $\mathbf{s}_{\mathbf{x}_i}$ are the columns of the matrix square-root of the covariance of \mathbf{x}). As we already pointed out earlier, this forms a sigma-point set in exactly the same manner as was the case for the SUT used by the UKF. More specifically, the resulting *weighted sigma-point set* used by Sterling's interpolation formula (which forms the core of the CDKF), is again $2L + 1$ points given by the prior mean plus/minus the columns (or rows) of the scaled matrix square-root of the

prior covariance matrix, i.e.,

$$\begin{aligned}\mathcal{X}_0 &= \bar{\mathbf{x}} & w_0^{(m)} &= \frac{h^2-L}{h^2} \\ \mathcal{X}_i &= \bar{\mathbf{x}} + (h\sqrt{\mathbf{P}_{\mathbf{x}}})_i & i=1,\dots,L & \quad w_i^{(m)} = \frac{1}{2h^2} & i=1,\dots,2L \\ \mathcal{X}_i &= \bar{\mathbf{x}} - (h\sqrt{\mathbf{P}_{\mathbf{x}}})_i & i=L+1,\dots,2L & \quad w_i^{(c_1)} = \frac{1}{4h^2} & i=1,\dots,2L \\ & & & \quad w_i^{(c_2)} = \frac{h^2-1}{4h^4} & i=1,\dots,2L\end{aligned}\tag{3.85}$$

where L is the dimension of \mathbf{x} . Similar to the SUT, each sigma-point is propagated through the true nonlinear function

$$\mathcal{Y}_i = \mathbf{g}(\mathcal{X}_i) , \quad i = 0, \dots, 2L ,\tag{3.86}$$

to form the posterior sigma-point set, \mathcal{Y}_i . Using the above results, we can now rewrite the Sterling approximation estimates of the posterior mean, covariance and cross-covariance (Equations 3.58, 3.65 and 3.71) as:

$$\bar{\mathbf{y}} \approx \sum_{i=0}^{2L} w_i^{(m)} \mathcal{Y}_i\tag{3.87}$$

$$\begin{aligned}\mathbf{P}_{\mathbf{y}} &\approx \sum_{i=1}^L w_i^{(c_1)} [\mathcal{Y}_i - \mathcal{Y}_{i+L}] [\mathcal{Y}_i - \mathcal{Y}_{i+L}]^T \\ &\quad + \sum_{i=1}^L w_i^{(c_2)} [\mathcal{Y}_i + \mathcal{Y}_{i+L} - 2\mathcal{Y}_0] [\mathcal{Y}_i + \mathcal{Y}_{i+L} - 2\mathcal{Y}_0]^T\end{aligned}\tag{3.88}$$

$$\mathbf{P}_{\mathbf{xy}} \approx \sum_{i=1}^L w_i^{(m)} [\mathcal{X}_i - \bar{\mathbf{x}}] [\mathcal{Y}_i - \mathcal{Y}_{i+L}]^T .\tag{3.89}$$

This resulting set of equations for calculating of the statistics of a random variable that undergoes a nonlinear transformation are of the same general form as those employed by the SUT inside the UKF. Both approaches (SUT and Sterling approximation) are thus essentially the same and can be summarized by the following three steps which we refer to in general as the *sigma-point approach* for the approximating the statistics of a random variable that undergoes a nonlinear transformation:

The Sigma-point Approach

1. A set of weighted sigma-points are deterministically calculated using the mean and

square-root decomposition of the covariance matrix of the prior random variable. As a minimal requirement the sigma-point set must completely capture the first and second order moments of the prior random variable. Higher order moments can be captured, if so desired, at the cost of using more sigma-points.

2. *The sigma-points are then propagated through the true nonlinear function using functional evaluations alone, i.e., no analytical derivatives are used, in order to generate a posterior sigma-point set.*
3. *The posterior statistics are calculated (approximated) using tractable functions of the the propagated sigma-points and weights.*

By comparing Equations 3.14 and 3.87, we see that both the SUT and the Sterling approximation approach calculate the posterior mean in an identical fashion, with the only difference being the values of the scalar weights. The subtle difference between the two approaches, however, lie in the approximation of the posterior covariance term (Equations 3.15 and 3.88). The SUT makes use of the indirect form of the posterior covariance approximation:

$$\mathbf{P}_y = E \left[(\mathbf{y} - \bar{\mathbf{y}}) (\mathbf{y} - \bar{\mathbf{y}})^T \right] \quad (3.90)$$

$$\approx \sum_{i=0}^{2L} w_i^{(c)} (\mathbf{y}_i - \bar{\mathbf{y}}) (\mathbf{y}_i - \bar{\mathbf{y}})^T \quad (3.91)$$

$$= \sum_{i=0}^{2L} w_i^{(c)} \left(\mathbf{y}_i - \sum_{j=0}^{2L} w_j^{(m)} \mathbf{y}_j \right) \left(\mathbf{y}_i - \sum_{j=0}^{2L} w_j^{(m)} \mathbf{y}_j \right)^T \quad (3.92)$$

$$= \mathbf{P}_y^{SUT}, \quad (3.93)$$

That is, in Equation 3.91 the posterior mean ($\bar{\mathbf{y}}$) is first calculated using Equation 3.14, the result of which is then substituted into the calculation of the posterior covariance

(Equation 3.92). When this approximation is fully expanded,

$$\begin{aligned}\mathbf{P}_{\mathbf{y}}^{SUT} &= \sum_{i=0}^{2L} w_i^{(c)} \left(\mathbf{y}_i - \sum_{j=0}^{2L} w_j^{(m)} \mathbf{y}_j \right) \left(\mathbf{y}_i - \sum_{j=0}^{2L} w_j^{(m)} \mathbf{y}_j \right)^T \\ &= \sum_{i=0}^{2L} w_i^{(c)} \left[\mathbf{y}_i \mathbf{y}_i^T - \mathbf{y}_i \left(\sum_{j=0}^{2L} w_j^{(m)} \mathbf{y}_j^T \right) - \left(\sum_{j=0}^{2L} w_j^{(m)} \mathbf{y}_j \right) \mathbf{y}_i^T + \right. \\ &\quad \left. + \sum_{j=0}^{2L} \sum_{k=0}^{2L} w_j^{(m)} w_k^{(m)} \mathbf{y}_j \mathbf{y}_k^T \right] \end{aligned} \tag{3.94}$$

$$\begin{aligned}&= \sum_{i=0}^{2L} w_i^{(c)} \mathbf{y}_i \mathbf{y}_i^T - \sum_{i=0}^{2L} \sum_{j=0}^{2L} w_i^{(c)} w_j^{(m)} (\mathbf{y}_i \mathbf{y}_j^T + \mathbf{y}_j \mathbf{y}_i^T) + \\ &\quad + \sum_{j=0}^{2L} \sum_{k=0}^{2L} \tilde{\gamma} w_j^{(m)} w_k^{(m)} \mathbf{y}_j \mathbf{y}_k^T, \end{aligned} \tag{3.95}$$

where $\tilde{\gamma} = \sum_{i=0}^{2L} w_i^{(c)}$, we see that it contains terms with *all* off the posterior sigma-point cross-products, i.e., $\mathbf{y}_i \mathbf{y}_j^T$ for $i = 0, \dots, 2L$ and $j = 0, \dots, 2L$, present.

The Sterling approximation approach on the other hand, makes use of the direct form of the posterior covariance approximation (Equation 3.60):

$$\mathbf{P}_{\mathbf{y}} = E \left[(\mathbf{y} - \mathbf{g}(\bar{\mathbf{x}})) (\mathbf{y} - \mathbf{g}(\bar{\mathbf{x}}))^T \right] - E [\mathbf{y} - \mathbf{g}(\bar{\mathbf{x}})] E [\mathbf{y} - \mathbf{g}(\bar{\mathbf{x}})]^T \tag{3.96}$$

$$\begin{aligned}&\approx \sum_{i=1}^L w_i^{(c_1)} (\mathbf{y}_i - \mathbf{y}_{i+L}) (\mathbf{y}_i - \mathbf{y}_{i+L})^T + \\ &\quad + \sum_{i=1}^L w_i^{(c_2)} [\mathbf{y}_i + \mathbf{y}_{i+L} - 2\mathbf{y}_0] [\mathbf{y}_i + \mathbf{y}_{i+L} - 2\mathbf{y}_0]^T \end{aligned} \tag{3.97}$$

$$= \mathbf{P}_{\mathbf{y}}^{Sterling}, \tag{3.98}$$

That is, the covariance approximation is a direct function of the propagated sigma-points (\mathbf{y}_i) and not of the propagated sigma-point *and* the approximated posterior mean ($\bar{\mathbf{y}} \approx \sum_{i=0}^{2L} w_i^{(m)} \mathbf{y}_i$). If we now expand Equation 3.98 further,

$$\begin{aligned} \mathbf{P}_{\mathbf{y}}^{Sterling} &= \sum_{i=1}^L w_i^{(c_1)} (\mathbf{y}_i - \mathbf{y}_{i+L}) (\mathbf{y}_i - \mathbf{y}_{i+L})^T + \\ &\quad + \sum_{i=1}^L w_i^{(c_2)} [\mathbf{y}_i + \mathbf{y}_{i+L} - 2\mathbf{y}_0] [\mathbf{y}_i + \mathbf{y}_{i+L} - 2\mathbf{y}_0]^T \end{aligned} \quad (3.99)$$

$$\begin{aligned} &= \sum_{i=1}^L w_i^{(c_1)} [\mathbf{y}_i \mathbf{y}_i^T - \mathbf{y}_i \mathbf{y}_{i+L}^T - \mathbf{y}_{i+L} \mathbf{y}_i^T + \mathbf{y}_{i+L} \mathbf{y}_{i+L}^T] + \\ &\quad + \sum_{i=1}^L w_i^{(c_2)} [4\mathbf{y}_0 \mathbf{y}_0^T - 2\mathbf{y}_{i+L} \mathbf{y}_0^T - 2\mathbf{y}_i \mathbf{y}_0^T - 2\mathbf{y}_0 \mathbf{y}_{i+L}^T + \\ &\quad + \mathbf{y}_{i+L} \mathbf{y}_{i+L}^T + \mathbf{y}_i \mathbf{y}_{i+L}^T - 2\mathbf{y}_0 \mathbf{y}_i^T + \mathbf{y}_{i+L} \mathbf{y}_i^T + \mathbf{y}_i \mathbf{y}_i^T] \end{aligned} \quad (3.100)$$

$$\begin{aligned} &= \sum_{i=1}^{2L} 2w_i^{(c_2)} \mathbf{y}_0 \mathbf{y}_0^T - \sum_{i=1}^{2L} 2w_i^{(c_2)} \mathbf{y}_i \mathbf{y}_0^T - \sum_{i=1}^{2L} 2w_i^{(c_2)} \mathbf{y}_0 \mathbf{y}_i^T + \\ &\quad + \sum_{i=1}^{2L} w_i^{(c_1)} \mathbf{y}_i \mathbf{y}_i^T - \sum_{i=1}^L w_i^{(c_1)} (\mathbf{y}_i \mathbf{y}_{i+L}^T + \mathbf{y}_{i+L} \mathbf{y}_i^T) , \end{aligned} \quad (3.101)$$

we see that general forms of Equation 3.101 and Equation 3.95 are similar. The differences being only the value of the weights and the fact that Equation 3.99 (Sterling) does not use some of the sigma-point cross-product terms in comparison to Equation 3.95 (SUT). Specifically, the Sterling interpolation approximation discards the following cross-products

$$\{\mathbf{y}_i \mathbf{y}_j^T, i = 1, \dots, 2L, j = 1, \dots, 2L, i \neq j, |i - j| \neq L\} . \quad (3.102)$$

This is a direct result of the design choice to discard some of the higher order cross-product terms (to avoid excessive computational cost) in the derivation of the Sterling approximation estimate of the covariance (see Section 3.3.2).

Although this thesis only deals with the SUT and Sterling approximation as specific implementations of the sigma-point approach, it does not imply that other related implementational variations might not be derived. Julier has in subsequent work [96] extended the SUT to both a simplex version that uses half the number of sigma-points (at a reduced computational cost and accuracy), as well as a higher-order version that not only captures (and propagates) the mean and covariance of the prior random variables, but also the skew. Implementations of the sigma-point approach based on other polynomial

interpolation formulas such as *Padé approximants* [57, 37] are also conceivable and is an area of future research.

In Section 4.2 of Chapter 4, we present an alternative interpretation of the sigma-point approach, based on a statistical technique called *weighted statistical linear regression*, which allows for further useful insight into why the SPKF is expected to perform better and more robustly than the EKF.

3.4.2 Sigma-point Kalman filters

Sigma-point Kalman filters are the collective name used for derivativeless Kalman filters that employ the deterministic sampling based sigma-point approach to calculate approximations of the optimal terms of the Gaussian approximate linear Bayesian update rule (Equations 2.5, 2.6 and 2.8). The already presented UKF and CDKF algorithms thus fall within this family as the core members. Later in this chapter we will present further extensions of these filters which expands the SPKF family.

Comparing the algorithmic specifications of the UKF (Algorithm 2, Section 3.2.3) with that of the CDKF (Algorithm 3, Section 3.3.4), we see that the UKF is already specified in the general sigma-point approach based SPKF formulation. It is thus convenient at this point to recast the algorithmic specification of the CDKF into the similar more general SPKF framework.

Algorithm 4 : The Unscented Kalman Filter (UKF) - SPKF formulation

Identical to original formulation. See Algorithm 2 in Section 3.2.3.

Algorithm 5 : The Central Difference Kalman Filter (CDKF) - SPKF formulation

- *Initialization:* $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$, $\mathbf{P}_{\mathbf{x}_0} = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$
- *For* $k = 1, \dots, \infty$:

1. Calculate sigma-points for time-update:

$$\hat{\mathbf{x}}_{k-1}^{a_v} = [\hat{\mathbf{x}}_{k-1} \quad \bar{\mathbf{v}}] \quad , \quad \mathbf{P}_{k-1}^{a_v} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_{k-1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_v \end{bmatrix} \quad (3.103)$$

$$\boldsymbol{\chi}_{k-1}^{a_v} = \left[\hat{\mathbf{x}}_{k-1}^{a_v} \quad \hat{\mathbf{x}}_{k-1}^{a_v} + h\sqrt{\mathbf{P}_{k-1}^{a_v}} \quad \hat{\mathbf{x}}_{k-1}^{a_v} - h\sqrt{\mathbf{P}_{k-1}^{a_v}} \right] \quad (3.104)$$

2. Time-update equations:

$$\boldsymbol{\chi}_{k|k-1}^x = \mathbf{f}(\boldsymbol{\chi}_{k-1}^x, \boldsymbol{\chi}_{k-1}^v, \mathbf{u}_{k-1}) \quad (3.105)$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\chi}_{i,k|k-1}^x \quad (3.106)$$

$$\begin{aligned} \mathbf{P}_{\mathbf{x}_k}^- &= \sum_{i=1}^L \left[w_i^{(c_1)} \left(\boldsymbol{\chi}_{i,k|k-1}^x - \boldsymbol{\chi}_{L+i,k|k-1}^x \right)^2 + \right. \\ &\quad \left. w_i^{(c_2)} \left(\boldsymbol{\chi}_{i,k|k-1}^x + \boldsymbol{\chi}_{L+i,k|k-1}^x - 2\boldsymbol{\chi}_{0,k|k-1}^x \right)^2 \right] \end{aligned} \quad (3.107)$$

3. Calculate sigma-points for measurement-update:

$$\hat{\mathbf{x}}_{k|k-1}^{a_n} = [\hat{\mathbf{x}}_k^- \quad \bar{\mathbf{n}}] \quad , \quad \mathbf{P}_{k|k-1}^{a_n} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_k}^- & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_n \end{bmatrix} \quad (3.108)$$

$$\boldsymbol{\chi}_{k|k-1}^{a_n} = \left[\hat{\mathbf{x}}_{k|k-1}^{a_n} \quad \hat{\mathbf{x}}_{k|k-1}^{a_n} + h\sqrt{\mathbf{P}_{k|k-1}^{a_n}} \quad \hat{\mathbf{x}}_{k|k-1}^{a_n} - h\sqrt{\mathbf{P}_{k|k-1}^{a_n}} \right] \quad (3.109)$$

4. Measurement-update equations:

$$\boldsymbol{\gamma}_{k|k-1} = \mathbf{h}(\boldsymbol{\chi}_{k|k-1}^x, \boldsymbol{\chi}_{k|k-1}^n) \quad (3.110)$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\gamma}_{i,k|k-1} \quad (3.111)$$

$$\begin{aligned} \mathbf{P}_{\tilde{\mathbf{y}}_k}^- &= \sum_{i=1}^L \left[w_i^{(c_1)} \left(\boldsymbol{\gamma}_{i,k|k-1} - \boldsymbol{\gamma}_{L+i,k|k-1} \right)^2 + \right. \\ &\quad \left. w_i^{(c_2)} \left(\boldsymbol{\gamma}_{i,k|k-1} + \boldsymbol{\gamma}_{L+i,k|k-1} - 2\boldsymbol{\gamma}_{0,k|k-1} \right)^2 \right] \end{aligned} \quad (3.112)$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sqrt{w_1^{(c_1)} \mathbf{P}_{\mathbf{x}_k}^-} [\boldsymbol{\gamma}_{1:L,k|k-1} - \boldsymbol{\gamma}_{L+1:2L,k|k-1}]^T \quad (3.113)$$

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k}^{-1} \quad (3.114)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (3.115)$$

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k} \mathbf{K}_k^T \quad (3.116)$$

- Parameters: $\mathbf{x}^{a_v} = [\mathbf{x}^T \quad \mathbf{v}^T]^T$, $\boldsymbol{\chi}^{a_v} = [(\boldsymbol{\chi}^x)^T \quad (\boldsymbol{\chi}^v)^T]^T$, $\mathbf{x}^{a_n} = [\mathbf{x}^T \quad \mathbf{n}^T]^T$, $\boldsymbol{\chi}^{a_n} =$

$[(\mathbf{x}^x)^T \ (\mathbf{x}^n)^T]^T$, $h \geq 1$ is the scalar central difference step size, L is the dimension of the augmented states, \mathbf{R}_v is the process-noise covariance, \mathbf{R}_n is the observation-noise covariance, and w_i are the weights as calculated in Equation 3.85. $(\cdot)^2$ is shorthand for the vector outer product, i.e. $\mathbf{a}^2 \doteq \mathbf{a}\mathbf{a}^T$.

- *General note:* Here we again augment the system state with the process noise and observation noise vectors (\mathbf{v}_k and \mathbf{n}_k) as we did for the UKF. For the CDKF, however, we split this augmentation between the time-update and measurement-update, i.e., for the time-update the augmented state vector and augmented covariance matrix is given by

$$\mathbf{x}_k^{a_v} = \begin{bmatrix} \mathbf{x}_k^T & \mathbf{v}_k^T \end{bmatrix}^T, \quad \mathbf{P}_k^{a_v} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_k} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_v \end{bmatrix}, \quad (3.117)$$

and by

$$\mathbf{x}_k^{a_n} = \begin{bmatrix} \mathbf{x}_k^T & \mathbf{n}_k^T \end{bmatrix}^T, \quad \mathbf{P}_k^{a_n} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_k} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_n \end{bmatrix}, \quad (3.118)$$

for the measurement-update.

We've found in practice that the CDKF and UKF perform equally well with negligible difference in estimation accuracy. Both generate estimates however that are clearly superior to those calculated by an EKF. The performance similarity of the UKF and CDKF is clearly demonstrated on nonlinear time-series estimation problem in Section 3.5.1. However, there is one advantage the CDKF has over the UKF: The CDKF uses only a single scalar scaling parameter, the central difference interval size h , as opposed to the three (α, κ, β) that the UKF uses. Once again this parameter determines the *spread* of the sigma-points around the prior mean. As mentioned earlier, the optimal setting for h is equal to the kurtosis of the prior RV. For Gaussian RVs the optimal value is thus $h = \sqrt{3}$.

3.5 SPKF Application to State, Parameter and Dual estimation

In this section we demonstrate (and experimentally verify) the versatility of the SPKF as a general inference tool with specific application to state estimation, parameter estimation and dual estimation. We will show how the SPKF consistently outperforms the EKF on a variety of inference and machine learning problems.

3.5.1 SPKF State Estimation

The UKF (and CDKF) was originally designed for state estimation applied to nonlinear control applications requiring full-state feedback [94, 99, 100, 147]. We provide an example for an *inverted double pendulum* control system. In addition, we provide a new application example corresponding to *noisy time-series estimation with neural networks*⁷. Another original contribution reported here is the implementation of a *sigma-point Kalman smoother* with application to time-series estimation using a novel neural network based forward-backward predictor.

Noisy time-series estimation : In this example, two SPKFs (UKF and CDKF) are used to estimate an underlying clean time-series corrupted by additive Gaussian white noise. The time-series used is the Mackey-Glass-30 chaotic series [129, 110] which is described by the following continuous time differential equation

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-3)}{1+x(t-30)^{10}}, \quad (3.119)$$

where t is the continuous time variable and $x(t)$ is the time-series amplitude at time t . For this experiment, we modeled the discrete time version of this time series as a nonlinear autoregression

$$x_k = f(x_{k-1}, x_{k-2}, \dots, x_{k-M}; \mathbf{w}) + v_k, \quad (3.120)$$

where the model f (parameterized by \mathbf{w}) was approximated by training a feed-forward neural network on a sampled clean sequence generated by Equation 3.119. The residual error after convergence was taken to be the process noise variance, i.e., σ_v^2 . Next, white Gaussian noise was added to the clean Mackey-Glass series to generate a noisy time-series $y_k = x_k + n_k$. The corresponding state-space representation is given by

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k; \mathbf{w}) + \mathbf{B}v_k \quad (3.121)$$

$$y_k = \mathbf{C}\mathbf{x}_k + n_k \quad (3.122)$$

⁷See Haykin [74] or Bishop [18] for a thorough review of neural network theory.

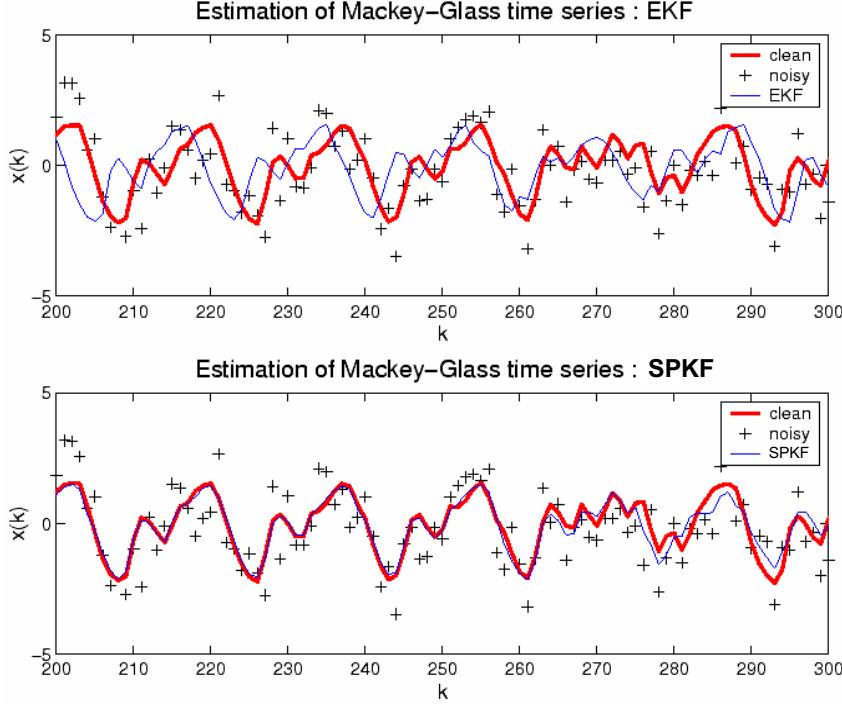


Figure 3.5: Estimation of Mackey-Glass time-series with the EKF and SPKF using a known model.

which can be expanded as

$$\begin{bmatrix} x_{k+1} \\ x_k \\ \vdots \\ x_{k-M+2} \end{bmatrix} = \begin{bmatrix} f(x_k, x_{k-1}, \dots, x_{k-M+1}; \mathbf{w}) \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ \vdots \\ x_{k-M+1} \end{bmatrix} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} v_k \quad (3.123)$$

$$y_k = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_k & x_{k-1} & \cdots & x_{k-M+1} \end{bmatrix}^T + n_k. \quad (3.124)$$

In the estimation problem, the noisy-time series y_k is the only observed input to either the EKF or SPKF algorithms (all utilize the known neural network model). Note that for time-series estimation, both the EKF and the SPKF are $\mathcal{O}(L^2)$ complexity. Figures 3.5 and 3.6 show sub-segments of the estimates generated by both the EKF and the SPKF (the original noisy time-series has a 3dB SNR). The superior performance of the SPKF algorithms are clearly visible. Table 3.1 summarizes the mean MSE as well as its variance for a Monte-Carlo run of 200 randomly initialized experiments. For each run a different

Table 3.1: Estimation of Mackey-Glass time-series with the EKF, UKF and CDKF using a known model. : Monte-Carlo averaged (200 runs) estimation error.

Algorithm	MSE (mean)	MSE (var)
Extended Kalman filter (EKF)	60.90	4.475e8
Unscented Kalman filter (UKF)	0.1115	0.0324
Central difference Kalman filter (CDKF)	0.1116	0.0357

realization of both the process and observation noise was generated. This results is also presented graphically in Figure 3.7, where the ensemble averaged (over all 200 runs) short-term MSE of the EKF and SPKF is plotted as a function of time (top plot). A 51 element median filter was used to smooth the resulting time-trajectories of the average MSE curves. As is clearly evident from both the table and the figure, the magnitude of the EKF errors is much larger (close to 2 orders of magnitude) than those of the SPKFs. The EKF not only has a worse average MSE performance, but the variance of the MSE is also extremely large. This is due to the fact that every once in while (for some runs) the EKF completely diverges for a significant number of samples, resulting in huge spikes in the estimates, as well as the resulting instantaneous MSE. The bottom plot of Figure 3.7 shows the distribution of the *log* of the Monte-Carlo averaged estimation errors for the EKF, UKF and CDKF using an unnormalized histogram presentation. Note the heavy tailed distribution for the EKF. This is due to the above mentioned “filter divergence” phenomenon where huge spikes in the EKF’s estimates occurs (these spikes can be seen in Figure 3.6). This skews the distribution (as well as the average MSE) to the right (larger errors). The almost indistinguishable performance of the UKF and CDKF is again evident.

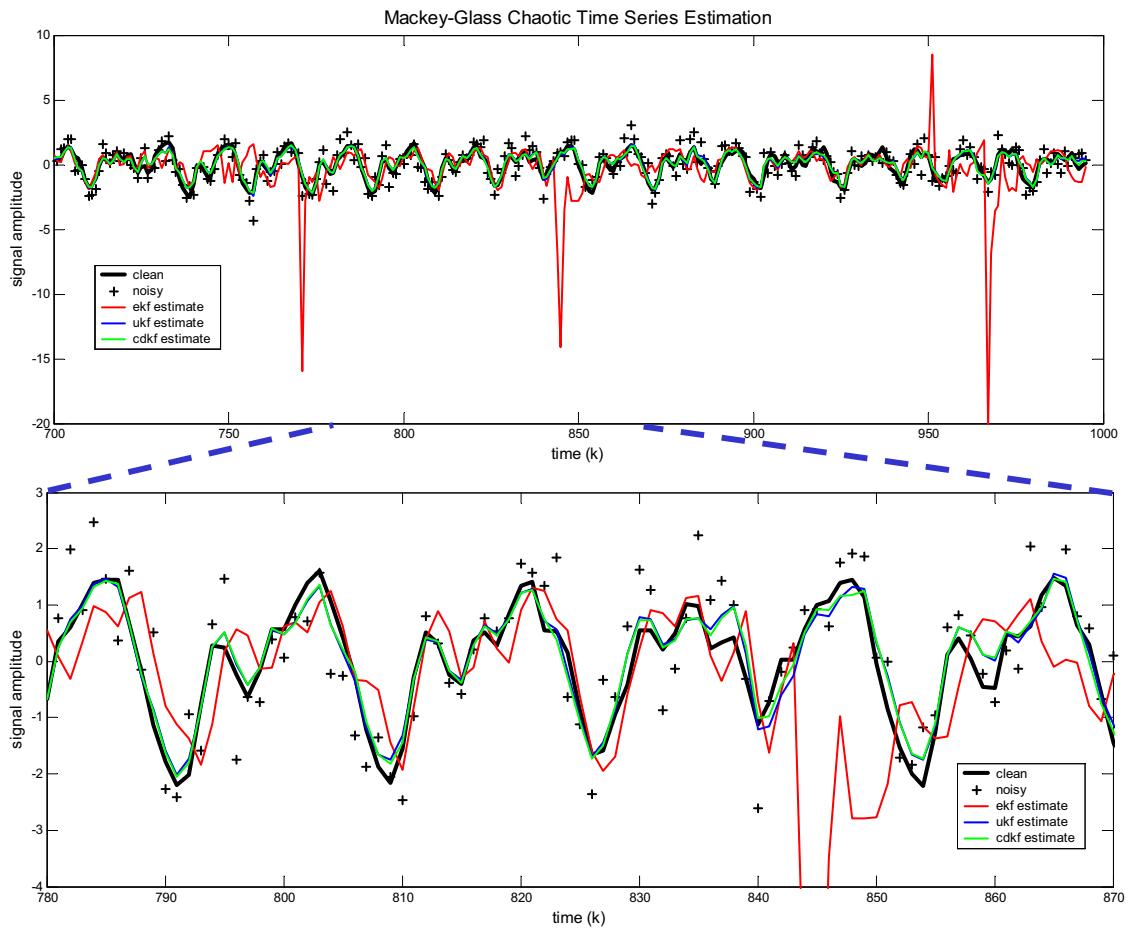


Figure 3.6: Estimation of Mackey-Glass time-series with the EKF and SPKF using a known model: (top) The plot shows a sub-segment of the underlying clean time-series as well as the noisy observed samples. The estimates generated by the EKF, UKF and CDKF are indicated. Note the large spikes in the estimates generated by the EKF. This is due (in part) to filter divergence. (bottom) This plot “zooms in” further on a section of the plot shown above. Note how the estimates generated by the two SPKF algorithms, the UKF and CDKF, are at the same time very close (similar) to each other and superior to those generated by the EKF.

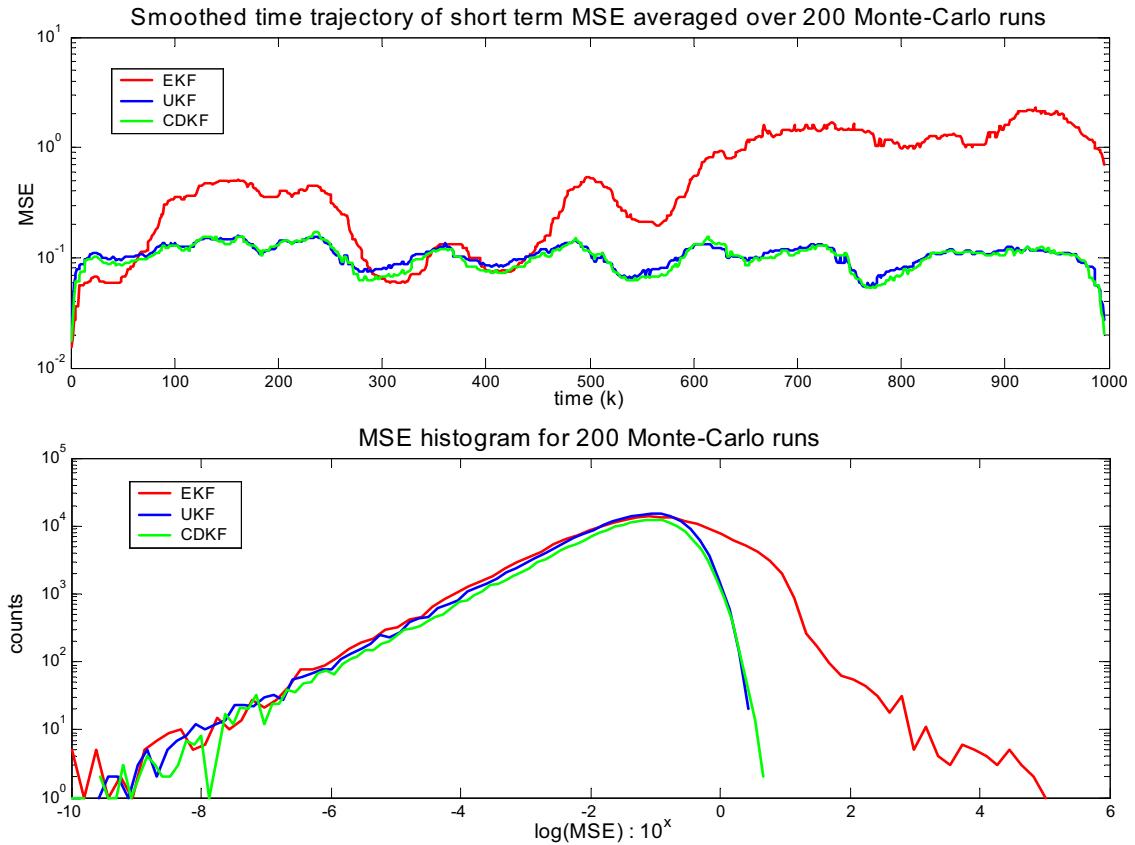


Figure 3.7: Monte-Carlo analysis of estimation error for Mackey-Glass nonlinear (chaotic) time-series estimation problem: These graphs show a comparison of averaged (over 200 Monte-Carlo runs) estimation error (MSE) for the complete sequence as generated by the EKF and SPKF algorithms. (top) This plot shows the smoothed (median filtered) Monte-Carlo averaged MSE trajectories for the different filters. Note the almost indistinguishable performance between the UKF and CDKF. (bottom) Histogram of estimation errors: This plot shows the distribution of the \log of the Monte-Carlo averaged estimation errors for the EKF, UKF and CDKF . Note the heavy tailed distribution for the EKF. This is due to the often occurring “filter divergence” phenomenon where huge spikes in the EKF’s estimates occurs. This skews the distribution (as well as the average MSE) to the right (larger errors). The almost indistinguishable performance of the UKF and CDKF is again evident .

Inverted Double pendulum: An inverted double pendulum (See Figure 3.8) has states corresponding to cart position and velocity, and top and bottom pendulum angle and angular velocity; and system parameters correspond the length and mass of each pendulum, and the cart mass:

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & \theta_1 & \dot{\theta}_1 & \theta_2 & \dot{\theta}_2 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} l_1 & l_2 & m_1 & m_2 & M \end{bmatrix} \quad (3.125)$$

The dynamic equations are

$$(M + m_1 + m_2)\ddot{x} - (m_1 + 2m_2)l_1\ddot{\theta}_1 \cos \theta_1 - l_2m_2\ddot{\theta}_2 \cos \theta_2 \quad (3.126)$$

$$\begin{aligned} &= u + (m_1 + 2m_2)l_1\dot{\theta}_1^2 \sin \theta_1 + m_2l_2\dot{\theta}_2^2 \sin \theta_2 \\ &- (m_1 + 2m_2)l_1\ddot{x} \cos \theta_1 + 4(\frac{m_1}{3} + m_2)l_1^2\ddot{\theta}_1 + 2m_2l_1l_2\ddot{\theta}_2 \cos(\theta_2 - \theta_1) \end{aligned} \quad (3.127)$$

$$\begin{aligned} &= (m_1 + 2m_2)gl_1 \sin \theta_1 + 2m_2l_1l_2\dot{\theta}_2^2 \sin(\theta_2 - \theta_1) \\ &- m_2\ddot{x}l_2 \cos \theta_2 + 2m_2l_1l_2\dot{\theta}_1 \cos(\theta_2 - \theta_1) + \frac{4}{3}m_2l_2^2\ddot{\theta}_2 \\ &= m_2gl_2 \sin \theta_2 - 2m_2l_1l_2\dot{\theta}_1^2 \sin(\theta_2 - \theta_1) \end{aligned} \quad (3.128)$$

These continuous-time dynamics are discretized with a sampling period of 0.02 seconds. The pendulum is stabilized by applying a control force, u to the cart. In this case we use a *state dependent Riccati equation (SDRE) controller* to stabilize the system ⁸. A state estimator is run outside the control loop in order to compare the EKF with the SPKF (i.e., the estimates states are not used in the feedback control for evaluation purposes). The observation corresponds to noisy measurements of the cart position, cart velocity, and angle of the top pendulum. This is a challenging problem, as no measurements are made for the bottom pendulum, nor for the angular velocity of the top pendulum. For this experiment, the pendulum is initialized in a jack-knife position (+25/-25 degrees) with a

⁸An SDRE controller [30] is designed by formulating the dynamic equations as $\mathbf{x}_{k+1} = \mathbf{A}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{B}(\mathbf{x}_k)\mathbf{u}_k$. Note, this representation is *not* a linearization, but rather a reformulation of the nonlinear dynamics into a pseudo-linear form. Based on this state-space representation, we design an optimal LQR controller, $\mathbf{u}_k = -\mathbf{R}^{-1}\mathbf{B}^T(\mathbf{x}_k)\mathbf{P}(\mathbf{x}_k)\mathbf{x}_k \equiv \mathbf{K}(\mathbf{x}_k)\mathbf{x}_k$, where $\mathbf{P}(\mathbf{x}_k)$ is a solution of the standard Riccati equations using state-dependent matrices $\mathbf{A}(\mathbf{x}_k)$ and $\mathbf{B}(\mathbf{x}_k)$. The procedure is repeated at every time step at the current state \mathbf{x}_k and provides local asymptotic stability of the plant [30]. The approach has been found to be far more robust than LQR controllers based on standard linearization techniques, and as well as many alternative “advanced” nonlinear control approaches.

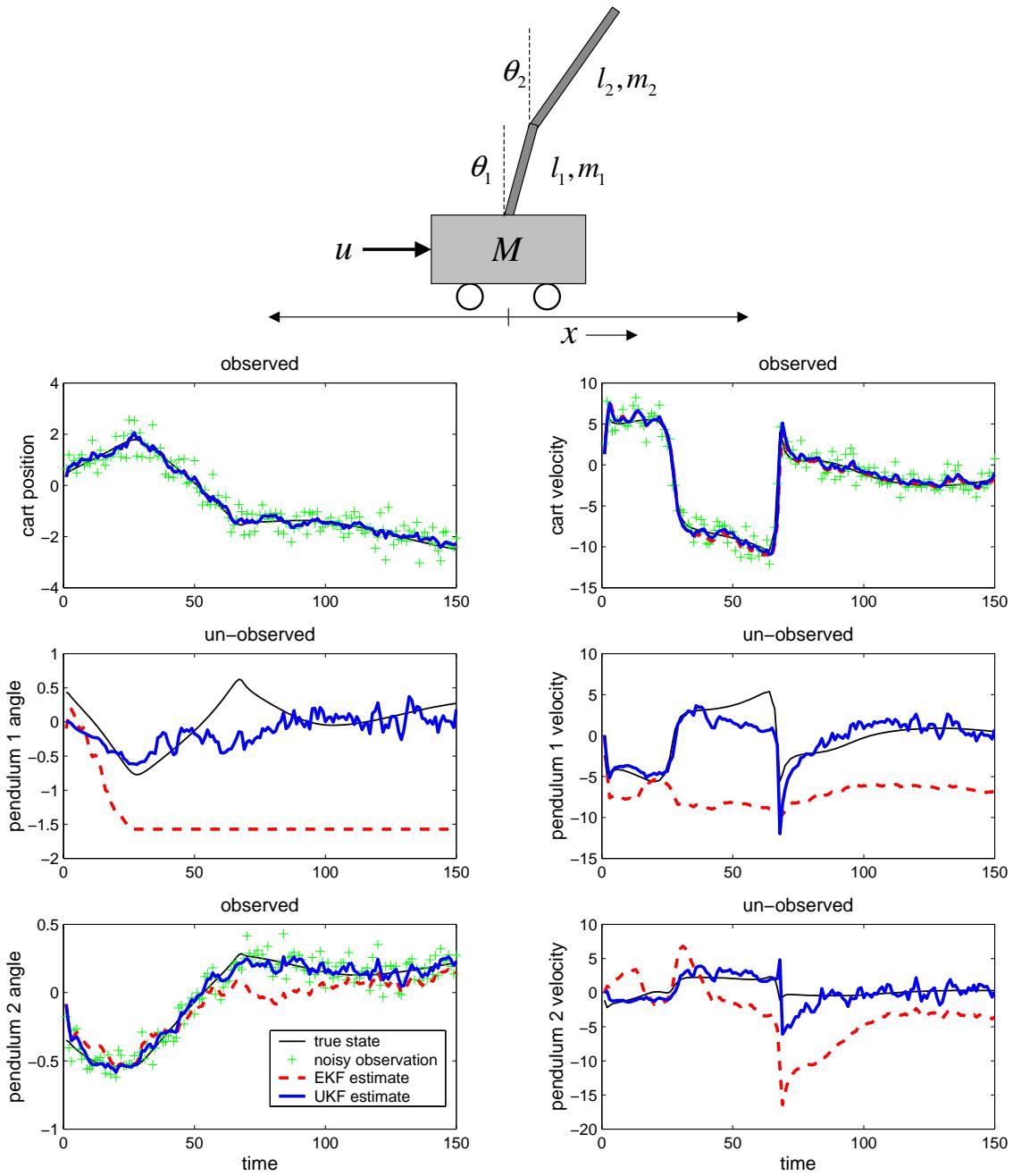


Figure 3.8: State estimation for double inverted pendulum problem: (top) Physical layout. (bottom) Estimation results. Only three noisy states are observed: cart position, cart velocity and the angle of the top pendulum. [10dB SNR]

cart offset of 0.5 meters. The resulting state estimates are shown in Figure 3.8. Clearly the SPKF is better able to track the unobserved states⁹. If the estimated states are used for feedback in the control loop, the SPKF system is still able to stabilize the pendulum, while the EKF system crashes. We will return to the double inverted pendulum problem later in this chapter for both model estimation and dual estimation.

3.5.2 SPKF Parameter Estimation

Recall that parameter estimation involves learning a nonlinear mapping $\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{w})$, where \mathbf{w} corresponds to the set of unknown parameters¹⁰. The nonlinear map $\mathbf{g}(\cdot)$, for example, may be a feed-forward or recurrent neural network (\mathbf{w} are the weights), with numerous applications in regression, classification, and dynamic modeling. It can also be a general nonlinear parametric model, such as the dynamic/kinematic vehicle model, parameterized by a finite set of coefficients. The Kalman filter framework may be used to estimate the parameters by writing a new state-space representation

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{r}_k \quad (3.129)$$

$$\mathbf{d}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{w}_k) + \mathbf{e}_k, \quad (3.130)$$

where \mathbf{w}_k correspond to a stationary process with identity state transition matrix, driven by process noise \mathbf{r}_k . The desired output \mathbf{d}_k corresponds to a nonlinear observation on \mathbf{w}_k .

From an optimization perspective, the following *prediction-error* cost is minimized:

$$J(\mathbf{w}) = \sum_{t=1}^k [\mathbf{d}_t - \mathbf{g}(\mathbf{x}_k, \mathbf{w})]^T (\mathbf{R}_e)^{-1} [\mathbf{d}_t - \mathbf{g}(\mathbf{x}_k, \mathbf{w})]. \quad (3.131)$$

Thus if the “noise” covariance \mathbf{R}_e is a constant diagonal matrix, then, in fact, it cancels out of the algorithm (this can be shown explicitly), and hence can be set arbitrarily (e.g., $\mathbf{R}_e = \frac{1}{2}\mathbf{I}$). Alternatively, \mathbf{R}_e can be set to specify a weighted MSE cost. The innovations covariance $E[\mathbf{r}_k \mathbf{r}_k^T] = \mathbf{R}_{\mathbf{r}_k}$, on the other hand, affects the convergence rate and tracking

⁹Note that if all 6 states are observed with noise, then the performance of the EKF and the SPKF is comparable.

¹⁰See Section 2.6.2 for an overview.

performance. Roughly speaking, the larger the covariance, the more quickly older data is discarded. There are several options on how to choose $\mathbf{R}_{\mathbf{r}_k}$:

- Set $\mathbf{R}_{\mathbf{r}_k}$ to an arbitrary “fixed” diagonal value, which may then be “annealed” toward zero as training continues.
- Set

$$\mathbf{R}_{\mathbf{r}_k} = (\lambda_{RLS}^{-1} - 1) \mathbf{P}_{\mathbf{w}_k}, \quad (3.132)$$

where $\lambda_{RLS} \in (0, 1]$ is often referred to as the “forgetting factor”, as defined in the recursive least-squares (RLS) algorithm [75]. This provides for an approximate exponentially decaying weighting on past data and is described in more detail in [143]. Note, λ_{RLS} should not be confused with the λ scaling parameter used for sigma point calculation in the UKF algorithm (Equation 3.11).

- Set

$$\mathbf{R}_{\mathbf{r}_k} = (1 - \alpha_{RM}) \mathbf{R}_{\mathbf{r}_{k-1}} + \alpha_{RM} \mathbf{K}_k [\mathbf{d}_k - \mathbf{g}(\mathbf{x}_k, \hat{\mathbf{w}}_k^-)] [\mathbf{d}_k - \mathbf{g}(\mathbf{x}_k, \hat{\mathbf{w}}_k^-)]^T \mathbf{K}_k^T, \quad (3.133)$$

which is a Robbins-Monro stochastic approximation scheme for estimating the innovations [124, 163, 183]. The method assumes the covariance of the Kalman update model should be consistent with the actual update model. Typically, $\mathbf{R}_{\mathbf{r}_k}$ is also constrained to be a diagonal matrix, which implies an independence assumption on the parameters. Note that a similar update may also be used for $\mathbf{R}_{\mathbf{e}_k}$, which will result in a time-varying (adaptive) weighting being applied to the cost function in Equation 3.131.

Our experience indicates that the *Robbins-Monro* method provides the fastest rate of absolute convergence and lowest final MMSE values (see experiments in the next section). The “fixed” $\mathbf{R}_{\mathbf{r}_k}$ in combination with annealing can also achieve good final MMSE performance, but requires more monitoring and a greater prior knowledge of the noise levels. For problems where the MMSE is zero, the covariance should be lower bounded to prevent the algorithm from stalling and potential numerical problems. The “forgetting-factor” and

“fixed” $\mathbf{R}_{\mathbf{r}_k}$ methods are most appropriate for on-line learning problems in which tracking of time varying parameters is necessary. In this case, the parameter covariance stays lower bounded, allowing the most recent data to be emphasized. This leads to some misadjustment, but also keeps the Kalman gain sufficiently large to maintain good tracking. In general, the various trade-offs between these different approaches is still an area of open research¹¹.

The SPKF represents an alternative to the EKF for parameter estimation. However, as the state-transition function is linear, the advantage of the SPKF may not be as obvious. Note the observation function is still nonlinear. Furthermore, the EKF essentially builds up an approximation to the inverse of the expected information matrix (called the Fisher information matrix) by taking outer products of the instantaneous gradient (Jacobian) of the posterior likelihood function (the posterior score function). This approximation is called the *empirical information matrix* [133, 132]. As we show in Section 4.5, the SPKF provides a more accurate estimate (in a statistical sense) of the empirical information matrix by building it up over time using the outer-product of the *expected Jacobian* of the posterior likelihood (posterior score function). Furthermore, the SPKF implicitly includes the variance of the “stochastic linearization error” of the nonlinear observation function into the effective $\mathbf{R}_{\mathbf{e}_k}$ being employed in Equation 3.131. This results in more robust behavior, helping the algorithm avoid getting stuck in certain local minima. These issues are covered in full detail in Section 4.5. While both the EKF and SPKF can be expected to achieve similar final MMSE performance, their convergence properties may differ. In addition, a distinct advantage of the SPKF occurs when either the architecture or error metric is such that differentiation with respect to the parameters is not easily derived, as is necessary in the EKF. The SPKF effectively approximates both the Jacobian and Hessian accurately (in a statistically average sense) through its sigma point propagation, without the need to perform any analytic differentiation.

Specific equations for SPKF parameter estimation are given in Algorithm 6 for the UKF and in Algorithm 7 for the CDKF. Simplifications have been made relative to the state estimation SPKF accounting for the specific form of the state-transition function.

¹¹See Chapter 7.

We have also provided two options on how the function output $\hat{\mathbf{d}}_k$ is calculated. In the first option, the output is given as

$$\hat{\mathbf{d}}_k = \sum_{i=0}^{2L} w_i^{(m)} \mathbf{y}_{i,k|k-1} \approx E[\mathbf{g}(\mathbf{x}_k, \mathbf{w}_k)] , \quad (3.134)$$

corresponding to the direct interpretation of the SPKF equations. The output is the expected value (mean) of a function of the random variable \mathbf{w} . In the second option, we have

$$\hat{\mathbf{d}}_k = \mathbf{g}(\mathbf{x}_k, \hat{\mathbf{w}}_k^-) , \quad (3.135)$$

corresponding to the typical interpretation, in which the output is the function with the current “best” set of parameters. This option yields convergence performance that is indistinguishable from the EKF. The first option, however, has different convergence characteristics, and requires further explanation. In the state-space approach to parameter estimation, absolute convergence is achieved when the parameter covariance $\mathbf{P}_{\mathbf{w}_k}$ goes to zero (this also forces the Kalman gain to zero). At this point, the output for either option is identical. However, prior to this, the finite covariance provides a form of averaging on the output of the function, which in turn prevents the parameters from going to the instantaneous (using the currently observed data) minimum of the error surface. Thus the method may help avoid falling into local minimum. Furthermore, it provides a form of built in regularization for short or noisy data sets that are prone to overfitting (see Section 4.5). Note that the complexity of the SPKF algorithm is still order L^3 (L is the number of parameters), due to the need to compute a matrix square-root at each time step. An order L^2 complexity (same as the EKF) can be achieved by using a recursive square-root formulation as given in Algorithms 14 and 15. Detail about the square-root formulation is given in Section 3.6.2.

Algorithm 6 : The UKF for Parameter Estimation

- *Initialization:* $\hat{\mathbf{w}}_0 = E[\mathbf{w}]$, $\mathbf{P}_{\mathbf{w}_0} = E[(\mathbf{w} - \hat{\mathbf{w}}_0)(\mathbf{w} - \hat{\mathbf{w}}_0)^T]$
- *For* $k = 1, \dots, \infty$:

1. Time-update equations:

$$\hat{\mathbf{w}}_k^- = \hat{\mathbf{w}}_{k-1} \quad (3.136)$$

$$\mathbf{P}_{\mathbf{w}_k^-} = \mathbf{P}_{\mathbf{w}_{k-1}} + \mathbf{R}_{\mathbf{r}_{k-1}} \quad (3.137)$$

2. Calculate sigma-points for measurement-update:

$$\boldsymbol{\chi}_{k|k-1} = \left[\hat{\mathbf{w}}_k^- \quad \hat{\mathbf{w}}_k^- + \gamma \sqrt{\mathbf{P}_{\mathbf{w}_k^-}} \quad \hat{\mathbf{x}}_k^- - \gamma \sqrt{\mathbf{P}_{\mathbf{w}_k^-}} \right] \quad (3.138)$$

3. Measurement-update equations:

$$\boldsymbol{\gamma}_{k|k-1} = \mathbf{g}(\mathbf{x}_k, \boldsymbol{\chi}_{k|k-1}) \quad (3.139)$$

$$\text{option 1: } \hat{\mathbf{d}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\gamma}_{i,k|k-1} \quad (3.140)$$

$$\text{option 2: } \hat{\mathbf{d}}_k^- = \mathbf{g}(\mathbf{x}_k, \hat{\mathbf{w}}_k^-) \quad (3.141)$$

$$\mathbf{P}_{\tilde{\mathbf{d}}_k} = \sum_{i=0}^{2L} w_i^{(c)} \left(\boldsymbol{\gamma}_{i,k|k-1} - \hat{\mathbf{d}}_k^- \right) \left(\boldsymbol{\gamma}_{i,k|k-1} - \hat{\mathbf{d}}_k^- \right)^T + \mathbf{R}_{\mathbf{e}_k} \quad (3.142)$$

$$\mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} = \sum_{i=0}^{2L} w_i^{(c)} \left(\boldsymbol{\chi}_{i,k|k-1} - \hat{\mathbf{w}}_k^- \right) \left(\boldsymbol{\gamma}_{i,k|k-1} - \hat{\mathbf{d}}_k^- \right)^T \quad (3.143)$$

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} \mathbf{P}_{\tilde{\mathbf{d}}_k}^{-1} \quad (3.144)$$

$$\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_k^- + \mathbf{K}_k \left(\mathbf{d}_k - \hat{\mathbf{d}}_k^- \right) \quad (3.145)$$

$$\mathbf{P}_{\mathbf{w}_k} = \mathbf{P}_{\mathbf{w}_k^-} - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{d}}_k} \mathbf{K}_k^T \quad (3.146)$$

- *Parameters:* \mathbf{R}_r is the artificial process-noise covariance and \mathbf{R}_n is the observation-noise covariance (See Section 3.5.2 for detail on how these should be calculated/adapted. $\gamma = \sqrt{L + \lambda}$, γ is the composite scaling parameter and λ is given by Eq. 3.11, L is the dimension of the state, and w_i are the weights as calculated in Eq. 3.12.
-

Algorithm 7 : The CDKF for Parameter Estimation

- *Initialization:* $\hat{\mathbf{w}}_0 = E[\mathbf{w}]$, $\mathbf{P}_{\mathbf{w}_0} = E[(\mathbf{w} - \hat{\mathbf{w}}_0)(\mathbf{w} - \hat{\mathbf{w}}_0)^T]$

- *For* $k = 1, \dots, \infty$:

1. Time-update equations:

$$\hat{\mathbf{w}}_k^- = \hat{\mathbf{w}}_{k-1} \quad (3.147)$$

$$\mathbf{P}_{\mathbf{w}_k^-} = \mathbf{P}_{\mathbf{w}_{k-1}} + \mathbf{R}_{\mathbf{r}_{k-1}} \quad (3.148)$$

2. Calculate sigma-points for measurement-update:

$$\boldsymbol{\chi}_{k|k-1} = \left[\hat{\mathbf{w}}_k^- \quad \hat{\mathbf{w}}_k^- + h\sqrt{\mathbf{P}_{\mathbf{w}_k^-}} \quad \hat{\mathbf{x}}_k^- - h\sqrt{\mathbf{P}_{\mathbf{w}_k^-}} \right] \quad (3.149)$$

3. Measurement-update equations:

$$\boldsymbol{\gamma}_{k|k-1} = \mathbf{g}(\mathbf{x}_k, \boldsymbol{\chi}_{k|k-1}) \quad (3.150)$$

$$\text{option 1: } \hat{\mathbf{d}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\gamma}_{i,k|k-1} \quad (3.151)$$

$$\text{option 2: } \hat{\mathbf{d}}_k^- = \mathbf{g}(\mathbf{x}_k, \hat{\mathbf{w}}_k^-) \quad (3.152)$$

$$\begin{aligned} \mathbf{P}_{\tilde{\mathbf{d}}_k} &= \sum_{i=1}^L \left[w_i^{(c_1)} (\boldsymbol{\gamma}_{i,k|k-1} - \boldsymbol{\gamma}_{L+i,k|k-1})^2 + \right. \\ &\quad \left. w_i^{(c_2)} (\boldsymbol{\gamma}_{i,k|k-1} + \boldsymbol{\gamma}_{L+i,k|k-1} - 2\boldsymbol{\gamma}_{0,k|k-1})^2 \right] + \mathbf{R}_{\mathbf{e}_k} \end{aligned} \quad (3.153)$$

$$\mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} = \sqrt{w_1^{(c_1)} \mathbf{P}_{\mathbf{w}_k^-}} [\boldsymbol{\gamma}_{1:L,k|k-1} - \boldsymbol{\gamma}_{L+1:2L,k|k-1}]^T \quad (3.154)$$

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} \mathbf{P}_{\tilde{\mathbf{d}}_k}^{-1} \quad (3.155)$$

$$\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_k^- + \mathbf{K}_k (\mathbf{d}_k - \hat{\mathbf{d}}_k^-) \quad (3.156)$$

$$\mathbf{P}_{\mathbf{w}_k} = \mathbf{P}_{\mathbf{w}_k^-} - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{d}}_k} \mathbf{K}_k^T \quad (3.157)$$

- *Parameters:* \mathbf{R}_r is the artificial process-noise covariance and \mathbf{R}_n is the observation-noise covariance (See Section 3.5.2 for detail on how these should be calculated/adapted. $h \geq 1$ is the scalar central difference step size, L is the dimension of the state, , and w_i are the weights as calculated in Equation 3.85. $(\cdot)^2$ is shorthand for the vector outer product.
-

Parameter Estimation Examples

We have performed a number of experiments to illustrate the performance of the SPKF parameter estimation approach. The first set of experiments corresponds to benchmark problems for neural network training, and serve to illustrate some of the differences between the EKF and the SPKF, as well as the different options discussed above. Two parametric optimization problems are also included, corresponding to model estimation of the inverted double pendulum, and the benchmark “*Rossmann’s Banana*” optimization problem.

Benchmark neural network regression and time-series problems: The *Mackay-Robot-Arm* dataset [128, 127] and the *Ikeda* chaotic time series [83] are used as benchmark problems to compare neural network training. For the Mackay-Robot-Arm problem we train a 2-12-2 MLP neural network to map the joint angles of a robot arm to the Cartesian coordinates of the robot’s hand. Figure 3.9 illustrates the differences in learning curves for the EKF versus SPKF using option 1 (see Equation 3.134). Note the slightly lower final MSE performance of the SPKF weight training. If option 2 for the SPKF output is used (see Equation 3.135), then the learning curves for the EKF and SPKF are indistinguishable; this has been found to be consistent with all experiments, thus we will not show explicit learning curves for the SPKF with option 2.

Figure 3.10 illustrates performance differences based on the choice of process noise covariance $\mathbf{R}_{\mathbf{r}_k}$. The Mackey-Glass and Ikeda time-series are used. The plots show only comparisons for the SPKF (differences are similar for the EKF). In general the Robbins-Monro method is the most robust approach with the fastest rate of convergence. In some examples, we have seen faster convergence with the “annealed” approach, however, this also requires additional insight and heuristic methods to monitor the learning. We should re-iterate that the “fixed” and “RLS-lambda” approaches are more appropriate for on-line tracking problems.

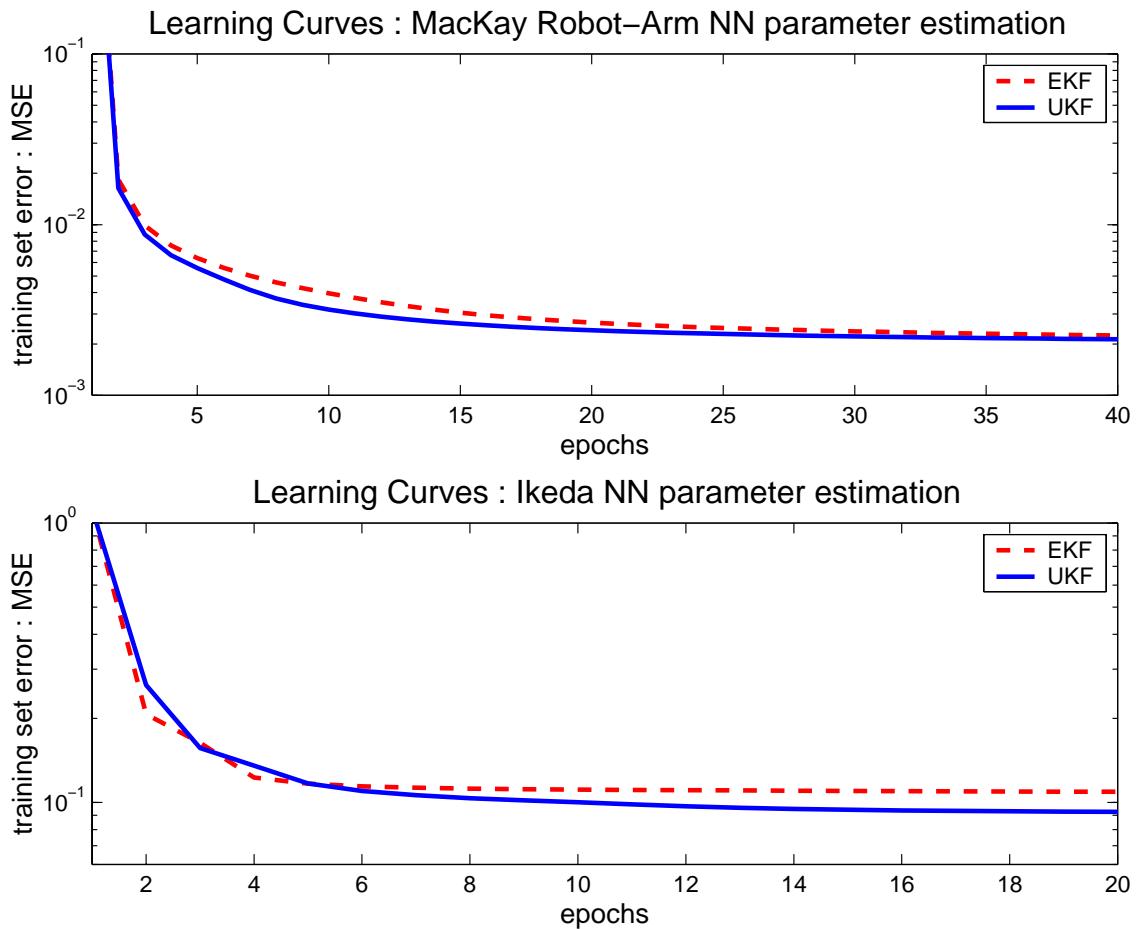


Figure 3.9: (top) MacKay-Robot-Arm problem : comparison of learning curves for the EKF and UKF training, 2-12-2 MLP, 'annealing' noise estimation. (bottom) Ikeda chaotic time series : comparison of learning curves for the EKF and SPKF training, 10-7-1 MLP, 'Robbins-Monro' noise estimation.

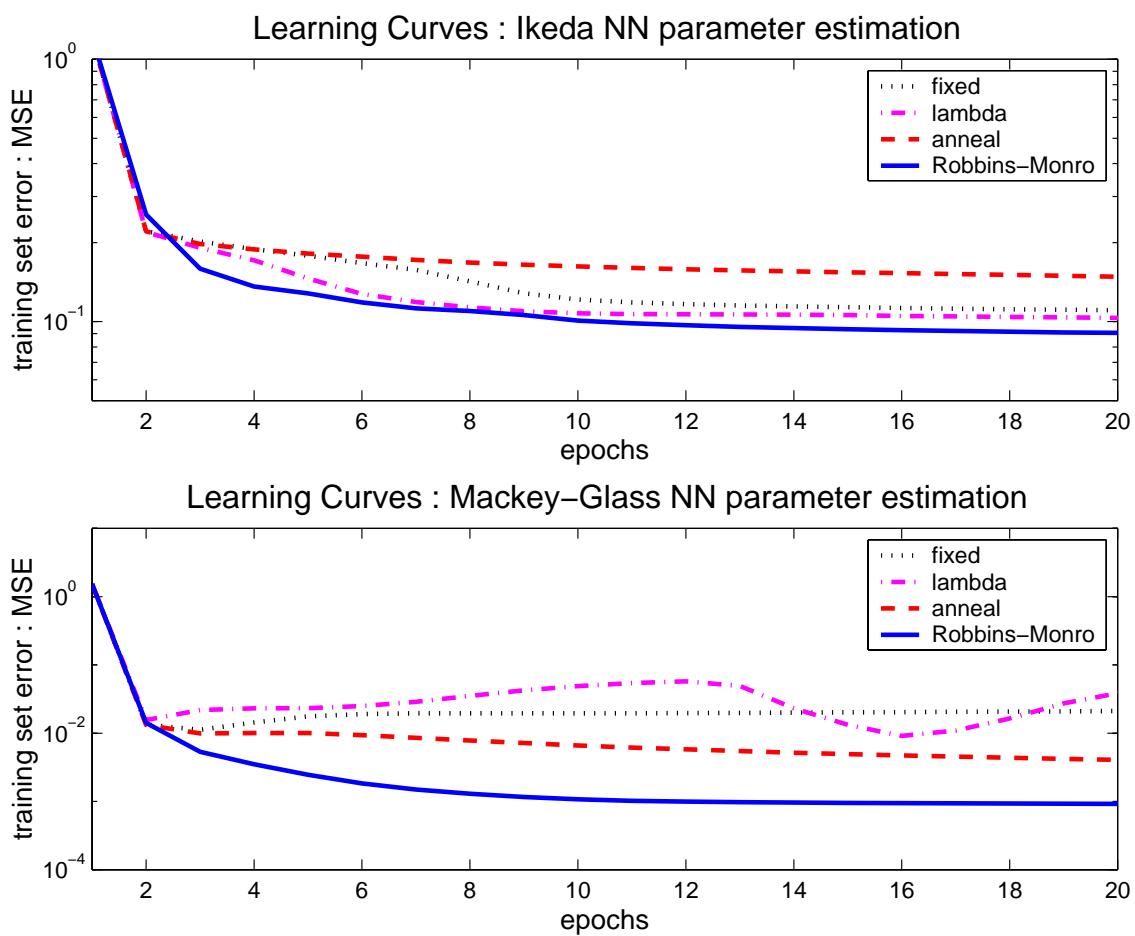


Figure 3.10: Neural network parameter estimation using different methods for noise estimation. (top) Ikeda chaotic time series. (bottom) Mackey-Glass chaotic time series.

Table 3.2: Inverted double pendulum parameter estimation: true vs. estimated parameter values.

	l_1	l_2	m_1	m_2	M
True model	0.50	0.75	0.75	0.50	1.50
SPKF estimate	0.50	0.75	0.75	0.50	1.49
EKF estimate	0.50	0.75	0.68	0.45	1.35

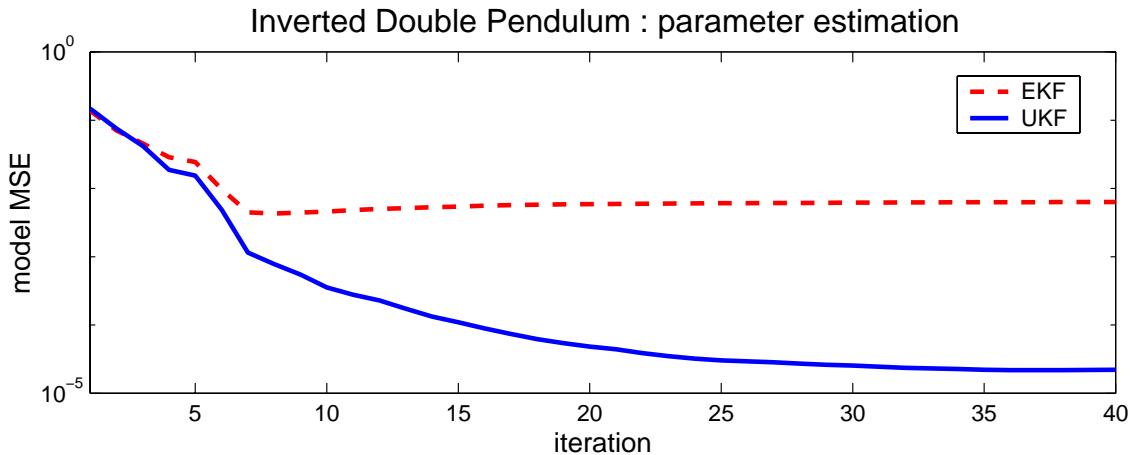


Figure 3.11: Inverted double pendulum parameter estimation: EKF vs. SPKF learning curves.

Inverted double pendulum: Returning to the inverted double pendulum (Section 3.5.1), we consider learning the system parameters, $\mathbf{w} = [l_1 \ l_2 \ m_1 \ m_2 \ M]$. These parameter values are treated as unknown with all initialized to 1.0. The full system state, $\mathbf{x} = [x \ \dot{x} \ \theta_1 \ \dot{\theta}_1 \ \theta_2 \ \dot{\theta}_2]$, is observed. Figure 3.11 shows the total model MSE versus iteration comparing the EKF to SPKF. Each iteration represents a pendulum crash with different initial conditions for the state (no control is applied). The final converged parameter estimates are tabulated in Table 3.2. Clearly, the EKF has converged to a biased solution, possibly corresponding to a local minimum in the error surface.

Four-regions classification : In the next parameter estimation example, we consider a benchmark pattern classification problem having four interlocking regions [178]. A three-layer feed-forward network (MLP) with 2-10-10-4 nodes is trained using inputs randomly drawn within the pattern space, $S = [-1, -1] \times [1, 1]$, with the desired output value of +0.8

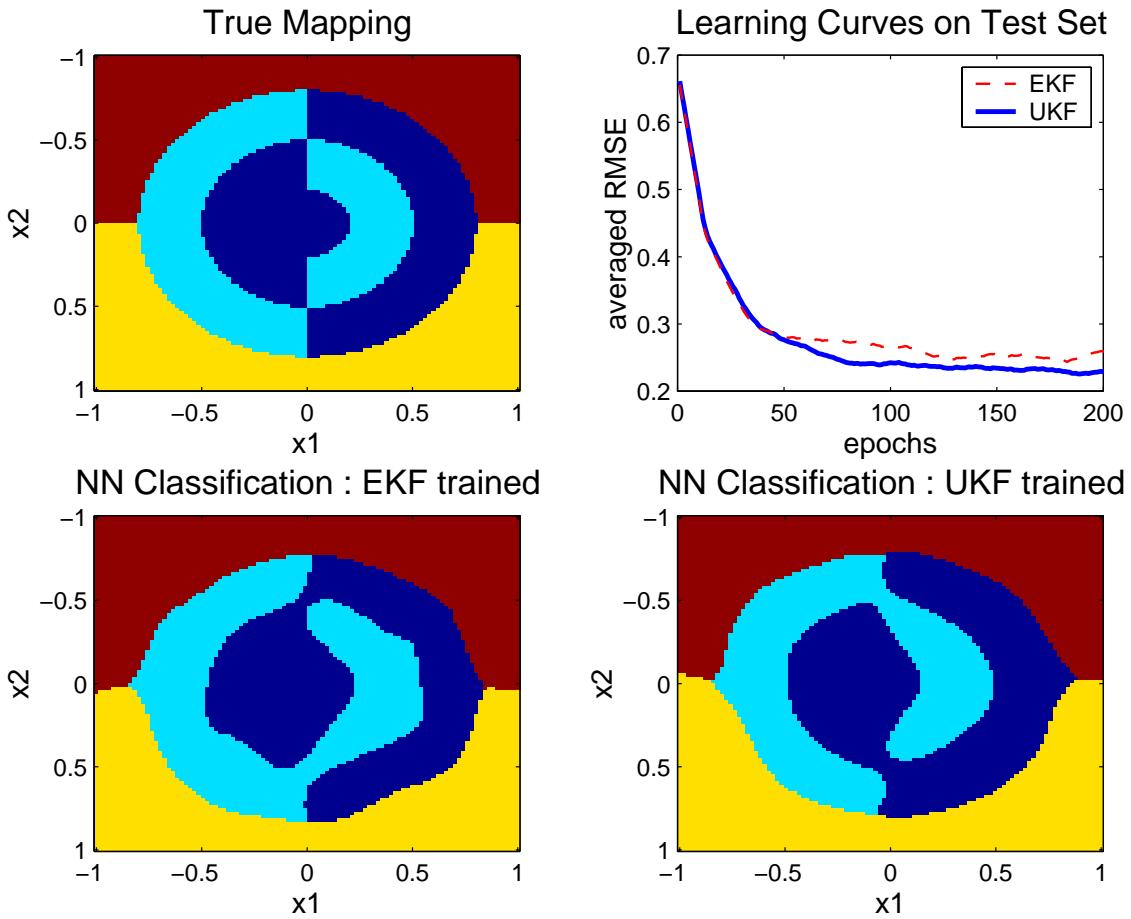


Figure 3.12: Singhal and Wu's *four-region* classification problem: (top-left) True mapping. (top-right) Learning curves on the test set. (bottom-left) Neural network classification: EKF trained. (bottom-right) Neural network classification: UKF trained [2-10-10-4 MLP; Robbins-Monro; 1 epoch=100 random examples].

if the pattern fell within the assigned region and -0.8 otherwise. Figure 3.12 illustrates the classification task, learning curves for the SPKF and EKF, and the final classification regions. For the learning curve, each epoch represents 100 randomly drawn input samples. The test set evaluated on each epoch corresponds to a uniform grid of 10,000 points. Again, we see the superior performance for the SPKF.

Rosenbrock’s Banana function: For the last parameter estimation example, we turn to a pure optimization problem. The “Banana function” [166] can be thought of as a two-dimensional surface which has a saddle-like curvature that bends around the origin. Specifically, we wish to find the values of x_1 and x_2 that minimizes the function

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 . \quad (3.158)$$

The true minimum is at $x_1 = 1$ and $x_2 = 1$. The Banana function is a well known test problem used to compare the convergence rates of competing minimization techniques. In order to use the SPKF or EKF, the basic parameter estimation equations need to be reformulated to minimize a non-MSE cost function. To do this we write the state-space equations in observed-error form [159]:

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \mathbf{r}_k \quad (3.159)$$

$$\mathbf{0} = -\mathbf{\epsilon}_k + \mathbf{e}_k , \quad (3.160)$$

where the target “observation” is fixed at zero, and $\mathbf{\epsilon}$ is an error term resulting in the optimization of the sum of instantaneous costs

$$J_k = \mathbf{\epsilon}_k^T \mathbf{\epsilon}_k . \quad (3.161)$$

The MSE cost is optimized by setting

$$\mathbf{\epsilon}_k = \mathbf{d}_k - \mathbf{g}(\mathbf{x}_k, \mathbf{w}_k) . \quad (3.162)$$

However, arbitrary costs (e.g., cross-entropy) can also be minimized simply by specifying $\mathbf{\epsilon}_k$

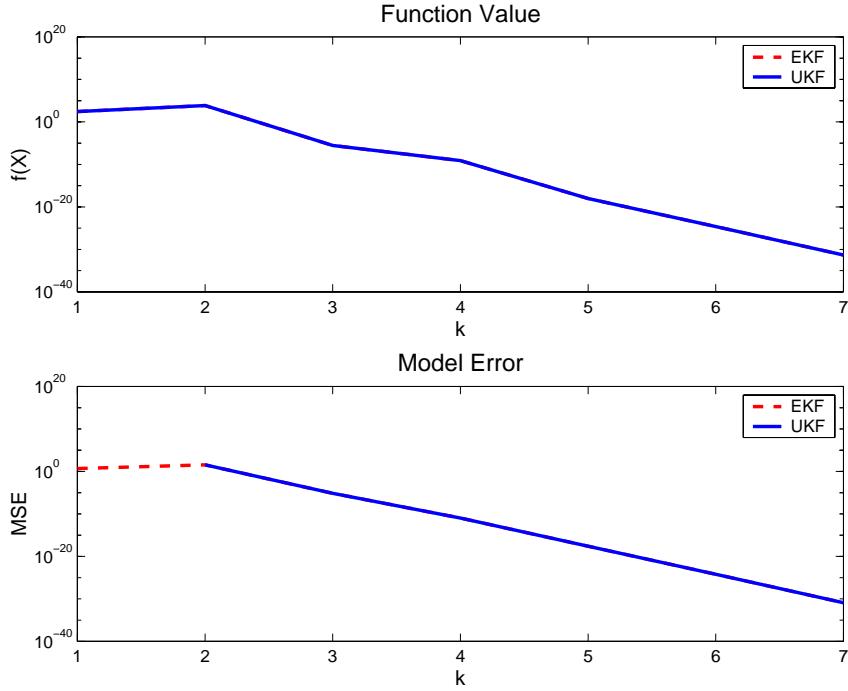


Figure 3.13: Rosenbrock’s “Banana” optimization problem.

appropriately. Further discussion of this approach is given in [143] and [76]. Reformulation of the SPKF equations requires changing only the effective output to be ϵ_k , and setting the desired response to zero. For the example at hand, we set

$$\epsilon_k = \begin{bmatrix} 10(x_2 - x_1^2) & (1 - x_1) \end{bmatrix}^T .$$

Furthermore, since this optimization problem is a special case of ‘noiseless’ parameter estimation where the actual error can be minimized to zero, we make use of Equation 3.162 (option 2) to calculate the output of the UKF algorithm. This will allow the UKF to reach the true minimum of the error surface more rapidly¹². We also set the scaling parameter α to a small value, which we have found to be appropriate again for zero MSE problems. Under these circumstances the performance of the SPKF and EKF is indistinguishable

¹²Note that the use of Option 1, where the expected value of the function is used as the output, essentially involves averaging of the output based on the current parameter covariance. This slows convergence in the case where zero MSE is possible since convergence of the state covariance to zero would also be necessary through proper annealing of the state process noise covariance \mathbf{R}_{r_k} .

as illustrated in Figure 3.13. Overall, the performance of the two filters are at least comparable to a number of alternative 2nd order optimization approaches (e.g., Davidon-Fletcher-Powell, Levenberg-Marquardt, etc. See “*optdemo*” in the *Matlab optimization toolbox* [187, 186]). The main purpose of this example was to illustrate the versatility of the SPKF to general optimization problems.

3.5.3 SPKF Dual Estimation

Recall that the dual estimation problem consists of simultaneously estimating the clean state \mathbf{x}_k and the model parameters \mathbf{w} from the noisy data \mathbf{y}_k (see Equation 2.67). A number of algorithmic approaches exist for this problem, including *joint* and *dual EKF* methods (recursive-prediction-error and maximum-likelihood versions [143]), and *expectation-maximization (EM)* approaches. A thorough coverage of these algorithms is given in [143] by Nelson, and in Chapters 5 and 6 of [76]. In this section, we present results for the dual-SPKF and joint-SPKF methods.

In the the *dual extended Kalman filter* [202], a separate state-space representation is used for the signal and the weights. Two EKFs are run simultaneously for signal and weight estimation. At every time-step, the current estimate of the weights is used in the signal-filter, and the current estimate of the signal-state is used in the weight-filter. In the *dual SPKF* algorithm, both state- and weight-estimation are done with the SPKF. See Figure 3.14 for a schematic representation of the dual filter framework. The two coupled DSSMs used by the dual filtering framework are given by

$$\begin{aligned}\mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{v}_{k-1}; \hat{\mathbf{w}}_{k-1}) \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k; \hat{\mathbf{w}}_{k-1}) ,\end{aligned}\tag{3.163}$$

for the state filter and by

$$\begin{aligned}\mathbf{w}_k &= \mathbf{w}_{k-1} + \mathbf{r}_{k-1} \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{v}_{k-1}; \mathbf{w}_k), \mathbf{n}_k; \mathbf{w}_k) ,\end{aligned}\tag{3.164}$$

for the parameter filter. Notice how the state filter makes use of the estimate of the

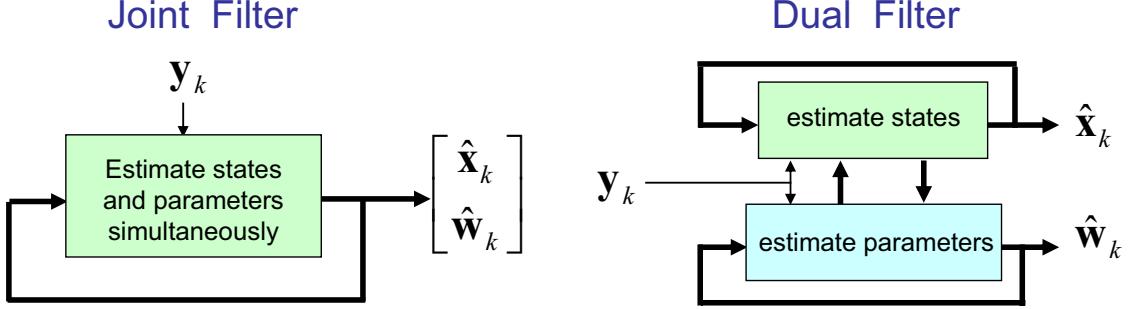


Figure 3.14: Schematic diagrams of two main dual estimation filter implementations: (left) joint filter (right) dual filter

system parameters, $\hat{\mathbf{w}}_{k-1}$, which treated as a constant given input. Likewise, the parameter filter uses the previous state estimate, $\hat{\mathbf{x}}_{k-1}$, as a given (known) input in the observation equation.

In the *joint extended Kalman filter* [131], the signal-state and weight vectors are concatenated into a single, higher-dimensional *joint state vector*, $\tilde{\mathbf{x}}_k = \begin{bmatrix} \mathbf{x}_k^T & \mathbf{w}_k^T \end{bmatrix}^T$. Estimation is done recursively by writing the DSSM for the joint state as

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{f}}(\tilde{\mathbf{x}}_k, \mathbf{u}_k, \tilde{\mathbf{v}}_k) \quad (3.165)$$

$$\mathbf{y}_k = \tilde{\mathbf{h}}(\tilde{\mathbf{x}}_k, \mathbf{n}_k), \quad (3.166)$$

which can be expanded to

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{w}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k; \mathbf{w}_k) \\ \mathbf{w}_k \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_k \end{bmatrix} \quad (3.167)$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k; \mathbf{w}_k), \quad (3.168)$$

where $\tilde{\mathbf{v}}_k = \begin{bmatrix} \mathbf{v}_k^T & \mathbf{r}_k^T \end{bmatrix}^T$, and running an EKF on the joint state-space to produce simultaneous estimates of the states \mathbf{x}_k and the parameters \mathbf{w} . Again, our approach is to use the SPKF instead of the EKF.

A careful analysis of the differences between the dual and joint filtering frameworks show how, in theory at least, the joint approach is expected to provide better estimates [143]. Since the joint filter concatenates the state and parameter random variables into

a single augmented state, it effectively models the cross-covariance between the state and parameter estimates, i.e.,

$$E \left[\left(\hat{\tilde{\mathbf{x}}}_k - E \left[\hat{\tilde{\mathbf{x}}}_k \right] \right) \left(\hat{\tilde{\mathbf{x}}}_k - E \left[\hat{\tilde{\mathbf{x}}}_k \right] \right)^T \right] = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_k} & \mathbf{P}_{\mathbf{x}_k \mathbf{w}_k} \\ \mathbf{P}_{\mathbf{w}_k \mathbf{x}_k} & \mathbf{P}_{\mathbf{w}_k} \end{bmatrix}. \quad (3.169)$$

This full covariance structure allows the joint-SPKF framework to not only accurately treat the dual uncertainty of the parameter and state estimates (through the sigma-point approach), but also to accurately model the interaction (correlation) between the states and parameters. The dual framework on the other hand, decouples (in a statistical sense) the dual estimation problem by treating the parameter estimate as a known fixed value and not as a random variable in the state filter, and the state estimate as a fixed known value (and not as a random variable) in the parameter filter. This effectively equates the off-diagonal blocks of Equation 3.169 to zero, i.e., $\mathbf{P}_{\mathbf{x}_k \mathbf{w}_k} = \mathbf{P}_{\mathbf{w}_k \mathbf{x}_k}^T = \mathbf{0}$. As pointed out in Section 2.6.3, this can have some effect on how the algorithm searches the combined state-parameter state-space in order to find the optimal solution.

Dual Estimation Experiments

In this section we present a number of representative dual estimation experiments, as we did for state- and parameter estimation above.

Noisy time-series: We present results on two time-series to provide a clear illustration of the use of the SPKF over the EKF. The first series is again the Mackey-Glass-30 chaotic series with additive noise ($\text{SNR} \approx 3\text{dB}$). The second time-series (also chaotic) comes from an autoregressive neural network with random weights driven by Gaussian process noise and also corrupted by additive white Gaussian noise ($\text{SNR} \approx 3\text{dB}$). A standard 6-10-1 MLP with *hyperbolic tangent* (\tanh) hidden activation functions and a linear output layer was used for all the filters in the Mackey-Glass problem. A 5-3-1 MLP was used for the second problem. The process and measurement noise variances associated with the state were assumed to be known. Note that in contrast to the state estimation example in the previous section, only the noisy time-series is observed. A clean reference is never provided

for training. Example training curves for the different dual and joint Kalman based estimation methods are shown in Figure 3.15. A final estimate for the Mackey-Glass series is also shown for the Dual SPKF. The superior performance of the SPKF based algorithms are clear.

Inverted double pendulum: In this dual estimation example, we again consider the inverted double pendulum (see Section 3.5.1 for model details). As mentioned earlier, we use a SDRE controller [30] to balance the system. The SDRE controller needs accurate estimates of the system state as well as system parameters in order to accurately and robustly balance the pendulum. In our experiment, only partial and noisy observations of the system states were available and the system parameters were also initialized to incorrect values. A SPKF (a joint-UKF in this case) is used to estimate both the underlying system states and the true system parameters using only the noisy observations at each time step, which is then fed back to the SDRE controller for closed-loop control. Figure 3.16 illustrates the performance of this adaptive control system by showing the evolution of the estimated and actual states (middle) as well as the system parameter estimates (bottom). At the start of the simulation both the states and parameters are unknown (the control system is unstable at this point). However, within one trial, the SPKF enables convergence and stabilization of the pendulum without a single crash! This is in stark contrast to standard system identification and adaptive control approaches, where numerous off-line trials, large amounts of training data and painstaking “hand-tuning” are often needed.

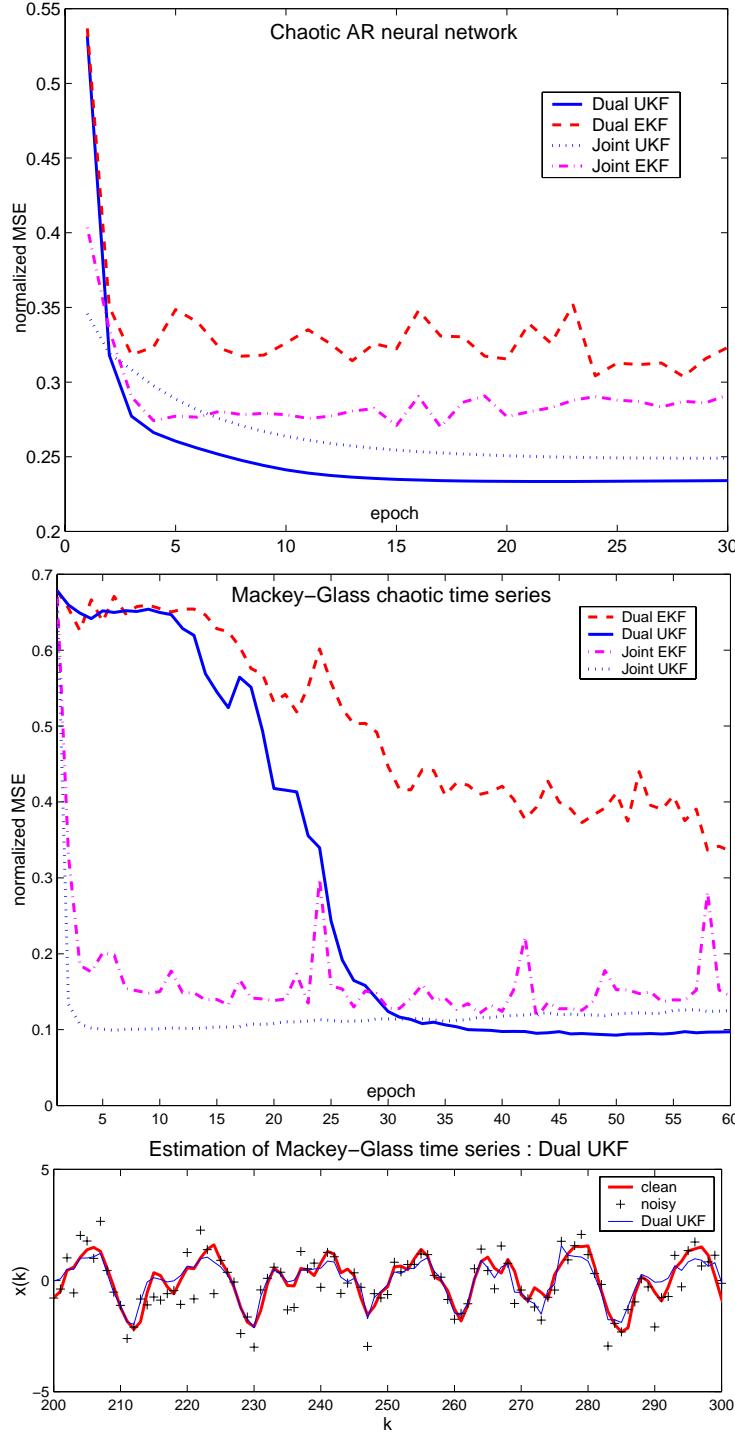


Figure 3.15: Comparative learning curves and results for the dual estimation experiments. Curves are averaged over 10 and 3 runs respectively using different initial weights. “Fixed” innovation covariances are used in the joint algorithms. “Annealed” covariances are used for the weight filter in the dual algorithms. (top) Chaotic AR neural network. (middle) Mackey-Glass chaotic time series. (bottom) Estimation of Mackey-Glass time series: dual SPKF.

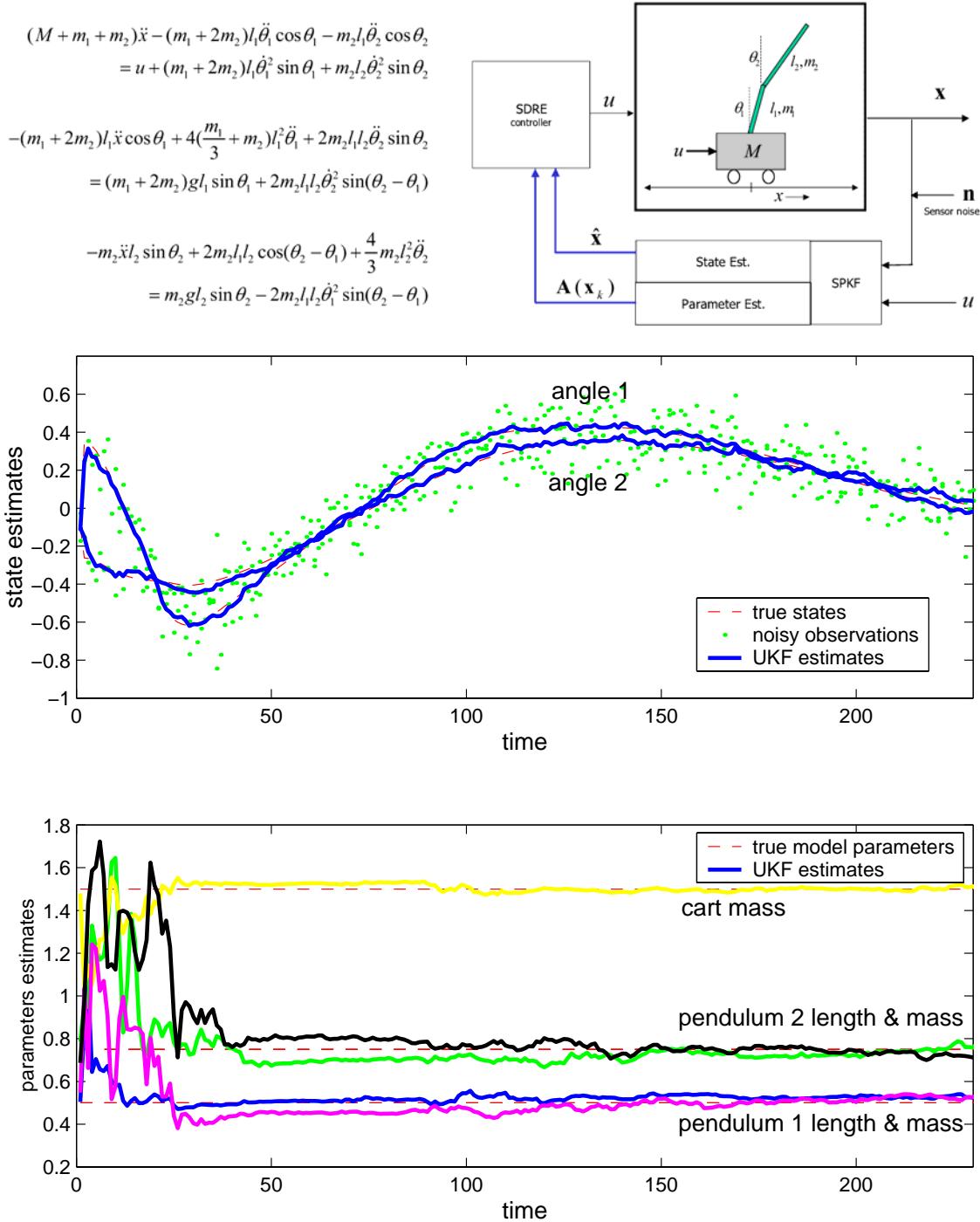


Figure 3.16: Inverted double pendulum dual estimation and control experiment. Dynamic model and schematic of control system (top plot). Estimation results are shown in the bottom plot: state estimates (top) and parameter estimates (bottom).

3.6 SPKF Extensions & Implementation Variations

In this section we introduce extensions and alternative implementational forms of the SPKF, which, depending on the specific form of the estimation problem, results in significant computational cost savings as well as an increase in numerical robustness. Where relevant, experimental verification is given.

3.6.1 Additive noise forms

For the special (but often found) case where the process and measurement noise are purely additive, the computational complexity of the SPKF can be reduced by a linear scaling factor. In such a case, the system state need not be augmented with the noise RV's. This reduces the dimension of the sigma points as well as the total number of sigma points used. The covariances of the noise sources are then incorporated into the state covariance using a simple additive procedure. This implementation is given in Algorithm 8 for the UKF and in Algorithm 9 for the CDKF. The complexity of the algorithm is order L^3 , where L is the dimension of the state. This is the same complexity as the EKF. The most costly operation is in forming the sample prior covariance matrix $\mathbf{P}_{\mathbf{x}_k}^-$. Depending on the form of \mathbf{f} , this may be simplified, e.g., for univariate time-series estimation or with parameter estimation (see Section 3.5.2) the complexity can be reduced further to order L^2 .

Algorithm 8 : The unscented Kalman filter (UKF) - additive noise case

- *Initialization:* $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$, $\mathbf{P}_{\mathbf{x}_0} = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$
 - *For* $k = 1, \dots, \infty$:
1. Calculate sigma-points:

$$\boldsymbol{\chi}_{k-1} = \left[\begin{array}{ccc} \hat{\mathbf{x}}_{k-1} & \hat{\mathbf{x}}_{k-1} + \gamma \sqrt{\mathbf{P}_{\mathbf{x}_{k-1}}} & \hat{\mathbf{x}}_{k-1} - \gamma \sqrt{\mathbf{P}_{\mathbf{x}_{k-1}}} \end{array} \right] \quad (3.170)$$

2. Time-update equations:

$$\boldsymbol{\chi}_{k|k-1}^* = \mathbf{f}(\boldsymbol{\chi}_{k-1}, \mathbf{u}_{k-1}) \quad (3.171)$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\chi}_{i,k|k-1}^* \quad (3.172)$$

$$\mathbf{P}_{\mathbf{x}_k}^- = \sum_{i=0}^{2L} w_i^{(c)} (\boldsymbol{\chi}_{i,k|k-1}^* - \hat{\mathbf{x}}_k^-) (\boldsymbol{\chi}_{i,k|k-1}^* - \hat{\mathbf{x}}_k^-)^T + \mathbf{R}_v \quad (3.173)$$

$$(see \ below) \ \boldsymbol{\chi}_{k|k-1} = \begin{bmatrix} \boldsymbol{\chi}_{k|k-1}^* & \boldsymbol{\chi}_{0,k|k-1}^* + \gamma \sqrt{\mathbf{R}_v} & \boldsymbol{\chi}_{0,k|k-1}^* - \gamma \sqrt{\mathbf{R}_v} \end{bmatrix} \quad (3.174)$$

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{h}(\boldsymbol{\chi}_{k|k-1}^x) \quad (3.175)$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\mathcal{Y}}_{i,k|k-1} \quad (3.176)$$

3. Measurement-update equations:

$$\mathbf{P}_{\tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} w_i^{(c)} (\boldsymbol{\mathcal{Y}}_{i,k|k-1} - \hat{\mathbf{y}}_k^-) (\boldsymbol{\mathcal{Y}}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T + \mathbf{R}_n \quad (3.177)$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sum_{i=0}^{2L} w_i^{(c)} (\boldsymbol{\chi}_{i,k|k-1} - \hat{\mathbf{x}}_k^-) (\boldsymbol{\mathcal{Y}}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T \quad (3.178)$$

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k}^{-1} \quad (3.179)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (3.180)$$

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k} \mathbf{K}_k^T \quad (3.181)$$

- *Parameters:* $\gamma = \sqrt{L+\lambda}$ is the composite scaling parameter and λ is given by Eq. 3.11, L is the dimension of the state, \mathbf{R}_v is the process-noise covariance, \mathbf{R}_n is the observation-noise covariance, and w_i are the weights as calculated in Equation 3.12. *Sigma-point set augmentation:* In (3.174), the sigma-points are *augmented* with additional points derived from the matrix square root of the process noise covariance, in order to incorporate the effect of the process noise on the observed sigma-points, $\boldsymbol{\mathcal{Y}}$. This requires setting $L \rightarrow 2L$ and recalculating the various weights w_i accordingly. Alternatively, we may *redraw* a complete new set of sigma points, i.e., $\boldsymbol{\chi}_{k|k-1} = [\hat{\mathbf{x}}_k^- \ \hat{\mathbf{x}}_k^- + \gamma \sqrt{\mathbf{P}_{\mathbf{x}_k}} \ \hat{\mathbf{x}}_k^- - \gamma \sqrt{\mathbf{P}_{\mathbf{x}_k}}]$. This alternative approach results in fewer sigma points being used, but also discards any odd-moments information captured by the original propagated sigma points.

Algorithm 9 : The central difference Kalman filter (CDKF) - additive noise case

- *Initialization:* $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$, $\mathbf{P}_{\mathbf{x}_0} = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$
- *For* $k = 1, \dots, \infty$:

1. Calculate sigma-points for time-update:

$$\boldsymbol{\chi}_{k-1} = \begin{bmatrix} \hat{\mathbf{x}}_{k-1} & \hat{\mathbf{x}}_{k-1} + h\sqrt{\mathbf{P}_{\mathbf{x}_{k-1}}} & \hat{\mathbf{x}}_{k-1} - h\sqrt{\mathbf{P}_{\mathbf{x}_{k-1}}} \end{bmatrix} \quad (3.182)$$

2. Time-update equations:

$$\boldsymbol{\chi}_{k|k-1} = \mathbf{f}(\boldsymbol{\chi}_{k-1}, \mathbf{u}_{k-1}) \quad (3.183)$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\chi}_{i,k|k-1} \quad (3.184)$$

$$\begin{aligned} \mathbf{P}_{\mathbf{x}_k}^- &= \sum_{i=1}^L \left[w_i^{(c_1)} (\boldsymbol{\chi}_{i,k|k-1} - \boldsymbol{\chi}_{L+i,k|k-1})^2 + \right. \\ &\quad \left. w_i^{(c_2)} (\boldsymbol{\chi}_{i,k|k-1} + \boldsymbol{\chi}_{L+i,k|k-1} - 2\boldsymbol{\chi}_{0,k|k-1})^2 \right] + \mathbf{R}_v \end{aligned} \quad (3.185)$$

3. Calculate sigma-points for measurement-update:

$$\boldsymbol{\chi}_{k|k-1}^* = \begin{bmatrix} \hat{\mathbf{x}}_k^- & \hat{\mathbf{x}}_k^- + h\sqrt{\mathbf{P}_{\mathbf{x}_k}^-} & \hat{\mathbf{x}}_k^- - h\sqrt{\mathbf{P}_{\mathbf{x}_k}^-} \end{bmatrix} \quad (3.186)$$

4. Measurement-update equations:

$$\boldsymbol{\gamma}_{k|k-1} = \mathbf{h}(\boldsymbol{\chi}_{k|k-1}^*) \quad (3.187)$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\gamma}_{i,k|k-1} \quad (3.188)$$

$$\begin{aligned} \mathbf{P}_{\tilde{\mathbf{y}}_k}^- &= \sum_{i=1}^L \left[w_i^{(c_1)} (\boldsymbol{\gamma}_{i,k|k-1} - \boldsymbol{\gamma}_{L+i,k|k-1})^2 + \right. \\ &\quad \left. w_i^{(c_2)} (\boldsymbol{\gamma}_{i,k|k-1} + \boldsymbol{\gamma}_{L+i,k|k-1} - 2\boldsymbol{\gamma}_{0,k|k-1})^2 \right] + \mathbf{R}_n \end{aligned} \quad (3.189)$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sqrt{w_1^{(c_1)} \mathbf{P}_{\mathbf{x}_k}^-} [\boldsymbol{\gamma}_{1:L,k|k-1} - \boldsymbol{\gamma}_{L+1:2L,k|k-1}]^T \quad (3.190)$$

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k}^{-1} \quad (3.191)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (3.192)$$

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k} \mathbf{K}_k^T \quad (3.193)$$

- *Parameters:* $h \geq 1$ is the scalar central difference step size, L is the dimension of the state, \mathbf{R}_v is the process-noise covariance, \mathbf{R}_n is the observation-noise covariance, and w_i are the

weights as calculated in Equation 3.85. $(\cdot)^2$ is shorthand for the vector outer product.

3.6.2 Square-root forms

One of the most costly operations in the SPKF is the calculation of the matrix square-root of the state covariance at each time step in order to form the sigma-point set (See Equations 3.19 and 3.104). Due to this and the need for more numerical stability (especially during the state covariance update), we derived numerically efficient *square-root* forms of both the UKF and the CDKF [191, 192]. These forms propagate and update the square-root of the state covariance directly in Cholesky factored form, using the sigma-point approach and the following linear algebra techniques: *QR decomposition*, *Cholesky factor updating* and *efficient pivot-based least squares*. The square-root SPKFs (SR-UKF and SR-CDKF) has equal (or marginally better) estimation accuracy when compared to the standard SPKF, but at the added benefit of reduced computational cost for certain DSSMs and a consistently increased numerical stability (all resulting covariance matrices are guaranteed to stay positive definite). For this reason, they are our preferred form for SPKF use in stand-alone estimators as well as in SMC/SPKF hybrids (see Chapter 6).

In the standard Kalman implementation, the state (or parameter) covariance $\mathbf{P}_{\mathbf{x}_k}$ is recursively calculated. The SPKF requires taking the matrix square-root $\mathbf{S}_{\mathbf{x}_k} \mathbf{S}_{\mathbf{x}_k}^T = \mathbf{P}_{\mathbf{x}_k}$, at each time step, which is $\mathcal{O}(L^3/6)$ using a Cholesky factorization. In the square-root SPKF (SR-SPKF), \mathbf{S}_k will be propagated directly, avoiding the need to refactorize at each time step. The algorithm will in general still be $\mathcal{O}(L^3)$ for state-estimation, but with improved numerical properties (e.g, guaranteed positive semi-definiteness of the state-covariances) similar to those of standard square-root Kalman filters [170]. However, for the special state-space formulation of parameter estimation, an $\mathcal{O}(L^2)$ implementation becomes possible (see Algorithms 14 and 15) with equivalent complexity to EKF parameter estimation.

As stated above, the SR-SPKF makes use of three powerful linear algebra techniques for theoretical and implementation details which we briefly review below:

- ***QR decomposition***: The QR decomposition or factorization of a matrix $\mathbf{A} \in \mathbb{R}^{LxN}$

is given by,

$$\mathbf{A}^T = \mathbf{Q}\mathbf{R},$$

where $\mathbf{Q} \in \mathbb{R}^{N \times N}$ is orthogonal, $\mathbf{R} \in \mathbb{R}^{N \times L}$ is upper triangular and $N \geq L$. The upper triangular part of \mathbf{R} , $\tilde{\mathbf{R}}$, is the transpose of the Cholesky factor of $\mathbf{P} = \mathbf{A}\mathbf{A}^T$, i.e., $\tilde{\mathbf{R}} = \mathbf{S}^T$, such that $\tilde{\mathbf{R}}^T \tilde{\mathbf{R}} = \mathbf{A}\mathbf{A}^T$. We use the shorthand notation $\text{qr}\{\cdot\}$ to denote a QR decomposition of a matrix where only $\tilde{\mathbf{R}}$ is returned. The computational complexity of a QR decomposition is $\mathcal{O}(NL^2)$. Note that performing a Cholesky factorization directly on $\mathbf{P} = \mathbf{A}\mathbf{A}^T$ is $\mathcal{O}(L^3/6)$ plus $\mathcal{O}(NL^2)$ to form $\mathbf{A}\mathbf{A}^T$.

- **Cholesky factor updating:** If \mathbf{S} is the original lower triangular Cholesky factor of $\mathbf{P} = \mathbf{A}\mathbf{A}^T$, then the Cholesky factor of the rank-1 update (or downdate)

$$\check{\mathbf{P}} = \mathbf{P} \pm \sqrt{\nu} \mathbf{u} \mathbf{u}^T$$

is denoted as

$$\check{\mathbf{S}} = \text{cholupdate}\{\mathbf{S}, \mathbf{u}, \pm\nu\}.$$

If \mathbf{u} is a matrix and not a vector, then the result is M consecutive updates of the Cholesky factor using the M columns of \mathbf{u} . This algorithm (available in *Matlab* as `cholupdate`) is only $\mathcal{O}(L^2)$ per update.

- **Efficient least squares:** The solution to the equation

$$(\mathbf{A}\mathbf{A}^T) \mathbf{x} = \mathbf{A}^T \mathbf{b}$$

also corresponds to the solution of the overdetermined least squares problem $\mathbf{Ax} = \mathbf{b}$. This can be solved efficiently using triangular QR decomposition with pivoting [156] (implemented in *Matlab*'s “/” operator).

The complete specifications for the new square-root filters are given in Algorithms 10 through 13 for state estimation and Algorithms 14 and 15 for parameter estimation. Below we describe the key parts of the square-root algorithms, and how they contrast with the standard implementations.

Square-root State Estimation

As in the standard SPKF, the filter is initialized by calculating the matrix square-root of the state covariance once via a Cholesky factorization. However, the propagated and updated Cholesky factor is then used in subsequent iterations to directly form the sigma-points. In Equation 3.198 the *time-update* of the Cholesky factor, $\mathbf{S}_{\mathbf{x}}^-$, is calculated using a QR decomposition of the compound matrix containing the weighted propagated sigma-points and the matrix square-root of the additive process noise covariance. The subsequent Cholesky update (or downdate) in Equation 3.199 is necessary (for the SR-UKF) since the zeroth weight, $W_0^{(c)}$, may be negative. The SR-CDKF has no such requirement. These two steps replace the *time-update* of $\mathbf{P}_{\mathbf{x}}^-$ in Equations 3.22 and 3.107, and is also $\mathcal{O}(L^3)$.

The same two-step approach¹³ is applied to the calculation of the Cholesky factor, $\mathbf{S}_{\tilde{\mathbf{y}}}$, of the observation-error covariance in Equations 3.202 and 3.203. This step is $\mathcal{O}(LM^2)$, where M is the observation dimension. In contrast to the way the Kalman gain is calculated in the standard SPKF (see Equations 3.27 and 3.114), we now use two nested inverse (or *least squares*) solutions to the following expansion of Equation 3.27:

$$\mathbf{K}_k (\mathbf{S}_{\tilde{\mathbf{y}}_k} \mathbf{S}_{\tilde{\mathbf{y}}_k}^T) = \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k}. \quad (3.194)$$

Since $\mathbf{S}_{\tilde{\mathbf{y}}_k}$ is square and triangular, efficient “back-substitutions” can be used to solve for \mathbf{K}_k directly without the need for a matrix inversion [156].

Finally, the posterior measurement update of the Cholesky factor of the state covariance is calculated in Equation 3.208 and 3.238 by applying M sequential Cholesky downdates to $\mathbf{S}_{\mathbf{x}_k}^-$. The downdate vectors are the columns of $\mathbf{U} = \mathbf{K}_k \mathbf{S}_{\tilde{\mathbf{y}}_k}$. This replaces the posterior update of $\mathbf{P}_{\mathbf{x}_k}^-$ in Equations 3.29 and 3.193, and is also $\mathcal{O}(LM^2)$.

¹³For the CDKF, only the first step is needed since all weights are positive (See Equations 3.228 and 3.233).

Algorithm 10 : The square-root UKF (SR-UKF) - general state estimation form

- Initialization: $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$, $\mathbf{S}_{\mathbf{x}_0} = \text{chol}\{E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]\}$, $\mathbf{S}_{\mathbf{v}} = \sqrt{\mathbf{R}_{\mathbf{v}}}$, $\mathbf{S}_{\mathbf{n}} = \sqrt{\mathbf{R}_{\mathbf{n}}}$

$$\hat{\mathbf{x}}_0^a = E[\mathbf{x}^a] = \begin{bmatrix} \hat{\mathbf{x}}_0 & \mathbf{0} & \mathbf{0} \end{bmatrix}^T, \quad \mathbf{S}_0^a = \text{chol}\{E[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T]\} = \begin{bmatrix} \mathbf{S}_{\mathbf{x}_0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{\mathbf{v}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{\mathbf{n}} \end{bmatrix}$$

- For $k = 1, \dots, \infty$:

1. Calculate sigma-points:

$$\boldsymbol{\chi}_{k-1}^a = \left[\hat{\mathbf{x}}_{k-1}^a \quad \hat{\mathbf{x}}_{k-1}^a + \gamma \mathbf{S}_{\mathbf{x}_{k-1}}^a \quad \hat{\mathbf{x}}_{k-1}^a - \gamma \mathbf{S}_{\mathbf{x}_{k-1}}^a \right] \quad (3.195)$$

2. Time-update equations:

$$\boldsymbol{\chi}_{k|k-1}^x = \mathbf{f}(\boldsymbol{\chi}_{k-1}^a, \boldsymbol{\chi}_{k-1}^v, \mathbf{u}_{k-1}) \quad (3.196)$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\chi}_{i,k|k-1}^x \quad (3.197)$$

$$\mathbf{S}_{\mathbf{x}_k}^- = \text{qr} \left\{ \left[\sqrt{w_1^{(c)}} (\boldsymbol{\chi}_{1:2L,k|k-1}^x - \hat{\mathbf{x}}_k^-) \right] \right\} \quad (3.198)$$

$$\mathbf{S}_{\mathbf{x}_k}^- = \text{cholupdate} \left\{ \mathbf{S}_{\mathbf{x}_k}^-, \boldsymbol{\chi}_{0,k|k-1}^x - \hat{\mathbf{x}}_k^-, w_0^{(c)} \right\} \quad (3.199)$$

$$\boldsymbol{\chi}_{k|k-1}^n = \mathbf{h}(\boldsymbol{\chi}_{i,k|k-1}^x, \boldsymbol{\chi}_{k-1}^n) \quad (3.200)$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\chi}_{i,k|k-1}^n \quad (3.201)$$

3. Measurement-update equations:

$$\mathbf{S}_{\tilde{\mathbf{y}}_k}^- = \text{qr} \left\{ \left[\sqrt{w_1^{(c)}} (\boldsymbol{\chi}_{1:2L,k|k-1}^n - \hat{\mathbf{y}}_k^-) \right] \right\} \quad (3.202)$$

$$\mathbf{S}_{\tilde{\mathbf{y}}_k}^- = \text{cholupdate} \left\{ \mathbf{S}_{\tilde{\mathbf{y}}_k}^-, \boldsymbol{\chi}_{0,k|k-1}^n - \hat{\mathbf{y}}_k^-, w_0^{(c)} \right\} \quad (3.203)$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sum_{i=0}^{2L} w_i^{(c)} (\boldsymbol{\chi}_{i,k|k-1}^n - \hat{\mathbf{x}}_k^-) (\boldsymbol{\chi}_{i,k|k-1}^n - \hat{\mathbf{y}}_k^-)^T \quad (3.204)$$

$$\mathbf{K}_k = (\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} / \mathbf{S}_{\tilde{\mathbf{y}}_k}^T) / \mathbf{S}_{\tilde{\mathbf{y}}_k} \quad (3.205)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (3.206)$$

$$\mathbf{U} = \mathbf{K}_k \mathbf{S}_{\tilde{\mathbf{y}}_k} \quad (3.207)$$

$$\mathbf{S}_{\mathbf{x}_k} = \text{cholupdate} \left\{ \mathbf{S}_{\mathbf{x}_k}^-, \mathbf{U}, -1 \right\} \quad (3.208)$$

- Parameters: $\mathbf{x}^a = \begin{bmatrix} \mathbf{x}^T & \mathbf{v}^T & \mathbf{n}^T \end{bmatrix}^T$, $\mathcal{X}^a = \begin{bmatrix} (\mathcal{X}^x)^T & (\mathcal{X}^v)^T & (\mathcal{X}^n)^T \end{bmatrix}^T$, $\gamma = \sqrt{L + \lambda}$ is the composite scaling parameter and λ is given by Eq. 3.11, L is the dimension of the state, \mathbf{R}_v is the process-noise covariance, \mathbf{R}_n is the observation-noise covariance, and w_i are the weights as calculated in Equation 3.12.
-

Algorithm 11 : The square-root UKF (SR-UKF) - state estimation (additive noise)

- Initialization: $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$, $\mathbf{S}_{\mathbf{x}_0} = \text{chol}\{E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]\}$, $\mathbf{S}_v = \sqrt{\mathbf{R}_v}$, $\mathbf{S}_n = \sqrt{\mathbf{R}_n}$
- For $k = 1, \dots, \infty$:

1. Calculate sigma-points:

$$\mathcal{X}_{k-1} = \begin{bmatrix} \hat{\mathbf{x}}_{k-1} & \hat{\mathbf{x}}_{k-1} + \gamma \mathbf{S}_{\mathbf{x}_{k-1}} & \hat{\mathbf{x}}_{k-1} - \gamma \mathbf{S}_{\mathbf{x}_{k-1}} \end{bmatrix} \quad (3.209)$$

2. Time-update equations:

$$\mathcal{X}_{k|k-1}^* = \mathbf{f}(\mathcal{X}_{k-1}, \mathbf{u}_{k-1}) \quad (3.210)$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \mathcal{X}_{i,k|k-1}^* \quad (3.211)$$

$$\mathbf{S}_{\mathbf{x}_k}^- = \text{qr} \left\{ \begin{bmatrix} \sqrt{w_1^{(c)}} (\mathcal{X}_{1:2L,k|k-1}^* - \hat{\mathbf{x}}_k^-) & \mathbf{S}_v \end{bmatrix} \right\} \quad (3.212)$$

$$\mathbf{S}_{\mathbf{x}_k}^- = \text{cholupdate} \left\{ \mathbf{S}_{\mathbf{x}_k}^-, \mathcal{X}_{0,k|k-1}^* - \hat{\mathbf{x}}_k^-, w_0^{(c)} \right\} \quad (3.213)$$

$$(\text{augment}) \quad \mathcal{X}_{k|k-1} = \begin{bmatrix} \mathcal{X}_{k|k-1}^* & \mathcal{X}_{0,k|k-1}^* + \gamma \mathbf{S}_v & \mathcal{X}_{0,k|k-1}^* - \gamma \mathbf{S}_v \end{bmatrix} \quad (3.214)$$

$$\mathbf{y}_{k|k-1} = \mathbf{h}(\mathcal{X}_{i,k|k-1}^*) \quad (3.215)$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \mathbf{y}_{i,k|k-1} \quad (3.216)$$

3. Measurement-update equations:

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = \text{qr} \left\{ \left[\sqrt{w_1^{(c)}} (\mathcal{Y}_{1:2L,k|k-1} - \hat{\mathbf{y}}_k^-) \quad \mathbf{S}_n \right] \right\} \quad (3.217)$$

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = \text{cholupdate} \left\{ \mathbf{S}_{\tilde{\mathbf{y}}_k}, \mathcal{Y}_{0,k|k-1} - \hat{\mathbf{y}}_k^-, w_0^{(c)} \right\} \quad (3.218)$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sum_{i=0}^{2L} w_i^{(c)} (\mathbf{x}_{i,k|k-1} - \hat{\mathbf{x}}_k^-) (\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T \quad (3.219)$$

$$\mathbf{K}_k = (\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} / \mathbf{S}_{\tilde{\mathbf{y}}_k}^T) / \mathbf{S}_{\tilde{\mathbf{y}}_k} \quad (3.220)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (3.221)$$

$$\mathbf{U} = \mathbf{K}_k \mathbf{S}_{\tilde{\mathbf{y}}_k} \quad (3.222)$$

$$\mathbf{S}_{\mathbf{x}_k} = \text{cholupdate} \left\{ \mathbf{S}_{\mathbf{x}_k}^-, \mathbf{U}, -1 \right\} \quad (3.223)$$

- *Parameters:* $\gamma = \sqrt{L + \lambda}$ is the composite scaling parameter and λ is given by Eq. 3.11, L is the dimension of the state, \mathbf{R}_v is the process-noise covariance, \mathbf{R}_n is the observation-noise covariance, and w_i are the weights as calculated in Equation 3.12. *Sigma-point set augmentation:* In (3.214) the sigma-point set is augmented in a similar fashion as in Algorithm 8. This alternative approach results in fewer sigma-points being used, but also discards any odd-moments information captured by the original propagated sigma points.
-

Algorithm 12 : The square-root CDKF (SR-CDKF) - general state estimation form

- *Initialization:* $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$, $\mathbf{S}_{\mathbf{x}_0} = \text{chol} \left\{ E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T] \right\}$, $\mathbf{S}_v = \sqrt{\mathbf{R}_v}$, $\mathbf{S}_n = \sqrt{\mathbf{R}_n}$
- *For* $k = 1, \dots, \infty$:

1. Calculate sigma points for time-update:

$$\hat{\mathbf{x}}_{k-1}^{av} = [\hat{\mathbf{x}}_{k-1} \quad \bar{\mathbf{v}}] \quad , \quad \mathbf{S}_{k-1}^{av} = \begin{bmatrix} \mathbf{S}_{\mathbf{x}_{k-1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_v \end{bmatrix} \quad (3.224)$$

$$\mathcal{X}_{k-1}^{av} = \begin{bmatrix} \hat{\mathbf{x}}_{k-1}^{av} & \hat{\mathbf{x}}_{k-1}^{av} + h\mathbf{S}_{k-1}^{av} & \hat{\mathbf{x}}_{k-1}^{av} - h\mathbf{S}_{k-1}^{av} \end{bmatrix} \quad (3.225)$$

2. Time-update equations:

$$\boldsymbol{\chi}_{k|k-1}^x = \mathbf{f}(\boldsymbol{\chi}_{k-1}^x, \boldsymbol{\chi}_{k-1}^v, \mathbf{u}_{k-1}) \quad (3.226)$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\chi}_{i,k|k-1}^x \quad (3.227)$$

$$\begin{aligned} \mathbf{S}_{\mathbf{x}_k}^- &= \text{qr} \left\{ \left[\sqrt{w_1^{(c_1)}} (\boldsymbol{\chi}_{1:L,k|k-1}^x - \boldsymbol{\chi}_{L+1:2L,k|k-1}^x) \right. \right. \\ &\quad \left. \left. \sqrt{w_1^{(c_2)}} (\boldsymbol{\chi}_{1:L,k|k-1}^x + \boldsymbol{\chi}_{L+1:2L,k|k-1}^x - 2\boldsymbol{\chi}_{0,k|k-1}^x) \right] \right\} \end{aligned} \quad (3.228)$$

3. Calculate sigma-points for measurement update:

$$\hat{\mathbf{x}}_{k|k-1}^{a_n} = [\hat{\mathbf{x}}_k^- \ \bar{\mathbf{n}}], \quad \mathbf{S}_{k|k-1}^{a_n} = \begin{bmatrix} \mathbf{S}_{\mathbf{x}_k}^- & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{\mathbf{n}} \end{bmatrix} \quad (3.229)$$

$$\boldsymbol{\chi}_{k|k-1}^{a_n} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k-1}^{a_n} & \hat{\mathbf{x}}_{k|k-1}^{a_n} + h\mathbf{S}_{k|k-1}^{a_n} & \hat{\mathbf{x}}_{k|k-1}^{a_n} - h\mathbf{S}_{k|k-1}^{a_n} \end{bmatrix} \quad (3.230)$$

4. Measurement-update equations:

$$\mathbf{y}_{k|k-1} = \mathbf{h}(\boldsymbol{\chi}_{k|k-1}^x, \boldsymbol{\chi}_{k|k-1}^n) \quad (3.231)$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \mathbf{y}_{i,k|k-1} \quad (3.232)$$

$$\begin{aligned} \mathbf{S}_{\tilde{\mathbf{y}}_k} &= \text{qr} \left\{ \left[\sqrt{w_1^{(c_1)}} (\mathbf{y}_{1:L,k|k-1} - \mathbf{y}_{L+1:2L,k|k-1}) \right. \right. \\ &\quad \left. \left. \sqrt{w_1^{(c_2)}} (\mathbf{y}_{1:L,k|k-1} - \mathbf{y}_{L+1:2L,k|k-1} - 2\mathbf{y}_{0,k|k-1}) \right] \right\} \end{aligned} \quad (3.233)$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sqrt{w_1^{(c_1)}} \mathbf{S}_{\mathbf{x}_k}^- [\mathbf{y}_{1:L,k|k-1} - \mathbf{y}_{L+1:2L,k|k-1}]^T \quad (3.234)$$

$$\mathbf{K}_k = (\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} / \mathbf{S}_{\tilde{\mathbf{y}}_k}^T) / \mathbf{S}_{\tilde{\mathbf{y}}_k} \quad (3.235)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (3.236)$$

$$\mathbf{U} = \mathbf{K}_k \mathbf{S}_{\tilde{\mathbf{y}}_k} \quad (3.237)$$

$$\mathbf{S}_{\mathbf{x}_k} = \text{cholupdate}\{\mathbf{S}_{\mathbf{x}_k}^-, \mathbf{U}, -1\} \quad (3.238)$$

- Parameters: $\mathbf{x}^{a_v} = [\mathbf{x}^T \ \mathbf{v}^T]^T$, $\boldsymbol{\chi}^{a_v} = [(\boldsymbol{\chi}^x)^T \ (\boldsymbol{\chi}^v)^T]^T$, $\mathbf{x}^{a_n} = [\mathbf{x}^T \ \mathbf{n}^T]^T$, $\boldsymbol{\chi}^{a_n} = [(\boldsymbol{\chi}^x)^T \ (\boldsymbol{\chi}^n)^T]^T$, $h \geq 1$ is the scalar central difference step size, L is the dimension of the augmented states, \mathbf{R}_v is the process-noise covariance, \mathbf{R}_n is the observation-noise covariance, and w_i are the weights as calculated in Equation 3.85. $(\cdot)^2$ is shorthand for the vector outer product.

Algorithm 13 : The square-root CDKF (SR-CDKF) - state estimation (additive noise)

- *Initialization:* $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$, $\mathbf{S}_{\mathbf{x}_0} = \text{chol}\{E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]\}$, $\mathbf{S}_{\mathbf{v}} = \sqrt{\mathbf{R}_{\mathbf{v}}}$, $\mathbf{S}_{\mathbf{n}} = \sqrt{\mathbf{R}_{\mathbf{n}}}$
- *For* $k = 1, \dots, \infty$:

1. Calculate sigma-points for time-update:

$$\boldsymbol{\chi}_{k-1} = \begin{bmatrix} \hat{\mathbf{x}}_{k-1} & \hat{\mathbf{x}}_{k-1} + h\mathbf{S}_{\mathbf{x}_{k-1}} & \hat{\mathbf{x}}_{k-1} - h\mathbf{S}_{\mathbf{x}_{k-1}} \end{bmatrix} \quad (3.239)$$

2. Time-update equations:

$$\boldsymbol{\chi}_{k|k-1} = \mathbf{f}(\boldsymbol{\chi}_{k-1}, \mathbf{u}_{k-1}) \quad (3.240)$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\chi}_{i,k|k-1} \quad (3.241)$$

$$\begin{aligned} \mathbf{S}_{\mathbf{x}_k}^- &= \text{qr} \left\{ \begin{bmatrix} \sqrt{w_1^{(c_1)}} (\boldsymbol{\chi}_{1:L,k|k-1}^x - \boldsymbol{\chi}_{L+1:2L,k|k-1}^x) \\ \sqrt{w_1^{(c_2)}} (\boldsymbol{\chi}_{1:L,k|k-1}^x + \boldsymbol{\chi}_{L+1:2L,k|k-1}^x - 2\boldsymbol{\chi}_{0,k|k-1}^x) & \mathbf{S}_{\mathbf{v}} \end{bmatrix} \right\} \end{aligned} \quad (3.242)$$

3. Calculate sigma-points for measurement-update:

$$\boldsymbol{\chi}_{k|k-1}^* = [\hat{\mathbf{x}}_k^- \quad \hat{\mathbf{x}}_k^- + h\mathbf{S}_{\mathbf{x}_k}^- \quad \hat{\mathbf{x}}_k^- - h\mathbf{S}_{\mathbf{x}_k}^-] \quad (3.243)$$

4. Measurement-update equations:

$$\boldsymbol{\gamma}_{k|k-1} = \mathbf{h}(\boldsymbol{\chi}_{k|k-1}^*) \quad (3.244)$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\gamma}_{i,k|k-1} \quad (3.245)$$

$$\begin{aligned} \mathbf{S}_{\tilde{\mathbf{y}}_k} &= \text{qr} \left\{ \begin{bmatrix} \sqrt{w_1^{(c_1)}} (\boldsymbol{\gamma}_{1:L,k|k-1} - \boldsymbol{\gamma}_{L+1:2L,k|k-1}) \\ \sqrt{w_1^{(c_2)}} (\boldsymbol{\gamma}_{1:L,k|k-1} - \boldsymbol{\gamma}_{L+1:2L,k|k-1} - 2\boldsymbol{\gamma}_{0,k|k-1}) & \mathbf{S}_{\mathbf{n}} \end{bmatrix} \right\} \end{aligned} \quad (3.246)$$

$$\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} = \sqrt{w_1^{(c_1)}} \mathbf{S}_{\mathbf{x}_k}^- [\boldsymbol{\gamma}_{1:L,k|k-1} - \boldsymbol{\gamma}_{L+1:2L,k|k-1}]^T \quad (3.247)$$

$$\mathbf{K}_k = (\mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} / \mathbf{S}_{\tilde{\mathbf{y}}_k}^T) / \mathbf{S}_{\tilde{\mathbf{y}}_k} \quad (3.248)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (3.249)$$

$$\mathbf{U} = \mathbf{K}_k \mathbf{S}_{\tilde{\mathbf{y}}_k} \quad (3.250)$$

$$\mathbf{S}_{\mathbf{x}_k} = \text{cholupdate}\{\mathbf{S}_{\mathbf{x}_k}^-, \mathbf{U}, -1\} \quad (3.251)$$

- *Parameters:* $h \geq 1$ is the scalar central difference step size, L is the dimension of the state, \mathbf{R}_v is the process-noise covariance, \mathbf{R}_n is the observation-noise covariance, and w_i are the weights as calculated in Equation 3.85. $(\cdot)^2$ is shorthand for the vector outer product.
-

In order to test the performance of the square-root SPKFs and compare it with the non-square-root forms, we repeated the nonlinear time-series estimation experiment of Section 3.5.1. We used a UKF, CDKF, SR-UKF and SR-CDKF to estimate the underlying clean state of a Mackey-Glass-30 chaotic time-series that has been corrupted by additive white Gaussian noise ($3dB$ SNR). The average estimation error statistics are summarized in Table 3.3. Clearly there are no difference (at least to the fourth decimal place) between the square-root and non-square-root versions of the SPKF. As before, a very small difference ($\approx 0.2\%$) in estimation accuracy exist between the UKF and CDKF (as well as their square-root versions). The CDKF has a slightly better mean estimation error, but higher variance than the UKF. These differences are so small that they can be ignored for all practical purposes. The choice of which SPKF to use is thus clear: If the form of the specific inference problem allows for the use of a square-root form, do so, since it increases the numerical robustness of the filter. However, the choice of if a SR-UKF or SR-CDKF must be used is simply a matter of implementational choice.

Table 3.3: Square-root SPKF state estimation results on Mackey-Glass chaotic time-series: Mean and variance of estimation MSE for 200 Monte-Carlo runs.

Algorithm	MSE (mean)	MSE (var)
Unscented Kalman filter (UKF)	0.1136	0.0336
Square-root UKF (SR-UKF)	0.1136	0.0336
Central difference Kalman filter (CDKF)	0.1134	0.0376
Square-root CDKF (SR-CDKF)	0.1134	0.0376

Square-root Parameter Estimation

The parameter estimation algorithm follows a similar framework as that of the state estimation square-root SPKF. However, an $\mathcal{O}(ML^2)$ algorithm, as opposed to $\mathcal{O}(L^3)$, is possible by taking advantage of the *linear* state transition function. Specifically, the time-update of the state covariance is given simply by

$$\mathbf{P}_{\mathbf{w}_k}^- = \mathbf{P}_{\mathbf{w}_{k-1}} + \mathbf{R}_{\mathbf{r}_k}, \quad (3.252)$$

where $\mathbf{R}_{\mathbf{r}_k}$ is the time-varying covariance of the “artificial” process noise (see Section 3.253 for discussion on selecting and adapting $\mathbf{R}_{\mathbf{r}_k}$). In the square-root filters, $\mathbf{S}_{\mathbf{w}_{k-1}}$ may thus be updated directly in Equations 3.256 and 3.268 using one of two options:

1. *RLS exponential decay*:

$$\mathbf{S}_{\mathbf{w}_k}^- = \lambda_{RLS}^{-1} \mathbf{S}_{\mathbf{w}_{k-1}} \quad (3.253)$$

This option (introduced by Nelson for standard, non-square-root parameter estimation [143]) corresponds to an exponential weighting on past data.

2. *Annealing or Robbins-Monro*:

$$\mathbf{S}_{\mathbf{w}_k}^- = \mathbf{S}_{\mathbf{w}_{k-1}} + \mathbf{D}_{\mathbf{r}_k} \quad (3.254)$$

Here the diagonal matrix $\mathbf{D}_{\mathbf{r}_k}$ is chosen to either a) approximate the effects of annealing a diagonal process noise covariance $\mathbf{R}_{\mathbf{r}_k}^{diag}$, or b) is calculated as the diagonal approximation of a Robbins-Monro stochastic estimate of the driving process noise (See Section 3.5.2). This update ensures the main diagonal of $\mathbf{P}_{\mathbf{w}_k}^-$ is exact. However, additional off-diagonal cross-terms $\mathbf{S}_{\mathbf{w}_{k-1}} \mathbf{D}_{\mathbf{r}_k}^T + \mathbf{D}_{\mathbf{r}_k} \mathbf{S}_{\mathbf{w}_{k-1}}^T$ are also introduced (though the effect appears negligible [191]).

Both options avoid the costly $\mathcal{O}(L^3)$ QR and Cholesky based updates necessary in the state-estimation filter. The computational efficiency of the square-root SPKF parameter estimation algorithms are demonstrated in Figure 3.17 on the *Mackay-Robot-Arm* problem. This is the same neural network training problem we addressed in Section 3.5.2,

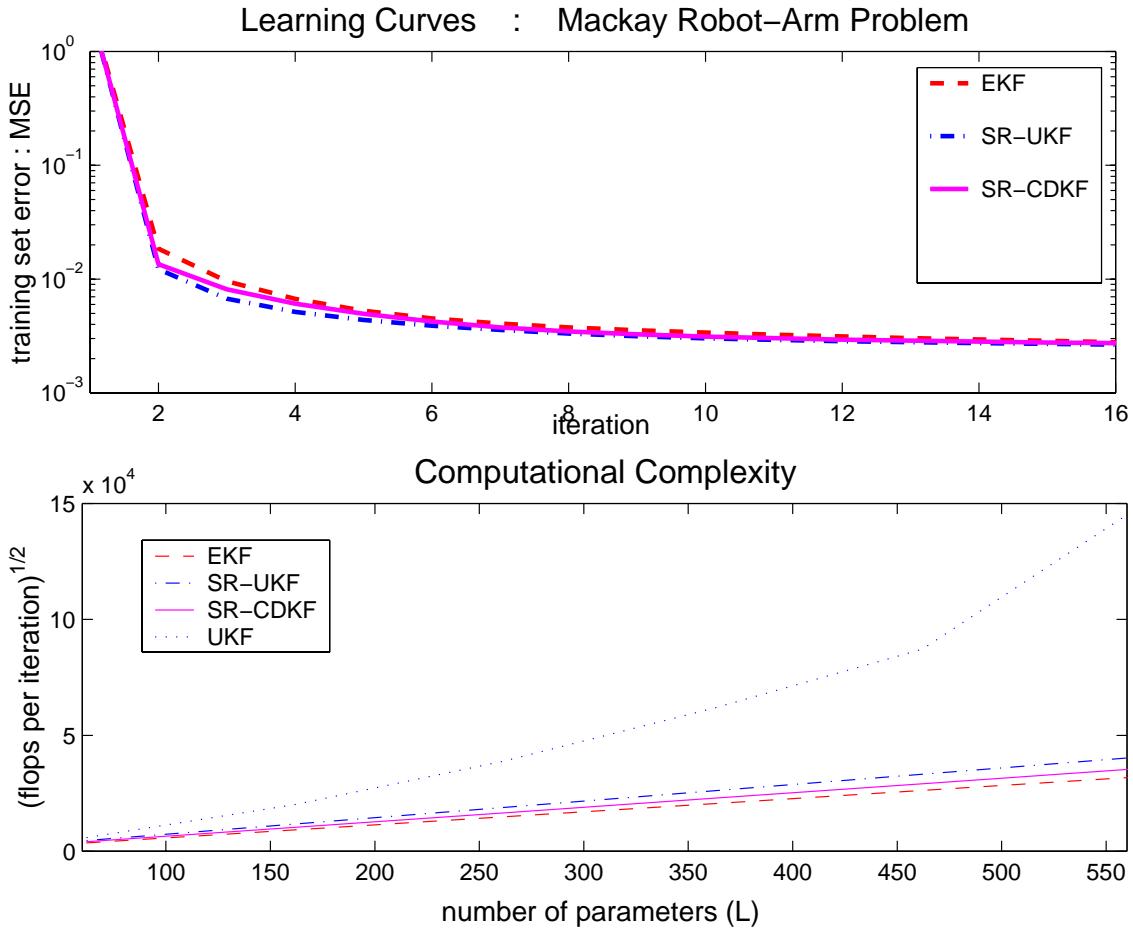


Figure 3.17: Square-root SPKF parameter estimation for *Mackay-Robot-Arm* neural network training problem: (top) Learning curves for EKF, SR-UKF and SR-CDKF (see Figure 3.9 for non-square root SPKF results). (bottom) Computational complexity: flops/epochs vs. number of parameters (neural network complexity).

except in this case we use a square-root SPKF (SR-UKF and SR-CDKF) to train the MLP neural network. We repeat the experiment a number of times, each time increasing the complexity (number of weights per neural network layer) and measure the total number of floating point calculations (flops) required for each iteration of neural network training. This results in the computational complexity graph shown in the bottom plot of Figure 3.17. Clearly the square-root SPKFs (as well as the EKF) are $\mathcal{O}(L^2)$ whereas the non-square-root SPKF (original UKF) is $\mathcal{O}(L^3)$. The claimed improvement in numerical efficiency has been achieved.

Algorithm 14 : The square-root UKF (SR-UKF) - parameter estimation form

- *Initialization:* $\hat{\mathbf{w}}_0 = E[\mathbf{w}]$, $\mathbf{S}_{\mathbf{w}_0} = \text{chol}\{E[(\mathbf{w} - \hat{\mathbf{w}}_0)(\mathbf{w} - \hat{\mathbf{w}}_0)^T]\}$

- *For* $k = 1, \dots, \infty$:

1. Time-update equations:

$$\hat{\mathbf{w}}_k^- = \hat{\mathbf{w}}_{k-1} \quad (3.255)$$

$$\mathbf{S}_{\mathbf{w}_k}^- = \sqrt{\lambda_{RLS}} \mathbf{S}_{\mathbf{w}_{k-1}} \quad \text{or} \quad \mathbf{S}_{\mathbf{w}_k}^- = \mathbf{S}_{\mathbf{w}_{k-1}} + \mathbf{D}_{\mathbf{r}_{k-1}} \quad (3.256)$$

where $\mathbf{D}_{\mathbf{r}_{k-1}} = -\text{diag}\{\mathbf{S}_{\mathbf{w}_{k-1}}\} + \sqrt{\text{diag}\{\mathbf{S}_{\mathbf{w}_{k-1}}\}^2 + \text{diag}\{\mathbf{R}_{\mathbf{r}_{k-1}}\}}$

2. Calculate sigma-points for measurement-update:

$$\boldsymbol{\chi}_{k|k-1} = [\hat{\mathbf{w}}_k^- \quad \hat{\mathbf{w}}_k^- + \gamma \mathbf{S}_{\mathbf{w}_k}^- \quad \hat{\mathbf{x}}_k^- - \gamma \mathbf{S}_{\mathbf{w}_k}^-] \quad (3.257)$$

3. Measurement-update equations:

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{g}(\mathbf{x}_k, \boldsymbol{\chi}_{k|k-1}) \quad (3.258)$$

$$\hat{\mathbf{d}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\mathcal{Y}}_{i,k|k-1} \quad (3.259)$$

$$\mathbf{S}_{\tilde{\mathbf{d}}_k} = \text{qr} \left\{ \left[\sqrt{w_1^{(c)}} (\boldsymbol{\mathcal{Y}}_{1:2L,k|k-1} - \hat{\mathbf{d}}_k^-) \quad \sqrt{\mathbf{S}_{\mathbf{n}}} \right] \right\} \quad (3.260)$$

$$\mathbf{S}_{\tilde{\mathbf{d}}_k} = \text{cholupdate}\{\mathbf{S}_{\tilde{\mathbf{d}}_k}, \boldsymbol{\mathcal{Y}}_{0,k|k-1} - \hat{\mathbf{d}}_k^-, w_0^{(c)}\} \quad (3.261)$$

$$\mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} = \sum_{i=0}^{2L} w_i^{(c)} (\boldsymbol{\chi}_{i,k|k-1} - \hat{\mathbf{w}}_k^-) (\boldsymbol{\mathcal{Y}}_{i,k|k-1} - \hat{\mathbf{d}}_k^-)^T \quad (3.262)$$

$$\mathbf{K}_k = \left(\mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} / \mathbf{S}_{\tilde{\mathbf{d}}_k}^T \right) / \mathbf{S}_{\tilde{\mathbf{d}}_k} \quad (3.263)$$

$$\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_k^- + \mathbf{K}_k (\mathbf{d}_k - \hat{\mathbf{d}}_k^-) \quad (3.264)$$

$$\mathbf{U} = \mathbf{K}_k \mathbf{S}_{\tilde{\mathbf{d}}_k} \quad (3.265)$$

$$\mathbf{S}_{\mathbf{w}_k} = \text{cholupdate}\{\mathbf{S}_{\mathbf{w}_k}^-, \mathbf{U}, -1\} \quad (3.266)$$

- *Parameters:* \mathbf{R}_r is the artificial process-noise covariance and \mathbf{R}_n is the observation-noise covariance. The $\text{diag}\{\cdot\}$ operator nulls all the elements of a square matrix except the main diagonal. $\gamma = \sqrt{L + \lambda}$ is the composite scaling parameter and λ is given by Eq. 3.11. L is the dimension of the state and w_i are the weights as calculated in Equation 3.12. The $+$ and $-$ operators add or subtract a column vector to each column of a matrix.
-

Algorithm 15 : The square-root CDKF (SR-CDKF) - parameter estimation form

- *Initialization:* $\hat{\mathbf{w}}_0 = E[\mathbf{w}]$, $\mathbf{S}_{\mathbf{w}_0} = \text{chol}\{E[(\mathbf{w} - \hat{\mathbf{w}}_0)(\mathbf{w} - \hat{\mathbf{w}}_0)^T]\}$
- *For* $k = 1, \dots, \infty$:

1. Time-update equations:

$$\hat{\mathbf{w}}_k^- = \hat{\mathbf{w}}_{k-1} \quad (3.267)$$

$$\mathbf{S}_{\mathbf{w}_k}^- = \sqrt{\lambda_{RLS}} \mathbf{S}_{\mathbf{w}_{k-1}} \quad \text{or} \quad \mathbf{S}_{\mathbf{w}_k}^- = \mathbf{S}_{\mathbf{w}_{k-1}} + \mathbf{D}_{\mathbf{r}_{k-1}}, \quad (3.268)$$

$$\text{where } \mathbf{D}_{\mathbf{r}_{k-1}} = -\text{diag}\{\mathbf{S}_{\mathbf{w}_{k-1}}\} + \sqrt{\text{diag}\{\mathbf{S}_{\mathbf{w}_{k-1}}\}^2 + \text{diag}\{\mathbf{R}_{\mathbf{r}_{k-1}}\}}$$

2. Calculate sigma-points for measurement-update:

$$\boldsymbol{\chi}_{k|k-1} = [\hat{\mathbf{w}}_k^- \quad \hat{\mathbf{w}}_k^- + h\mathbf{S}_{\mathbf{w}_k}^- \quad \hat{\mathbf{x}}_k^- - h\mathbf{S}_{\mathbf{w}_k}^-] \quad (3.269)$$

3. Measurement-update equations:

$$\boldsymbol{\gamma}_{k|k-1} = \mathbf{g}(\mathbf{x}_k, \boldsymbol{\chi}_{k|k-1}) \quad (3.270)$$

$$\hat{\mathbf{d}}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \boldsymbol{\gamma}_{i,k|k-1} \quad (3.271)$$

$$\begin{aligned} \mathbf{S}_{\tilde{\mathbf{d}}_k} &= \text{qr} \left\{ \left[\sqrt{w_1^{(c_1)}} (\boldsymbol{\gamma}_{1:L,k|k-1} - \boldsymbol{\gamma}_{L+1:2L,k|k-1}) \right. \right. \\ &\quad \left. \left. \sqrt{w_1^{(c_2)}} (\boldsymbol{\gamma}_{1:L,k|k-1} - \boldsymbol{\gamma}_{L+1:2L,k|k-1} - 2\boldsymbol{\gamma}_{0,k|k-1}) \quad \sqrt{\mathbf{S}_{\mathbf{n}}} \right] \right\} \end{aligned} \quad (3.272)$$

$$\mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} = \sqrt{w_i^{(c_1)}} \mathbf{S}_{\mathbf{w}_k}^- [\boldsymbol{\gamma}_{1:L,k|k-1} - \boldsymbol{\gamma}_{L+1:2L,k|k-1}]^T \quad (3.273)$$

$$\mathbf{K}_k = \left(\mathbf{P}_{\mathbf{w}_k \mathbf{d}_k} / \mathbf{S}_{\tilde{\mathbf{d}}_k}^T \right) / \mathbf{S}_{\tilde{\mathbf{d}}_k} \quad (3.274)$$

$$\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_k^- + \mathbf{K}_k (\mathbf{d}_k - \hat{\mathbf{d}}_k^-) \quad (3.275)$$

$$\mathbf{U} = \mathbf{K}_k \mathbf{S}_{\tilde{\mathbf{d}}_k} \quad (3.276)$$

$$\mathbf{S}_{\mathbf{w}_k} = \text{cholupdate}\{\mathbf{S}_{\mathbf{w}_k}^-, \mathbf{U}, -1\} \quad (3.277)$$

- *Parameters:* \mathbf{R}_r is the artificial process-noise covariance and \mathbf{R}_n is the observation-noise covariance (See Section 3.5.2 for detail on how these should be calculated/adapted. The $\text{diag}\{\cdot\}$ operator nulls all the elements of a square matrix except the main diagonal. $h \geq 1$ is the scalar central difference step size, L is the dimension of the state, , and w_i are the weights as calculated in Equation 3.85. $(\cdot)^2$ is shorthand for the vector outer product.
-

3.6.3 SPKF Based Smoothing

The sigma-point Kalman smoother: As has been discussed, the Kalman filter is a recursive algorithm providing the conditional expectation of the state \mathbf{x}_k given all observations $\mathbf{Y}_k = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_k\}$ up to the current time k . In contrast, the Kalman smoother, estimates the state given all observations past and future, $\mathbf{Y}_N = \{\mathbf{y}_0, \mathbf{y}_2, \dots, \mathbf{y}_N\}$, where N is the final time. Kalman smoothers are commonly used for applications such as trajectory planning, noncausal noise reduction, and the *E-step* in the EM-Algorithm [176, 66]. A thorough treatment of the Kalman smoother in the linear case is given in [116]. The basic idea is to run a Kalman filter forward in time to estimate the mean and covariance $(\hat{\mathbf{x}}_{k|k}^f, \mathbf{P}_{\mathbf{x}_{k|k}}^f)$ of the state given *past* data. A second Kalman filter is then run backward in time to produce a backward-time predicted mean and covariance $(\hat{\mathbf{x}}_{k|k+1}^b, \mathbf{P}_{\mathbf{x}_{k|k+1}}^b)$ ¹⁴ given the *future* data. These two estimates are then combined, producing the following *smoothed* statistics given *all* the data:

$$(\mathbf{P}_{\mathbf{x}_k}^s)^{-1} = (\mathbf{P}_{\mathbf{x}_{k|k}}^f)^{-1} + (\mathbf{P}_{\mathbf{x}_{k|k+1}}^b)^{-1} \quad (3.278)$$

$$\hat{\mathbf{x}}_k^s = \mathbf{P}_{\mathbf{x}_k}^s \left[(\mathbf{P}_{\mathbf{x}_{k|k}}^f)^{-1} \hat{\mathbf{x}}_{k|k}^f + (\mathbf{P}_{\mathbf{x}_{k|k+1}}^b)^{-1} \hat{\mathbf{x}}_{k|k+1}^b \right] \quad (3.279)$$

For the nonlinear case, the EKF replaces the Kalman filter. The use of the EKF for the forward filter is straightforward. However, implementation of the backward filter is achieved by using the following linearized backward-time system:

$$\mathbf{x}_{k-1} = \mathbf{A}^{-1} \mathbf{x}_k - \mathbf{A}^{-1} \mathbf{B} \mathbf{v}_k , \quad (3.280)$$

that is, the forward nonlinear dynamics are linearized, and then inverted for the backward model. A linear Kalman filter is then applied.

Our proposed *sigma-point Kalman smoother* (SPKS), replaces the EKF with a SPKF. In addition, we consider using a nonlinear backward model as well, either derived from first principles, or by training a backward predictor using a neural network model, as illustrated for the time series case in Figure 3.18. The nonlinear backward model allows us to take

¹⁴The notation $\hat{\mathbf{x}}_{i|j}$ means the estimate of \mathbf{x} at time i based on information at time j . The backward predicted state at time k , based on information at time $k+1$, is thus written as, $\hat{\mathbf{x}}_{k|k+1}^b$.

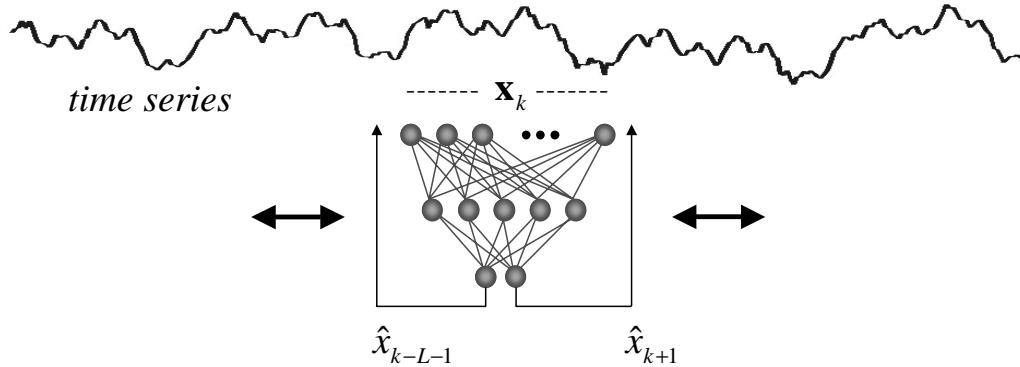


Figure 3.18: Forward/backward neural network prediction training for sigma-point Kalman smoother.

Table 3.4: Comparison of smoother performance: The estimation results (normalized MSE) for two nonlinear time-series estimation problems are summarized. On the left the results for the Mackey-Glass experiment is given and on the right the results for an chaotic autoregressive neural network. [EKS1 = extended Kalman smoother with linearized backward model; EKS2 = extended Kalman smoother with second nonlinear backward model; SPKS = sigma-point Kalman smoother using forward-backward neural predictor shown in Figure 3.18.

Mackey-Glass				Chaotic AR-NN			
Algorithm	Normalized MSE			Algorithm	Normalized MSE		
	forward	backward	smoothed		forward	backward	smoothed
EKS1	0.20	0.70	0.27	EKS1	0.35	0.32	0.28
EKS2	0.20	0.31	0.19	EKS2	0.35	0.22	0.23
SPKS	0.10	0.24	0.08	SPKS	0.23	0.21	0.16

full advantage of the SPKF, which requires no linearization step.

To illustrate performance, we reconsider the noisy Mackey-Glass time-series problem of the previous section, as well as a second time series generated using chaotic autoregressive neural network. The table below compares *smoother* performance. In this case, the network models are trained on the clean time-series, and then tested on the noisy data using either the standard extended Kalman smoother with linearized backward model (EKS1), an extended Kalman smoother with a second nonlinear backward model (EKS2), and the sigma-point Kalman smoother (SPKS). The forward, backward, and smoothed estimation errors are reported. Again, the performance benefits of the sigma-point approach is clear.

3.7 Chapter Summary

In this chapter the sigma-point Kalman filter (SPKF) was introduced as a better alternative to the EKF for Gaussian approximate probabilistic inference in general nonlinear DSSMs. The underlying unifying technique common to all SPKFs, the sigma-point approach, was introduced as a method to calculate the statistics of a random variable that undergoes a nonlinear transformation. These calculations form the core of the optimal Kalman time and measurement update equations, which is simply the original (optimal) recursive Bayesian estimation integral equations recast under a Gaussian assumption.

We showed how two SPKF algorithms, the UKF and CDKF, although derived from different starting assumptions, both employ the sigma-point approach as their core algorithmic component for calculating the posterior Gaussian statistics necessary to for Gaussian approximate inference in a Kalman framework. In Chapter 4 we will show, through a theoretical analysis, how both these approaches (and all SPKFs for that matter) achieve essentially the same level of accuracy that surpasses that of the EKF. This result was already foreshadowed in this chapter through experimental verification.

In Section 3.5 we demonstrated through a large number of experimental results how we extended the use of the SPKF from its original state-estimation for vehicular control origins, to the larger complete probabilistic inference field, namely. state estimation (with new application to nonlinear time-series prediction), parameter estimation (machine learning and system ID) and dual-estimation. Through these experiments we verified the consistently better or equal performance of the SPKF compared to that of the EKF.

Other detail covered in Section 3.5 is the discussion of a number of process-noise estimation and adaptation methods with specific application to parameter estimation. We also showed how the SPKF (for parameter estimation) can be modified to minimize general non-MSE cost functions (e.g. entropic cost, etc.), allowing for its application to a larger group of general optimization problems.

Section 3.6 covers the derivation of a number of implementational variations of the SPKF that provide better computational efficiency and increased numerical robustness. Starting from the original state-estimation UKF and CDKF, a number of new SPKF

algorithms, including numerically stable (robust) square-root forms, were derived that take advantage of the specific form of the DSSM for certain state, parameter and dual estimation problems in order to increase computational efficiency.

Throughout this chapter, where relevant, we presented detailed implementation orientated summaries of all of the derived and discussed SPKF algorithms. These allow for easy verifiable implementation. This fact notwithstanding, all of the discussed algorithms are also available in pre-coded Matlab form as part of the ReBEL toolkit. See Appendix C for more detail.

Chapter 4

Sigma-Point Kalman Filters: Theoretical Analysis

4.1 Introduction

In this chapter some of the theoretical properties of the SPKF family of filters are analyzed. We first present an alternative (partial) interpretation of the sigma-point approach based on a technique called *weighted statistical linear regression*, which allows for further useful insight into why the SPKF is expected to perform better and more robustly than the EKF for Gaussian approximate nonlinear estimation. After that, we will determine the accuracy with which the SPKF calculates the posterior statistics of the random variables they operate on. We will show how all SPKFs achieve at least second order accuracy in the calculation of the posterior mean and covariance. In Section 4.4 we briefly comment on some of the apparent similarities between the sigma-point approach and a numerical integration method called Gaussian quadrature.

One of the longer and more important sections of this chapter, which also constitutes one of the original contributions of this work, is the analysis of the SPKF for parameter estimation presented in Section 4.5. In this section, we show the relationship between the SPKF and other second order nonlinear optimization methods, casting it into a general online (stochastic) *adaptive modified Newton method* framework that minimizes an instantaneous nonlinear least squares cost function.

We conclude this chapter by a brief summary of the most salient and important characteristics of the SPKF family of algorithms as revealed by the derivations in the previous

chapter and the analysis of this chapter.

4.2 Alternate Interpretation of Sigma-Point Approach

As pointed out in the previous chapter, all sigma-point Kalman filters utilize a general deterministic sampling framework called *the sigma-point approach*, to calculate Gaussian approximations to the optimal Bayesian recursion terms (Equations 2.5, 2.6 and 2.8). Based on an analysis of the UKF by Lefebvre [114], it is possible to interpret the sigma-point approach as implicitly performing *statistical linearization* [62] of the nonlinear functions in question through the implicit use of a technique known as *weighted statistical linear regression* (WSLR). WSLR is a statistical linearization technique that takes into account the uncertainty or “probabilistic spread” of the prior random variable when linearizing a nonlinear function that operates on that random variable (takes it as an argument). By doing so, the resulting linearized function is more accurate in an average statistical sense, than simply using a first order truncated Taylor-series expansion of the function around a single point, say the prior mean.

In Chapter 2 (Section 2.4) we indicated and demonstrated how the EKF’s inaccuracies stem from the fact that it *does not* take into account the probabilistic spread of the prior state random variables when it linearizes the process and observation models using analytical first-order Taylor series truncations. This implies the assumption that the prior random variables are highly “peaked up” around their mean values, resulting in the highly inaccurate approximations of the posterior statistics when the nonlinearities in question are significant over the true distribution of the prior random variable.

We will next introduce the concepts of statistical linearization and weighted statistical linear regression (WSLR) and show how the sigma-point approach, and hence all SPKFs, makes implicit use of it. It is important to note at this point however, that a WSLR interpretation of the sigma-point approach does not capture all of its salient features. For instance, it is possible to extend the sigma-point approach through the use of more samples and careful constrained weight/location optimization, to capture and accurately propagate higher order moment information such as skew, kurtosis, etc. [96, 101]. These extra

properties of the sigma-point approach falls outside the narrower interpretation framework that the WSLR technique offers. That said, it does not detract from the validity and utility of using the WSLR interpretation to analyze the general 2nd order sigma-point approach framework.

4.2.1 Statistical linearization

The following exposition follows that of Gelb in [62] closely. We seek a linear approximation of a general nonlinear vector function $\mathbf{g}(\mathbf{x})$, operating on vector random variable \mathbf{x} with probability density function $p(\mathbf{x})$, i.e.,

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) \approx \mathbf{Ax} + \mathbf{b} , \quad (4.1)$$

where \mathbf{A} and \mathbf{b} are a matrix and a vector to be determined. The aim here is determine the linearization parameters, \mathbf{A} and \mathbf{b} , such that the approximation takes into account the prior distribution of \mathbf{x} and hence perform better in an average (statistical) sense than a simple first order Taylor series truncation. Defining the approximation error as

$$\boldsymbol{\varepsilon} \doteq \mathbf{g}(\mathbf{x}) - \mathbf{Ax} - \mathbf{b} , \quad (4.2)$$

we desire to choose \mathbf{A} and \mathbf{b} such that the quantity

$$J = E [\boldsymbol{\varepsilon}^T \mathbf{W} \boldsymbol{\varepsilon}] \quad (4.3)$$

is minimized for some symmetric positive semidefinite matrix \mathbf{W} . Substituting Equation 4.2 into Equation 4.3 and setting the partial derivative of J with respect to the elements of \mathbf{b} equal to zero, we obtain:

$$E [\mathbf{W} (\mathbf{g}(\mathbf{x}) - \mathbf{Ax} - \mathbf{b})] = \mathbf{0} . \quad (4.4)$$

Therefore, \mathbf{b} is given by

$$\begin{aligned}\mathbf{b} &= E[\mathbf{g}(\mathbf{x})] - \mathbf{A}E[\mathbf{x}] \\ &= \bar{\mathbf{y}} - \mathbf{A}\bar{\mathbf{x}}.\end{aligned}\quad (4.5)$$

Substituting \mathbf{b} from Equation 4.5 into J and taking the partial derivative with respect to the elements of \mathbf{A} , we obtain

$$E[\mathbf{W}[\mathbf{A}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T + (\mathbf{g}(\mathbf{x}) - \bar{\mathbf{y}})\tilde{\mathbf{x}}^T]] = \mathbf{0}, \quad (4.6)$$

where $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$. Solving Equation 4.6 for \mathbf{A} gives

$$\begin{aligned}\mathbf{A} &= E[(\mathbf{g}(\mathbf{x}) - \bar{\mathbf{y}})\tilde{\mathbf{x}}^T]E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T]^{-1} \\ &= E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{y} - \bar{\mathbf{y}})^T]^T E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T]^{-1} \\ &= \mathbf{P}_{\mathbf{xy}}^T \mathbf{P}_{\mathbf{x}}^{-1},\end{aligned}\quad (4.7)$$

where $\mathbf{P}_{\mathbf{x}}$ is the covariance matrix of \mathbf{x} and $\mathbf{P}_{\mathbf{xy}}$ is the cross-covariance matrix of \mathbf{x} and $\mathbf{y} = \mathbf{g}(\mathbf{x})$. Notice that both \mathbf{b} and \mathbf{A} as given by Equations 4.5 and 4.7, are independent of the weighting matrix \mathbf{W} ; hence, they provide a generalized minimum mean square error linearized approximation of $\mathbf{g}(\mathbf{x})$.

To summarize, statistical linearization as presented above, derives a MMSE optimal linear approximation of a general nonlinear function that takes into account the prior mean and covariance of the random variable it operates upon. This linearization will be more accurate in statistical sense, with respect to the distribution of the prior random variable, than a simple first order Taylor-series truncation approach. Next we will show how statistical linearization can be realized through the use of a method called weighted statistical linear regression and how it relates to the sigma-point approach

4.2.2 Weighted statistical linear regression (WSLR)

Consider a nonlinear function $\mathbf{y} = \mathbf{g}(\mathbf{x})$ which is evaluated in N points $(\chi_i, i = 1 \dots N)$, i.e.,

$$\gamma_i = \mathbf{g}(\chi_i), i = 1 \dots N, \quad (4.8)$$

where the points χ_i are chosen such that they capture certain statistical properties of \mathbf{x} , such as the mean and covariance, through the sample based estimators of the following general form:

$$\bar{\mathbf{x}} = \sum_{i=1}^N w_i \chi_i \quad (4.9)$$

$$\mathbf{P}_x = \sum_{i=1}^N w_i (\chi_i - \bar{\mathbf{x}})(\chi_i - \bar{\mathbf{x}})^T, \quad (4.10)$$

where w_i is a set of N scalar regression weights such that

$$\sum_{i=1}^N w_i = 1. \quad (4.11)$$

Now, the aim is to find the linear regression

$$\mathbf{y} = \mathbf{Ax} + \mathbf{b} \quad (4.12)$$

that minimizes the weighted sum of squared errors

$$\{\mathbf{A}, \mathbf{b}\} = \operatorname{argmin} \sum_{i=1}^N w_i \boldsymbol{\varepsilon}_i^T \boldsymbol{\varepsilon}_i, \quad (4.13)$$

where the point-wise linearization error $\boldsymbol{\varepsilon}_i$ is defined as

$$\boldsymbol{\varepsilon}_i = \gamma_i - (\mathbf{A}\chi_i + \mathbf{b}). \quad (4.14)$$

Equation 4.13 can be interpreted as a finite sample based approximation of the true expected statistical linearization cost given by Equation 4.3, i.e.,

$$J = E [\boldsymbol{\varepsilon}^T \mathbf{W} \boldsymbol{\varepsilon}] \approx \sum_{i=1}^N w_i \boldsymbol{\varepsilon}_i^T \boldsymbol{\varepsilon}_i . \quad (4.15)$$

On the left hand side of Equation 4.15, the expectation is taken with respect to the distribution of \mathbf{x} , $p(\mathbf{x})$. Since the regression points $\boldsymbol{\chi}_i$ are chosen such that they capture the prior mean and covariance of \mathbf{x} , the right-hand side of the equation is a valid approximation of the true expected cost under a Gaussian approximate assumption.

If we now further define the following estimators of the posterior (Gaussian approximate) statistics of the propagated regression points $\boldsymbol{\gamma}_i$,

$$\bar{\mathbf{y}} \approx \hat{\mathbf{y}} = \sum_{i=1}^N w_i \boldsymbol{\gamma}_i \quad (4.16)$$

$$\mathbf{P}_{\mathbf{y}} \approx \hat{\mathbf{P}}_{\mathbf{y}} = \sum_{i=1}^N w_i (\boldsymbol{\gamma}_i - \hat{\mathbf{y}})(\boldsymbol{\gamma}_i - \hat{\mathbf{y}})^T \quad (4.17)$$

$$\mathbf{P}_{\mathbf{x}\mathbf{y}} \approx \hat{\mathbf{P}}_{\mathbf{x}\mathbf{y}} = \sum_{i=1}^N w_i (\boldsymbol{\chi}_i - \bar{\mathbf{x}})(\boldsymbol{\gamma}_i - \hat{\mathbf{y}})^T , \quad (4.18)$$

we can use the statistical linearization results from the previous section to solve for \mathbf{A} and \mathbf{b} in Equations 4.12 and 4.13. This weighted statistical linear regression solution is thus given by:

$$\mathbf{A} = \hat{\mathbf{P}}_{\mathbf{x}\mathbf{y}}^T \mathbf{P}_{\mathbf{x}}^{-1} \quad (4.19)$$

$$\mathbf{b} = \hat{\mathbf{y}} - \mathbf{A} \bar{\mathbf{x}} . \quad (4.20)$$

The covariance of the statistical linearization error can be approximated by

$$\begin{aligned}
\mathbf{P}_\epsilon &\approx \sum_{i=1}^N w_i \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T \\
&= \sum_{i=1}^N w_i [\boldsymbol{\gamma}_i - (\mathbf{A}\boldsymbol{\chi}_i + \mathbf{b})] [\boldsymbol{\gamma}_i - (\mathbf{A}\boldsymbol{\chi}_i + \mathbf{b})]^T \\
&= \sum_{i=1}^N w_i [\boldsymbol{\gamma}_i - \mathbf{A}\boldsymbol{\chi}_i - \hat{\mathbf{y}} + \mathbf{A}\bar{\mathbf{x}}] [\boldsymbol{\gamma}_i - \mathbf{A}\boldsymbol{\chi}_i - \hat{\mathbf{y}} + \mathbf{A}\bar{\mathbf{x}}]^T \\
&= \sum_{i=1}^N w_i [(\boldsymbol{\gamma}_i - \hat{\mathbf{y}}) - \mathbf{A}(\boldsymbol{\chi}_i - \bar{\mathbf{x}})] [(\boldsymbol{\gamma}_i - \hat{\mathbf{y}}) - \mathbf{A}(\boldsymbol{\chi}_i - \bar{\mathbf{x}})]^T \\
&= \hat{\mathbf{P}}_{\mathbf{y}} - \mathbf{A}\hat{\mathbf{P}}_{\mathbf{xy}} - \hat{\mathbf{P}}_{\mathbf{xy}}^T \mathbf{A}^T + \mathbf{A}\mathbf{P}_{\mathbf{x}}\mathbf{A}^T \\
&= \hat{\mathbf{P}}_{\mathbf{y}} - \mathbf{A}\mathbf{P}_{\mathbf{x}}\mathbf{A}^T,
\end{aligned} \tag{4.21}$$

where we substituted¹ $\hat{\mathbf{P}}_{\mathbf{xy}}^T = \mathbf{A}\mathbf{P}_{\mathbf{x}}$ from Equation 4.19.

Once this statistical linearization has been determined we can now approximate the nonlinear function $\mathbf{g}(\cdot)$ as

$$\begin{aligned}
\mathbf{y} &= \mathbf{g}(\mathbf{x}) \\
&\cong \mathbf{Ax} + \mathbf{b} + \boldsymbol{\varepsilon},
\end{aligned} \tag{4.22}$$

where $\boldsymbol{\varepsilon}$ is the linearization error as defined above. The posterior statistics of \mathbf{y} can now also be expressed (approximated) by the following statistical linear regression forms

$$\hat{\mathbf{y}}^{slr} \doteq \mathbf{A}\bar{\mathbf{x}} + \mathbf{b} \tag{4.23}$$

$$\hat{\mathbf{P}}_{\mathbf{y}}^{slr} \doteq \mathbf{A}\mathbf{P}_{\mathbf{x}}\mathbf{A}^T + \mathbf{P}_\epsilon. \tag{4.24}$$

where we use the expected value of Equation 4.22 for the mean approximation and the expected value of its outer product for the covariance approximation. Note how the variance of the linearization error, \mathbf{P}_ϵ , is added back into linearly propagated prior covariance, $\mathbf{A}\mathbf{P}_{\mathbf{x}}\mathbf{A}^T$, when calculating $\hat{\mathbf{P}}_{\mathbf{y}}^{slr}$. This correction term makes intuitive sense: The more severe the nonlinearity is over the “uncertainty region” of \mathbf{x} , the larger our linearization

¹We assume the covariance matrix of \mathbf{x} , $\mathbf{P}_{\mathbf{x}}$, is symmetric and positive-definite and hence invertible.

error and error covariance will be, and accordingly the normal linear approximation of \mathbf{P}_y , $\mathbf{A}\hat{\mathbf{P}}_x\mathbf{A}^T$, will be less accurate. The correction term \mathbf{P}_ϵ thus needs to be larger to compensate.

4.2.3 Relationship between WSLR and the sigma-point approach

Using the results from the previous section and the general form of the sigma-point approach equations (See Chapter 3), we will now show how the sigma-point approach makes implicit use of statistical linear regression to obtain an implied statistically linearized DSSM for use within the Kalman filter framework.

If we substitute Equation 4.16 into Equation 4.20 and the result into Equation 4.23, we obtain

$$\begin{aligned}\hat{\mathbf{y}}^{slr} &= \mathbf{A}\bar{\mathbf{x}} + \sum_{i=1}^N w_i \boldsymbol{\gamma}_i - \mathbf{A}\bar{\mathbf{x}} \\ &= \sum_{i=1}^N w_i \boldsymbol{\gamma}_i.\end{aligned}\quad (4.25)$$

If we now assume that the prior regression points $\boldsymbol{\chi}_i$ are chosen using a sigma-point selection scheme as used by the sigma-point approach, i.e.,

$$\begin{aligned}\boldsymbol{\chi}_i &\equiv \boldsymbol{\chi}_i & i=0, \dots, 2L \\ \boldsymbol{\chi}_0 &= \bar{\mathbf{x}} & i=0 \\ \boldsymbol{\chi}_i &= \bar{\mathbf{x}} + \zeta (\sqrt{\mathbf{P}_x})_i & i=1, \dots, L \\ \boldsymbol{\chi}_i &= \bar{\mathbf{x}} - \zeta (\sqrt{\mathbf{P}_x})_i & i=L+1, \dots, 2L\end{aligned},\quad (4.26)$$

where ζ is a scaling parameter, then

$$\begin{aligned}\boldsymbol{\gamma}_i &= \mathbf{g}(\boldsymbol{\chi}_i) \\ &= \mathbf{g}(\boldsymbol{\chi}_i) \\ &= \mathbf{y}_i\end{aligned}$$

is equivalent to the posterior (propagated) set of sigma-points as used by the sigma-point approach. Under these assumptions, the weighted statistical linear regression calculated

posterior mean $\hat{\mathbf{y}}^{slr}$ (Equation 4.25) is equivalent to the sigma-point calculated posterior mean as given by Equation 3.14.

Likewise, if we substitute Equation 4.17 into Equation 4.21 and the result into Equations 4.24, we obtain

$$\begin{aligned}\hat{\mathbf{P}}_{\mathbf{y}}^{slr} &= \mathbf{A}\mathbf{P}_{\mathbf{x}}\mathbf{A}^T + \sum_{i=1}^N w_i(\boldsymbol{\gamma}_i - \bar{\mathbf{y}})(\boldsymbol{\gamma}_i - \bar{\mathbf{y}})^T - \mathbf{A}\mathbf{P}_{\mathbf{x}}\mathbf{A}^T \\ &= \sum_{i=1}^N w_i(\boldsymbol{\gamma}_i - \bar{\mathbf{y}})(\boldsymbol{\gamma}_i - \bar{\mathbf{y}})^T,\end{aligned}\quad (4.27)$$

$$= \sum_{i=1}^N w_i(\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^T \quad (4.28)$$

which, under the same assumptions as above, is equivalent to the way that the sigma-point approach calculates the posterior covariance (Equation 3.15). The same can be done for the cross-covariance resulting in a form identical to Equation 3.16.

This analysis shows that the WSLR method is *implicitly* used for the purpose of statistical linearization when posterior statistics are calculated using the sigma-point approach. Note that the correction term in Equation 4.24, i.e., $\mathbf{P}_{\mathbf{e}}$, are never explicitly calculated in the sigma-point approach, but its effect is implicitly incorporated through the way that the posterior statistics are approximated. This implied correction term gives useful insight into why the sigma-point approach (and hence SPKFs) tend to generate posterior covariance estimates that are consistent. The EKF on the other hand simply approximates the posterior covariance by $\hat{\mathbf{P}}_{\mathbf{y}}^{lin} = \mathbf{A}_{lin}\mathbf{P}_{\mathbf{x}}\mathbf{A}_{lin}^T$, where \mathbf{A}_{lin} is the analytically calculated Jacobian of $\mathbf{g}(\cdot)$ evaluated at $\mathbf{x} = \bar{\mathbf{x}}$. This often results in overly confident (not consistent) estimates. In this statistical linearization formulation, the matrix \mathbf{A} can be thought of as the *expected* Jacobian of the nonlinear function over the “uncertainty range” of the prior random variable \mathbf{x} . Here the expectation is taken with regard to the prior *Gaussian* approximate density of \mathbf{x} . This expected Jacobian plays an important role when the SPKF is used for recursive nonlinear parameter estimation, with specific regard to generalization and dealing with local minima. This will be discussed further in Section 4.5.

A final interesting insight into the use of the sigma-point can be found by looking at

the general form of the optimal linear Kalman update used in the measurement update step of the SPKF. This is given by,

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}\boldsymbol{\nu}_k , \quad (4.29)$$

where $\hat{\mathbf{x}}_k^-$ is the optimal prediction $\hat{\mathbf{x}}_k^- = E[\mathbf{x}_k | \mathbf{y}_{1:k-1}]$, $\boldsymbol{\nu}_k$ is the innovation (new information) given by $\boldsymbol{\nu}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k^- = \mathbf{y}_k - E[\mathbf{y}_k | \hat{\mathbf{x}}_k^-]$ and \mathbf{K} is the Kalman gain. Equation 4.29 can thus be interpreted as a linear combination of the best prediction of the state and the new information contained in the innovation. In this formulation, the Kalman gain, given by

$$\mathbf{K} = \mathbf{P}_{\mathbf{x}\mathbf{y}} \mathbf{P}_{\mathbf{y}}^{-1} , \quad (4.30)$$

serves the purpose of optimally propagating the new information in the output (observation) space, back down into the hidden state-space before combining it with the predicted state value. If we compare Equation 4.30 with Equation 4.19, we see that the Kalman gain has the same general form as that of the optimal WSLR gain term, \mathbf{A} . The difference being, however, that the Kalman gain equation (Equation 4.30) replaces $\mathbf{P}_{\mathbf{x}\mathbf{y}}^T$ with $\mathbf{P}_{\mathbf{y}\mathbf{x}}^T = \mathbf{P}_{\mathbf{x}\mathbf{y}}$ and $\mathbf{P}_{\mathbf{x}}^{-1}$ with $\mathbf{P}_{\mathbf{y}}^{-1}$. Using the just discussed implicit WSLR view of the sigma-point approach, this can thus be interpreted as propagating the innovation information downwards (from the \mathbf{y} -space to the \mathbf{x} -space) using a statistically linearized inverse observation function.

4.2.4 Demonstration of statistical linearization as implicitly used by the sigma-point approach

Figure 4.1 gives a graphical demonstration of the sigma-point approach (through the implicit use of WSLR) and contrasts its performance with that of the normal first-order Taylor series linearization employed by the EKF. On the bottom x-axis the prior Gaussian random variable (GRV) \mathbf{x} is stylistically indicated by a green Gaussian distribution with mean $\bar{\mathbf{x}}$. This GRV undergoes a transformation through a arbitrary nonlinear function (thick blue curve) in order to produce the posterior random variable \mathbf{y} . The true mean

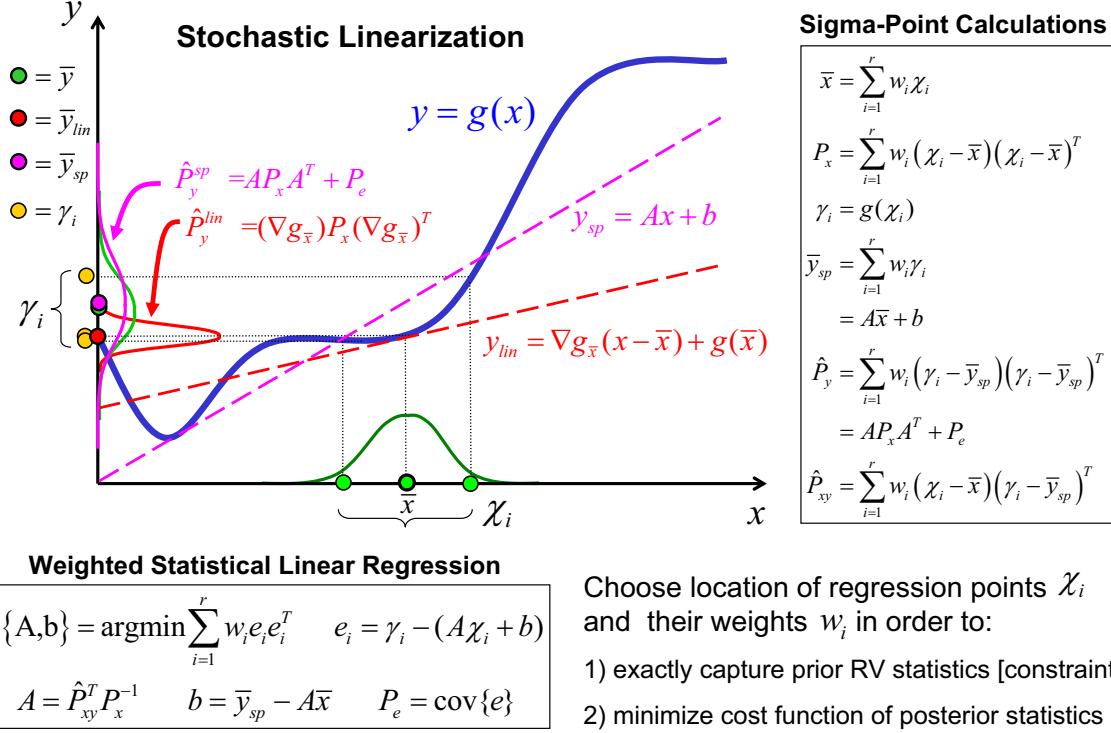


Figure 4.1: Stochastic linearization also known as weighted statistical linear regression. This technique is implicitly employed in all sigma-point Kalman filters.

and covariance² of this posterior RV is indicated by the green Gaussian distribution and dot on the y-axis. If we linearize the nonlinear function through a first-order Taylor series expansion around the prior mean \bar{x} (as done in the EKF), we obtain the dashed red line as a linear approximation. The EKF then analytically propagates the prior mean and covariance through this linearized system resulting in the approximate posterior distribution indicated in red on the y-axis. The resulting mean (red dot) is clearly biased with respect to the true posterior mean (green dot) and the covariance (red curve) is much more peaked (inconsistent) with respect to the true covariance (green curve). Clearly the linearization employed by the EKF results in highly inaccurate results. In contrast, the sigma-point approach is more accurate (and consistent). The prior set of sigma-points (green dots on x-axis) are chosen such that they accurately capture the prior mean and covariance of x .

²Remember, the true posterior distribution will not be Gaussian (due to the nonlinear mapping), but it will still have a mean and covariance. It is these first and second order moments we are trying to capture as accurately as possible.

using a standard sigma-point selection scheme. In this case only three sigma-points are needed (we showed in Chapter 3 how in general $2L+1$ points are needed for GRVs, where L is the dimension of the variable): The first sigma-point lies at the mean, $\mathbf{x}_0 = \bar{x}$, and the other two are symmetrically located around the mean, i.e., $\mathbf{x}_1 = \bar{x} - \sigma_x$ and $\mathbf{x}_2 = \bar{x} + \sigma_x$ where σ_x is the standard deviation of the distribution of x . The sigma-points are now propagated through the true nonlinear function resulting in the posterior sigma-point set, \mathbf{y}_i (yellow dots on y-axis). These posterior points (and their weights) are now used to calculate the approximate posterior mean and covariance (using Equations 4.16 and 4.17). The resulting mean and covariance are indicated in magenta on the y-axis. The implicit statistical linearization (WSLR) is indicated by the dashed magenta linear function. This clearly differs from the normal EKF linearization (red dashed line), not only in gradient but also in offset. The difference in gradient is directly due to the fact that the WSLR linearization calculates an approximate expected Jacobian whereas the EKF simply calculates the Jacobian at the prior mean (gradient of tangent line to nonlinear function evaluated at \bar{x}). The difference in offset is due to the fact that the WSLR linearization employs a non-zero bias-correction term \mathbf{b} , whereas in the EKF linearization, the corresponding bias term, $\nabla \mathbf{g}_{\bar{x}}(\mathbf{x} - \bar{x})$, equates to zeros when evaluated at the linearization point. The superior performance of the sigma-point approach over that of the EKF linearization is clearly evident.

4.3 Accuracy of Sigma-point Approach

We can gain valuable insight into the SPKF family of filters by analyzing the theoretical accuracy of the sigma-point approach calculated posterior statistics. For this purpose we will use the same multi-dimensional Taylor series analysis as in Section 2.3.

Assume a random variable \mathbf{x} with mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_{\mathbf{x}}$ undergoes an arbitrary nonlinear transformation,

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) = \mathbf{g}(\bar{\mathbf{x}} + \boldsymbol{\delta}_{\mathbf{x}}) , \quad (4.31)$$

where $\boldsymbol{\delta}_{\mathbf{x}}$ is a *zero-mean* random variable with the same covariance $\mathbf{P}_{\mathbf{x}}$ as \mathbf{x} . Given this

formulation, we can again expand³ $\mathbf{g}(\mathbf{x})$ as

$$\mathbf{y} = \mathbf{g}(\bar{\mathbf{x}}) + \mathbf{D}_{\delta_{\mathbf{x}}} \mathbf{g} + \frac{1}{2!} \mathbf{D}_{\delta_{\mathbf{x}}}^2 \mathbf{g} + \frac{1}{3!} \mathbf{D}_{\delta_{\mathbf{x}}}^3 \mathbf{g} + \frac{1}{4!} \mathbf{D}_{\delta_{\mathbf{x}}}^4 \mathbf{g} + \dots . \quad (4.32)$$

The SPKF calculates the posterior mean from the propagated sigma-points using Equation 4.16. By substituting Equation 4.32 into 4.16 and using the sigma-points and their weights as given by Equations 3.12 and 3.85 for the UKF and CDKF respectively, we can now calculate the accuracy of the posterior mean:

4.3.1 Posterior mean accuracy

For the UKF, the sigma-points are given by

$$\mathcal{X}_i = \bar{\mathbf{x}} \pm \sqrt{(L + \lambda)} \boldsymbol{\sigma}_{\mathbf{x}_i} \quad (4.33)$$

$$= \bar{\mathbf{x}} \pm \tilde{\boldsymbol{\sigma}}_{\mathbf{x}_i} \quad (4.34)$$

where $\boldsymbol{\sigma}_{\mathbf{x}_i}$ denotes the i th column⁴ of the matrix square-root of $\mathbf{P}_{\mathbf{x}}$. This implies that

$$\sum_{i=1}^L \boldsymbol{\sigma}_{\mathbf{x}_i} \boldsymbol{\sigma}_{\mathbf{x}_i}^T = \mathbf{P}_{\mathbf{x}} . \quad (4.35)$$

Given this formulation of the sigma-points, we can again write the propagation of each point through the nonlinear function as a Taylor series expansion about $\bar{\mathbf{x}}$:

$$\mathcal{Y}_i = \mathbf{g}(\mathcal{X}_i) = \mathbf{g}(\bar{\mathbf{x}}) + \mathbf{D}_{\tilde{\boldsymbol{\sigma}}_{\mathbf{x}_i}} \mathbf{g} + \frac{1}{2!} \mathbf{D}_{\tilde{\boldsymbol{\sigma}}_{\mathbf{x}_i}}^2 \mathbf{g} + \frac{1}{3!} \mathbf{D}_{\tilde{\boldsymbol{\sigma}}_{\mathbf{x}_i}}^3 \mathbf{g} + \frac{1}{4!} \mathbf{D}_{\tilde{\boldsymbol{\sigma}}_{\mathbf{x}_i}}^4 \mathbf{g} + \dots , \quad (4.36)$$

where the total differential operator $\mathbf{D}_{\tilde{\boldsymbol{\sigma}}_{\mathbf{x}_i}}$ is defined the same as in Equation 2.12 (see Chapter 2, Equations 2.12-2.14). The l th term in the multi-dimensional Taylor series expansion (Equation 4.36) is thus given by

$$\mathbf{D}_{\tilde{\boldsymbol{\sigma}}_{\mathbf{x}_i}}^l \mathbf{g} = \frac{1}{l!} \left[\sum_{j=1}^{N_x} \tilde{\boldsymbol{\sigma}}_{\mathbf{x}_{i,j}} \frac{\partial}{\partial x_j} \right]^l \mathbf{g}(\mathbf{x}) \Big|_{\mathbf{x}=\bar{\mathbf{x}}} , \quad (4.37)$$

³See Section 2.3 for definitions of all the relevant operators such as $\mathbf{D}_{\delta_{\mathbf{x}}}^i \mathbf{g}$, etc.

⁴ $\boldsymbol{\sigma}_{\mathbf{x}_i}$ is thus treated as a column vector.

where $\tilde{\sigma}_{\mathbf{x}_{i,j}}$ is the j th component of $\tilde{\sigma}_{\mathbf{x}_i}$, $\frac{\partial}{\partial x_j}$ is the normal partial derivative operator with respect to x_j (the j th component of \mathbf{x}), and $N_{\mathbf{x}}$ is the dimension of \mathbf{x} (i.e. $\mathbf{x} \in \mathbb{R}^{N_{\mathbf{x}}}$).

Using Equations 3.14, 3.13 and 3.12, the UKF calculated posterior mean is given by

$$\bar{\mathbf{y}}_{UKF} = \frac{\lambda}{L + \lambda} \mathbf{g}(\bar{\mathbf{x}}) + \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \left[\mathbf{g}(\bar{\mathbf{x}}) + \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}} \mathbf{g} + \frac{1}{2!} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}}^2 \mathbf{g} + \frac{1}{3!} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}}^3 \mathbf{g} + \dots \right] \quad (4.38)$$

$$= \mathbf{g}(\bar{\mathbf{x}}) + \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \left[\frac{1}{2} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}}^2 \mathbf{g} + \frac{1}{4!} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}}^4 \mathbf{g} + \frac{1}{6!} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}}^6 \mathbf{g} + \dots \right], \quad (4.39)$$

where the simplification in the last line is justified by the fact that the sigma-points are symmetrically distributed around $\bar{\mathbf{x}}$ resulting in zero odd moment terms. Using the same expansion as in Equations 2.18-2.19 (see Chapter 2), we can rewrite the second order term in Equation 4.39 as

$$\frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}}^2 \mathbf{g} = \frac{1}{2(L + \lambda)} \left\{ \sum_{i=1}^{2L} \frac{1}{2} \left[\mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}} \left(\mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}} \mathbf{g} \right)^T \right] \right\} \quad (4.40)$$

$$= \frac{1}{4(L + \lambda)} \left\{ \sum_{i=1}^{2L} \left[(\nabla^T \tilde{\sigma}_{\mathbf{x}_i} \tilde{\sigma}_{\mathbf{x}_i}^T \nabla) \mathbf{g}(\mathbf{x}) \Big|_{\mathbf{x}=\bar{\mathbf{x}}} \right] \right\} \quad (4.41)$$

$$= \frac{1}{4(L + \lambda)} \left\{ \sum_{i=1}^{2L} \left[\left(\nabla^T \sqrt{L + \lambda} \sigma_{\mathbf{x}_i} \sigma_{\mathbf{x}_i}^T \sqrt{L + \lambda} \nabla \right) \mathbf{g}(\mathbf{x}) \Big|_{\mathbf{x}=\bar{\mathbf{x}}} \right] \right\} \quad (4.42)$$

$$= \frac{L + \lambda}{4(L + \lambda)} \left[\nabla^T \left(\sum_{i=1}^{2L} \sigma_{\mathbf{x}_i} \sigma_{\mathbf{x}_i}^T \right) \nabla \right] \mathbf{g}(\mathbf{x}) \Big|_{\mathbf{x}=\bar{\mathbf{x}}} \quad (4.43)$$

$$= \frac{L + \lambda}{4(L + \lambda)} (\nabla^T 2\mathbf{P}_{\mathbf{x}} \nabla) \mathbf{g}(\mathbf{x}) \Big|_{\mathbf{x}=\bar{\mathbf{x}}} \quad (4.44)$$

$$= \frac{1}{2} (\nabla^T \mathbf{P}_{\mathbf{x}} \nabla) \mathbf{g}(\mathbf{x}) \Big|_{\mathbf{x}=\bar{\mathbf{x}}} \quad (4.45)$$

where we made use of the fact that $\sum_{i=1}^{2L} \sigma_{\mathbf{x}_i} \sigma_{\mathbf{x}_i}^T = 2\mathbf{P}_{\mathbf{x}}$ (see Equation 4.35). Substituting this result into Equation 4.39 gives us,

$$\bar{\mathbf{y}}_{UKF} = \mathbf{g}(\bar{\mathbf{x}}) + \frac{1}{2} (\nabla^T \mathbf{P}_{\mathbf{x}} \nabla) \mathbf{g}(\mathbf{x}) \Big|_{\mathbf{x}=\bar{\mathbf{x}}} + \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \left[\frac{1}{4!} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}}^4 \mathbf{g} + \frac{1}{6!} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_i}}^6 \mathbf{g} + \dots \right]. \quad (4.46)$$

When we compare Equations 4.46 and 2.20, we can clearly see that the true posterior mean

and the mean calculated by the UKF agrees exactly to the third order and that errors are only introduced in the fourth and higher-order terms. The magnitude of these errors depends on the choice of the composite scaling parameter λ as well as the higher-order derivatives of the nonlinear function \mathbf{g} . In contrast (see Chapter 2), the EKF linearization approach calculates the posterior mean as, $\bar{\mathbf{y}}_{LIN} = \mathbf{g}(\bar{\mathbf{x}})$, which only agrees with the true posterior mean up to the first order.

By noting that the CDKF weights and sigma-points can be derived from the UKF weights after the following substitution is made,

$$h^2 = L + \lambda ,$$

we can derive exactly the same result for the posterior mean calculated by the CDKF, i.e.,

$$\bar{\mathbf{y}}_{CDKF} = \mathbf{g}(\bar{\mathbf{x}}) + \frac{1}{2} (\nabla^T \mathbf{P}_{\mathbf{x}} \nabla) \mathbf{g}(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}} + \frac{1}{2h^2} \sum_{i=1}^{2L} \left[\frac{1}{4!} \mathbf{D}_{\delta_{\mathbf{x}_i}}^4 \mathbf{g} + \frac{1}{6!} \mathbf{D}_{\delta_{\mathbf{x}_i}}^6 \mathbf{g} + \dots \right] . \quad (4.47)$$

See [147] for the full derivation.

4.3.2 Posterior covariance accuracy

As derived in Section 2.3, the true posterior covariance is given by

$$\begin{aligned} \mathbf{P}_{\mathbf{y}} &= \mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T - \frac{1}{4} E [\mathbf{D}_{\delta_{\mathbf{x}}}^2 \mathbf{g}] E [\mathbf{D}_{\delta_{\mathbf{x}}}^2 \mathbf{g}]^T \\ &\quad + E \left[\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{i!j!} \mathbf{D}_{\delta_{\mathbf{x}}}^i \mathbf{g} (\mathbf{D}_{\delta_{\mathbf{x}}}^j \mathbf{g})^T}_{\forall i,j: \text{ such that } ij>1} \right] \\ &\quad - \left(\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{(2i)!(2j)!} E [\mathbf{D}_{\delta_{\mathbf{x}}}^{2i} \mathbf{g}] E [\mathbf{D}_{\delta_{\mathbf{x}}}^{2j} \mathbf{g}]^T}_{\forall i,j: \text{ such that } ij>1} \right) , \end{aligned} \quad (4.48)$$

which can be further simplified as

$$\begin{aligned}
 \mathbf{P}_y &= \mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T - \frac{1}{4} [\mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T] [\mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T]^T \\
 &\quad + E \left[\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{i!j!} \mathbf{D}_{\delta_{\mathbf{x}}}^i \mathbf{g} (\mathbf{D}_{\delta_{\mathbf{x}}}^j \mathbf{g})^T}_{\forall i,j: \text{ such that } ij>1} \right] \\
 &\quad - \left(\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{(2i)!(2j)!} E [\mathbf{D}_{\delta_{\mathbf{x}}}^{2i} \mathbf{g}] E [\mathbf{D}_{\delta_{\mathbf{x}}}^{2j} \mathbf{g}]^T}_{\forall i,j: \text{ such that } ij>1} \right). \tag{4.49}
 \end{aligned}$$

Using the results from Equations 4.46, 4.47, 3.5 and 3.88, and a similar approach⁵ as the one used above to calculate the accuracy in the mean approximation, we can derive the following expressions for the SPKF calculated posterior covariance:

$$\begin{aligned}
 \mathbf{P}_y^{UKF} &= \mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T - \frac{1}{4} [\mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T] [\mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T]^T \\
 &\quad + \frac{1}{2(L+\lambda)} \sum_{k=1}^{2L} \left[\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{i!j!} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_k}}^i \mathbf{g} (\mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_k}}^j \mathbf{g})^T}_{\forall i,j: \text{ such that } ij>1} \right] \\
 &\quad - \left(\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{4(2i)!(2j)!(L+\lambda)^2} \sum_{k=1}^{2L} \sum_{m=1}^{2L} \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_k}}^{2i} \mathbf{g} (\mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_m}}^{2j} \mathbf{g})^T}_{\forall i,j: \text{ such that } ij>1} \right). \tag{4.50}
 \end{aligned}$$

Comparing Equations 4.50 and 4.49 and noting that $\delta_{\mathbf{x}} = \sigma_{\mathbf{x}}$ in a statistical sense (with regard to the covariance of \mathbf{x}), it is clear that the UKF calculates the posterior covariance accurately in the first two terms, with errors only introduced at the fourth- and higher-order moments. Julier and Uhlmann [99] show how the absolute term-by-term errors of

⁵For brevity's sake the full derivation is not presented here. The interested reader is referred to [95, 147] for a complete derivation of not only the posterior covariance expressions, but also for other terms such as the skew and kurtosis.

these higher-order moments are again consistently smaller for the sigma-point approach than for the linearized (EKF) case that truncates the Taylor series after the first term, that is,

$$\mathbf{P}_y^{LIN} = \mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T.$$

For this derivation, we have assumed the value of the β parameter in the UKF covariance calculation to be zero. If extra knowledge about the shape of the prior distribution of \mathbf{x} is known, β can be set to a non-zero value that minimizes the error in some of the higher (≥ 4) order moments. Julier [96] shows how the error in the kurtosis of the posterior distribution is minimized for a Gaussian \mathbf{x} when $\beta = 2$. Using a similar approach as above and the same substitution for the sigma-points weights as in the mean calculation, we can derive the following expression for the CDKF calculated posterior covariance (see [147] for full derivation):

$$\begin{aligned} \mathbf{P}_y^{CDKF} &= \mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T - \frac{1}{4} [\mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T] [\mathbf{G}_{\bar{\mathbf{x}}} \mathbf{P}_{\mathbf{x}} \mathbf{G}_{\bar{\mathbf{x}}}^T]^T \\ &\quad + \frac{1}{h^2} \left[\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{(2i+1)!(2j+1)!} \sum_{k=1}^L \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_k}}^{2i+1} \mathbf{g} (\mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_k}}^{2j+1} \mathbf{g})^T}_{\forall i,j: \text{ such that } ij > 1} \right] \\ &\quad - \left(\underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{(2i)!(2j)!h^4} \sum_{k=1}^L \mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_k}}^{2i} \mathbf{g} (\mathbf{D}_{\tilde{\sigma}_{\mathbf{x}_k}}^{2j} \mathbf{g})^T}_{\forall i,j: \text{ such that } ij > 1} \right). \end{aligned} \quad (4.51)$$

Clearly, the CDKF has the same exact accuracy in the first and second terms of the posterior covariance as the UKF. In fact, any SPKF based on a valid sigma-point approach implementation will exhibit this accuracy in the posterior covariance and mean as derived above for the UKF and CDKF. As discussed in Section 3.4.1, the differences between the UKF and CDKF implementations of the SPKF covariance approximation, lies in the choice of cross-product terms to discard in the higher-order (> 2) terms of the full Taylor-series expansion. This is reflected in the differences between the last two compound terms of Equation 4.50 (UKF) and the last two compound terms of Equation 4.51 (CDKF).

In [147], Norgaard shows how the CDKF has a slightly smaller absolute error (compared to the UKF) in the fourth order term and also guarantees positive semi-definiteness of the posterior covariance. Notwithstanding this small theoretical advantages with regard to approximation accuracy, we've found in practice that the CDKF and UKF perform equally well with negligible difference in estimation accuracy. Both generate estimates however that are clearly superior to those calculated by an EKF. The performance similarity of the UKF and CDKF is clearly demonstrated on nonlinear time-series estimation problem in Section 3.5.1. However, there is one advantage the CDKF has over the UKF: The CDKF uses only a single scalar scaling parameter, the central difference interval size h , as opposed to the three (α, κ, β) that the UKF uses. Once again this parameter determines the *spread* of the sigma-points around the prior mean. As mentioned earlier, the optimal setting for h is equal to the kurtosis of the prior RV. For Gaussian RVs the optimal value is thus $h = \sqrt{3}$.

4.3.3 Demonstration of accuracy of sigma-point approach

In order to demonstrate the 2nd order accuracy of the sigma-point approach we revisit the scalar analytical example first used in Section 2.5.1 to demonstrate the difference in accuracy between the EKF's linearization approximations and the true analytically calculated posterior statistics. The nonlinear transformation is the simple scalar quadratic function

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

We will now repeat this experiment, but include the sigma-point approach calculated posterior statistics approximations (UKF and CDKF) for comparison. The results are tabulated in Table 4.1. As expected, the UKF and CDKF (using the sigma-point approach) calculates the posterior statistics exactly with the same level of 2nd order accuracy as the analytical approach. Since the nonlinearity in question has no higher order (> 2) derivatives the sigma-point approach will be exact. Notice again the highly inaccurate estimates generated by the EKF's first order linearization approach.

Table 4.1: Mean and covariance of a nonlinear transformed scalar Gaussian random variable. The table compares the true analytically calculated posterior mean and covariance of $y = g(x) = x^2$, i.e., \bar{y} and σ_y^2 , with Monte Carlo (100,000 samples) calculated verifications thereof. The first order linearization approximations (as employed by the EKF) as well as those calculated by the SPKF's sigma-point approach (UKF and CDKF) of these values are also shown. The experiment was repeated three times, the only difference being the covariance of the prior random variable x , i.e., x is a Gaussian random variable with mean $\bar{x} = 1$ and covariance $\sigma_x^2 = \{0.1, 1, 10\}$ for each experiment.

$\bar{x} = 1$	$\sigma_x^2 = 0.1$		$\sigma_x^2 = 1$		$\sigma_x^2 = 10$	
$y = g(x) = x^2$	\bar{y}	σ_y^2	\bar{y}	σ_y^2	\bar{y}	σ_y^2
True statistics (analytical)	1.1	0.42	2	6	11	240
True statistics (Monte Carlo)	1.100	0.419	2.000	5.980	10.99	239.8
Approximated statistics (EKF)	1	0.4	1	4.0	1	40
Approximated statistics (UKF)	1.100	0.420	2.000	6.000	11.00	240.0
Approximated statistics (CDKF)	1.100	0.420	2.000	6.000	11.00	240.0

4.4 Relationship between Sigma-Point Approach and Gaussian Quadrature

Further insight into the sigma-point approach can be gained by relating it to a numerical integral evaluation technique called *Gaussian quadrature* [79, 156] utilizing the *Gauss-Hermite* rule:

In the scalar case, the Gauss-Hermite rule is given by

$$\int_{-\infty}^{\infty} f(x) \frac{1}{\sqrt{2\pi}} e^{-x^2} dx = \sum_{i=1}^m w_i f(x_i), \quad (4.52)$$

where the equality holds for all polynomials, $f(\cdot)$, of degree up to $2m-1$ and the quadrature abscissas, x_i , and weights, w_i , are determined according to the specific rule type. To be more specific: The abscissas for quadrature order n are given by the roots x_i of the Hermite polynomials $H_n(x)$, which occur symmetrically about 0. The weights are given by

$$w_i = -\frac{A_{n+1}\gamma_n}{A_n H'_n(x_i) H_{n+1}(x_i)} = \frac{A_n}{A_{n-1}} \frac{\gamma_{n-1}}{H_{n-1}(x_i) H'_n(x_i)}, \quad (4.53)$$

where $\gamma_n = \sqrt{\pi}2^n n!$ and A_n is the coefficient of x^n in $H_n(x)$. For Hermite polynomials,

$$A_n = 2^n ,$$

such that

$$\frac{A_n}{A_{n-1}} = 2 .$$

Substituting these results back into Equation 4.53 gives us the weights:

$$w_i = -\frac{2^{n+1}n!\sqrt{\pi}}{H_{n+1}(x_i)H'_n(x_i)} \quad (4.54)$$

$$= \frac{2^n(n-1)!\sqrt{\pi}}{H_{n-1}(x_i)H'_n(x_i)} \quad (4.55)$$

$$= \frac{2^{n+1}n!\sqrt{\pi}}{[H'_n(x_i)]^2} \quad (4.56)$$

$$= \frac{2^{n+1}n!\sqrt{\pi}}{[H_{n+1}(x_i)]^2} \quad (4.57)$$

$$= \frac{2^{n-1}n!\sqrt{\pi}}{n^2 [H_{n-1}(x_i)]^2} , \quad (4.58)$$

where Equation 4.56 and 4.57 follow using the *recurrence relation* [205]

$$H'_n(x) = 2nH_{n-1}(x) = 2xH_n(x) - H_{n+1}(x) ,$$

to obtain

$$H'_n(x_i) = 2nH_{n-1}(x_i) = -H_{n+1}(x_i) ,$$

and Equation 4.58 follows from a proof by Abramowitz and Stegun [205]. Due to the complexity of calculating the specific values for the abscissas and weights, published tables of precalculated solutions are often utilized when specific Gauss-Hermite filters are implemented. However, for small n these values can be computed analytically. Table 4.2 lists the values for $n = 2 \dots 4$.

For the scalar case, the UKF with $\alpha = 1$, $\beta = 0$ and $\kappa = 2$ coincides with the three point Gauss-Hermite quadrature rule.

Ito and Xiong [88] recently published an algorithm called the *Gauss-Hermite filter* that

Table 4.2: Gauss-Hermite quadrature abscissas and weights for small n .

n	x_i	w_i
2	$\pm \frac{1}{2}\sqrt{2}$	$\frac{1}{2}\sqrt{\pi}$
3	0	$\frac{2}{3}\sqrt{\pi}$
	$\pm \frac{1}{2}\sqrt{6}$	$\frac{1}{6}\sqrt{\pi}$
4	$\pm \sqrt{\frac{3-\sqrt{6}}{2}}$	$\frac{\sqrt{\pi}}{4(3-\sqrt{6})}$
	$\pm \sqrt{\frac{3+\sqrt{6}}{2}}$	$\frac{\sqrt{\pi}}{4(3+\sqrt{6})}$

utilizes the Gauss-Hermite quadrature method to calculate the recursive Bayesian estimation integrals under a Gaussian assumption. It is for this same purpose that the SPKF uses the sigma-point approach. As we showed above, the Gauss-Hermite rule share a number of similarities with the sigma-point approach (i.e., point wise evaluation of nonlinearities and weighted sum statistics calculation (see Equation 4.52)), with one important distinction that is both its biggest advantage and disadvantage: Whereas the sigma-point approach still *approximates* the Bayesian integrals (albeit much more accurately than the EKF), the Gauss-Hermite rule calculates these integrals *exactly* under a Gaussian assumption. This accuracy comes at an exceedingly hight cost though, namely. $\mathcal{O}(m^L)$, where m is the polynomial order of the nonlinearity and L is the dimension of the system state. Clearly, the Gauss-Hermite filter is only practically implementable (computationally tractable) for low dimensional DSSMs, and hence much more limited in practice than the SPKF (which is $\mathcal{O}(L^3)$ for general DSSMs and $\mathcal{O}(L^2)$ for parameter estimation). For a full exposition of the Gauss-Hermite filter including experimental verification for low dimensional state estimation problems, see [88].

4.5 Theoretical Analysis of SPKF based Parameter Estimation

When used as a parameter estimation method, the EKF can be viewed as an efficient second-order nonlinear programming approach similar to the Gauss-Newton update rule [12, 126]. In this section we will cast the SPKF algorithm for parameter estimation into a

similar framework, showing its particular relationship with other 2nd order optimization methods. This analysis provides further insight into the subtle⁶ differences between the EKF and SPKF for parameter estimation, and gives at least a theoretically underpinned intuitive reason why better global optimization performance can be expected from the SPKF: We hypothesize that the SPKF is less likely to get stuck in local minima of the optimization error surface due to the inherent statistical averaging that takes place in the sigma-point approach, i.e., Jacobians are replaced by stochastically *averaged Jacobians*, etc.

Our analysis is twofold: First we will derive the SPKF based parameter estimation measurement update from a *maximum a-posteriori* (MAP) perspective, utilizing the implicit WSLR form of the nonlinear prediction equation. Using the MAP result, we will then proceed to show the equivalence of the SPKF measurement update (for parameter estimation) to an online stochastic Gauss-Newton method that iteratively approximates the inverse of the empirical Fisher information matrix by the outer product of the stochastically averaged (expected) Jacobian matrices of the posterior likelihood function.

4.5.1 MAP Parameter Estimation

Using the standard state-space model for parameter estimation (Equations 3.129 and 3.130 and Bayes rule, we can write the posterior distribution of the system parameters (\mathbf{w}) conditioned on all of the observations as

$$p(\mathbf{w}_k | \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_{1:k} | \mathbf{w}_k)p(\mathbf{w}_k)}{p(\mathbf{z}_{1:k})}, \quad (4.59)$$

⁶Since the state transition function (process model) for parameter estimation is linear (see Equation 3.129), the EKF and SPKF algorithms differ only in the measurement update.

where we substituted \mathbf{d} (the noisy observation) in Equation 3.130 with \mathbf{z} . This posterior can now be expanded as follows:

$$p(\mathbf{w}_k | \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k, \mathbf{z}_{1:k-1} | \mathbf{w}_k)p(\mathbf{w}_k)}{p(\mathbf{z}_{1:k})} \quad (4.60)$$

$$= \frac{p(\mathbf{z}_k | \mathbf{z}_{1:k-1}, \mathbf{w}_k)p(\mathbf{z}_{1:k-1} | \mathbf{w}_k)p(\mathbf{w}_k)}{p(\mathbf{z}_{1:k})} \quad (4.61)$$

$$= \frac{p(\mathbf{z}_k | \mathbf{w}_k)p(\mathbf{z}_{1:k-1} | \mathbf{w}_k)p(\mathbf{w}_k)p(\mathbf{z}_{1:k-1})}{p(\mathbf{z}_{1:k})p(\mathbf{z}_{1:k-1})} \quad (4.62)$$

$$= \frac{p(\mathbf{z}_k | \mathbf{w}_k)p(\mathbf{w}_k | \mathbf{z}_{1:k-1})}{p(\mathbf{z}_{1:k})/p(\mathbf{z}_{1:k-1})}, \quad (4.63)$$

where we made use of the conditional independence of the observation given the current state [$p(\mathbf{z}_k | \mathbf{z}_{1:k-1}, \mathbf{w}_k) \doteq p(\mathbf{z}_k | \mathbf{w}_k)$] going from Equation 4.61 to 4.62, and applied Bayes rule going from Equation 4.62 to 4.63 after multiplying above and below by $p(\mathbf{z}_{1:k-1})$.

The MAP estimate of the parameters $\hat{\mathbf{w}}_k^{MAP}$ can now be found by choosing the \mathbf{w}_k that maximizes the posterior density. Since the denominator of Equation 4.63 is not a function of \mathbf{w} , maximizing the posterior $p(\mathbf{w}_k | \mathbf{z}_{1:k})$ with respect to \mathbf{w}_k is equivalent to maximizing the numerator of Equation 4.63 with respect to \mathbf{w}_k . So, our MAP estimate is given by:

$$\hat{\mathbf{w}}_k^{MAP} = \arg \max_{\mathbf{w}_k} [p(\mathbf{z}_k | \mathbf{w}_k)p(\mathbf{w}_k | \mathbf{z}_{1:k-1})]. \quad (4.64)$$

It is often more convenient to write the MAP estimate as the value of \mathbf{w}_k that *minimizes* the negative of the *log* of the right hand side of Equation 4.64, i.e.,

$$\begin{aligned} \hat{\mathbf{w}}_k^{MAP} &= \arg \min_{\mathbf{w}_k} [-\ln(p(\mathbf{z}_k | \mathbf{w}_k)p(\mathbf{w}_k | \mathbf{z}_{1:k-1}))] \\ &= \arg \min_{\mathbf{w}_k} [-\ln(p(\mathbf{z}_k | \mathbf{w}_k)) - \ln(p(\mathbf{w}_k | \mathbf{z}_{1:k-1}))] \\ &= \arg \min_{\mathbf{w}_k} [J(\mathbf{w}_k)], \end{aligned} \quad (4.65)$$

where

$$J(\mathbf{w}_k) = -\ln(p(\mathbf{z}_k | \mathbf{w}_k)) - \ln(p(\mathbf{w}_k | \mathbf{z}_{1:k-1})), \quad (4.66)$$

is called the *posterior log-likelihood function*.

Next we need to determine what the actual probability density functions $p(\mathbf{z}_k | \mathbf{w}_k)$ and $p(\mathbf{w}_k | \mathbf{z}_{1:k-1})$ look like. In the Kalman MAP framework it is assumed that all densities

are Gaussian (see Appendix A). Given this, we can write the prior density of the system parameters as

$$p(\mathbf{w}_k | \mathbf{z}_{1:k-1}) = \frac{1}{\sqrt{(2\pi)^{L_w} |\mathbf{P}_{\mathbf{w}_k}^-|}} \exp \left[-\frac{1}{2} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-)^T (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) \right], \quad (4.67)$$

where $\hat{\mathbf{w}}_k^-$ is the prior estimate of the system parameters (i.e. before the information in the new observation is incorporated), $\mathbf{P}_{\mathbf{w}_k}^-$ is its covariance and L_w is the dimension of the parameter state vector. In a similar fashion the observation likelihood is given by

$$p(\mathbf{z}_k | \mathbf{w}_k) = \frac{1}{\sqrt{(2\pi)^{L_z} |\mathbf{R}_e|}} \exp \left[-\frac{1}{2} (\mathbf{z}_k - \mathbf{g}(\mathbf{w}_k))^T (\mathbf{R}_e)^{-1} (\mathbf{z}_k - \mathbf{g}(\mathbf{w}_k)) \right], \quad (4.68)$$

where $\mathbf{g}(\cdot)$ is the nonlinear observation function, \mathbf{R}_e is the observation noise covariance⁷ and L_z is the dimension of the observation vector.

Next we must reformulate⁸ the nonlinear observation model,

$$\mathbf{z}_k = \mathbf{g}(\mathbf{w}_k) + \mathbf{e}_k, \quad (4.69)$$

($\mathbf{e}_k \sim \mathcal{N}(\mathbf{e}; \mathbf{0}, \mathbf{R}_e)$ is the zero-mean Gaussian observation noise) into a *statistically linearized* form using weighted statistical linear regression (WSLR) as employed by the sigma-point approach (see Section 4.2). By doing this we implicitly assume that the statistical linearization of $\mathbf{g}(\cdot)$ over the probabilistic spread (uncertainty) of the underlying parameter (state) random variable \mathbf{w} , is a good approximation, i.e., errors are relatively small and normally distributed). This condition can be met if we make sure that the sigma-point set is correctly constructed, i.e., that they capture needed prior statistics of \mathbf{w} . Once met, the resulting statistically linearized approximation of $\mathbf{g}(\cdot)$ will be more robust (and valid) than the *first order analytically linearized* (in a single point) approximation employed by the EKF. The EKF linearization has a much stricter (and hence harder to satisfy) assumption that the current estimate $\hat{\mathbf{w}}_k^-$, is close enough to the true value (realization) of \mathbf{w} such that the expansion is valid. Furthermore (as discussed in length in Sections 2.4 and 4.2) the EKF linearization disregards the fact that the prior estimate of the parameters $\hat{\mathbf{w}}_k^-$ is a random variable with inherent uncertainty which will further influence the validity of the

⁷We assume the observation noise is zero mean.

⁸This reformulation is needed to allow for a tractable analysis.

first-order truncated Taylor series linearization.

In order to reformulate Equation 4.69 into a statistically linearized form, we write down the WSLR approximation of the nonlinear observation function $\mathbf{g}(\mathbf{w}_k)$ using Equations 4.22:

$$\begin{aligned}\mathbf{g}(\mathbf{w}_k) &= \mathbf{A}_k \mathbf{w}_k + \mathbf{b}_k + \boldsymbol{\varepsilon}_k \\ &= \mathbf{A}_k \mathbf{w}_k + \hat{\mathbf{z}}_k^- - \mathbf{A}_k \hat{\mathbf{w}}_k^- + \boldsymbol{\varepsilon}_k \\ &= \mathbf{A} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) + \hat{\mathbf{z}}_k^- + \boldsymbol{\varepsilon}_k ,\end{aligned}\quad (4.70)$$

where \mathbf{A} and \mathbf{b} is given by Equations 4.19 and 4.20. $\boldsymbol{\varepsilon}_k$ is the statistical linearization error (as defined in Section 4.2.2) which is modeled as a zero mean Gaussian random variable with covariance $\mathbf{P}_{\mathbf{e}}$, i.e., $\boldsymbol{\varepsilon}_k \sim \mathcal{N}(\boldsymbol{\varepsilon}; \mathbf{0}, \mathbf{P}_{\mathbf{e}})$. This covariance matrix is identical to $\mathbf{P}_{\boldsymbol{\varepsilon}}$ in Equation 4.21. $\hat{\mathbf{z}}_k^-$ is the observation prediction as calculated by the sigma-point approach (see Equation 4.16). Substituting Equation 4.70 into 4.69 and reordering terms gives us

$$\begin{aligned}\mathbf{z}_k &= \mathbf{A} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) + \hat{\mathbf{z}}_k^- + \boldsymbol{\varepsilon}_k + \mathbf{e}_k \\ &= \mathbf{A} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) + \hat{\mathbf{z}}_k^- + \tilde{\mathbf{e}}_k ,\end{aligned}\quad (4.71)$$

where $\tilde{\mathbf{e}}_k = \boldsymbol{\varepsilon}_k + \mathbf{e}_k$ (the sum of the stochastic linearization error and the true observation noise) is termed the “*effective observation noise*” which is modeled as zero-mean Gaussian random variable with covariance

$$\mathbf{R}_{\tilde{\mathbf{e}}} = \mathbf{P}_{\boldsymbol{\varepsilon}} + \mathbf{R}_{\mathbf{e}} ,\quad (4.72)$$

i.e., $\tilde{\mathbf{e}}_k \sim \mathcal{N}(\tilde{\mathbf{e}}; \mathbf{0}, \mathbf{R}_{\tilde{\mathbf{e}}})$. Here we make the common assumption that $\boldsymbol{\varepsilon}_k$ and \mathbf{e}_k are uncorrelated and independent, allowing the additive form of the effective covariance. This alternative form of the observation model (Equation 4.71) allows us to rewrite the observation likelihood density function (Equation 4.68) as

$$p(\mathbf{z}_k | \mathbf{w}_k) = \frac{1}{\sqrt{2\pi L_{\mathbf{z}} |\mathbf{R}_{\tilde{\mathbf{e}}}|}} \exp \left[-\frac{1}{2} (\mathbf{z}_k - \mathbf{A} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-)^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{z}_k - \mathbf{A} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-) \right] .\quad (4.73)$$

We can now rewrite the posterior log-likelihood function by substituting Equations 4.67 and 4.73 into Equation 4.66:

$$\begin{aligned} J(\mathbf{w}_k) = & \frac{1}{2} [\mathbf{z}_k - \mathbf{A} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-]^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} [\mathbf{z}_k - \mathbf{A} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-] + \\ & + \frac{1}{2} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-)^T (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) . \end{aligned} \quad (4.74)$$

The MAP estimate is now found by substituting Equation 4.74 back into Equation 4.65 and solving for the minimum value of \mathbf{w}_k . The differential condition

$$\frac{\partial}{\partial \mathbf{w}_k} J(\hat{\mathbf{w}}_k^{MAP}) = 0 , \quad (4.75)$$

necessary for an extremum, is called the *maximum posterior likelihood equation* [12] or *score function* [132] for \mathbf{w}_k . For the case at hand, it is calculated by setting the derivative of Equation 4.74 (with respect to \mathbf{w}_k) equal to zero, i.e.,

$$(\mathbf{P}_{\mathbf{w}_k}^-)^{-1} (\hat{\mathbf{w}}_k^{MAP} - \hat{\mathbf{w}}_k^-) - \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} [\mathbf{z}_k - \mathbf{A} (\hat{\mathbf{w}}_k^{MAP} - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-] = 0 . \quad (4.76)$$

We now solve this equation for $\hat{\mathbf{w}}_k^{MAP}$:

$$\begin{aligned} \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} [\mathbf{z}_k - \mathbf{A} (\hat{\mathbf{w}}_k^{MAP} - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-] &= (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} (\hat{\mathbf{w}}_k^{MAP} - \hat{\mathbf{w}}_k^-) \\ \left[(\mathbf{P}_{\mathbf{w}_k}^-)^{-1} + \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} \right] (\hat{\mathbf{w}}_k^{MAP} - \hat{\mathbf{w}}_k^-) &= \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^-) \\ \hat{\mathbf{w}}_k^{MAP} - \hat{\mathbf{w}}_k^- &= \left[(\mathbf{P}_{\mathbf{w}_k}^-)^{-1} + \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} \right]^{-1} \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^-) \\ \hat{\mathbf{w}}_k^{MAP} &= \hat{\mathbf{w}}_k^- + \left[(\mathbf{P}_{\mathbf{w}_k}^-)^{-1} + \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} \right]^{-1} \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^-) \end{aligned} \quad (4.77)$$

which is the maximum a posteriori estimate of \mathbf{w}_k . Making the following substitution,

$$\check{\mathbf{P}} \doteq \left[(\mathbf{P}_{\mathbf{w}_k}^-)^{-1} + \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} \right]^{-1} , \quad (4.78)$$

we can rewrite the MAP estimate of the parameter vector as

$$\hat{\mathbf{w}}_k^{MAP} = \hat{\mathbf{w}}_k^- + \check{\mathbf{P}} \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^-) . \quad (4.79)$$

Using the definitions of \mathbf{A} (Equation 4.19), $\mathbf{R}_{\tilde{\mathbf{e}}}$ (Equation 4.72) and $\mathbf{P}_{\mathbf{e}}$ (Equation 4.21,

we can rewrite the first part of the second term of Equation 4.79 as

$$\begin{aligned}
\check{\mathbf{P}} \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} &= \check{\mathbf{P}} \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\tilde{\mathbf{e}}}) (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\tilde{\mathbf{e}}})^{-1} \\
&= \check{\mathbf{P}} (\mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{R}_{\tilde{\mathbf{e}}}) (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\tilde{\mathbf{e}}})^{-1} \\
&= \check{\mathbf{P}} (\mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{A}^T) (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\tilde{\mathbf{e}}})^{-1} \\
&= \check{\mathbf{P}} \left[\mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} \right] \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\tilde{\mathbf{e}}})^{-1} \\
&= \check{\mathbf{P}} \check{\mathbf{P}}^{-1} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\tilde{\mathbf{e}}})^{-1} \\
&= \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\tilde{\mathbf{e}}})^{-1} \\
&= \mathbf{P}_{\mathbf{w}_k}^- \left[\mathbf{P}_{\mathbf{w}_k \mathbf{z}_k}^T (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} \right]^T (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\tilde{\mathbf{e}}})^{-1} \\
&= \mathbf{P}_{\mathbf{w}_k \mathbf{z}_k} (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\mathbf{e}} + \mathbf{P}_{\boldsymbol{\epsilon}})^{-1} \\
&= \mathbf{P}_{\mathbf{w}_k \mathbf{z}_k} (\mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{R}_{\mathbf{e}} + \mathbf{P}_{\mathbf{z}} - \mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T)^{-1} \\
&= \mathbf{P}_{\mathbf{w}_k \mathbf{z}_k} (\mathbf{R}_{\mathbf{e}} + \mathbf{P}_{\mathbf{z}})^{-1} . \tag{4.80}
\end{aligned}$$

Equation 4.80 is clearly recognizable as the SPKF calculated Kalman gain, i.e.

$$\mathbf{K} = \mathbf{P}_{\mathbf{w}_k \mathbf{z}_k} (\mathbf{R}_{\mathbf{e}} + \mathbf{P}_{\mathbf{z}})^{-1} . \tag{4.81}$$

When this result is substituted back into Equation 4.79, we obtain

$$\hat{\mathbf{w}}_k^{MAP} = \hat{\mathbf{w}}_k^- + \mathbf{K} (\mathbf{z}_k - \hat{\mathbf{z}}_k^-) , \tag{4.82}$$

which is the standard SPKF measurement update of the state. *This implies (and proves) that the SPKF parameter estimation algorithm is equivalent to a maximum posterior likelihood estimate of the underlying parameters under a Gaussian posterior (and noise distribution) assumption.* Furthermore, through equivalence, the covariance of the MAP estimate

will be the same as that calculated by the SPKF⁹, i.e.,

$$\mathbf{P}_{\mathbf{w}_k} = \mathbf{P}_{\mathbf{w}_k}^- - \mathbf{K} (\mathbf{R}_e + \mathbf{P}_z^-) \mathbf{K}^T . \quad (4.83)$$

Another important result we need to derive is the interpretation of the compound term $\check{\mathbf{P}}$ defined in Equation 4.78 and used in Equation 4.79. Using the *matrix inversion lemma*¹⁰ and the definitions of \mathbf{A} (Equation 4.19), $\mathbf{R}_{\tilde{e}}$ (Equation 4.72) and \mathbf{P}_e (Equation 4.21), we can rewrite $\check{\mathbf{P}}$ as

$$\begin{aligned} \check{\mathbf{P}} &= \left[(\mathbf{P}_{\mathbf{w}_k}^-)^{-1} + \mathbf{A}^T (\mathbf{R}_e + \mathbf{P}_\epsilon)^{-1} \mathbf{A} \right]^{-1} \\ &= \mathbf{P}_{\mathbf{w}_k}^- - \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T (\mathbf{R}_e + \mathbf{P}_\epsilon + \mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T)^{-1} \mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \\ &= \mathbf{P}_{\mathbf{w}_k}^- - \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T (\mathbf{R}_e + \mathbf{P}_z^- - \mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T + \mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T)^{-1} \mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \\ &= \mathbf{P}_{\mathbf{w}_k}^- - \mathbf{P}_{\mathbf{w}_k}^- \mathbf{A}^T (\mathbf{R}_e + \mathbf{P}_z^-)^{-1} \mathbf{A} \mathbf{P}_{\mathbf{w}_k}^- \\ &= \mathbf{P}_{\mathbf{w}_k}^- - \mathbf{P}_{\mathbf{w}_k}^- \left[\mathbf{P}_{\mathbf{w}_k \mathbf{z}_k}^T (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} \right]^T (\mathbf{R}_e + \mathbf{P}_z^-)^{-1} \left[\mathbf{P}_{\mathbf{w}_k \mathbf{z}_k}^T (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} \right] \mathbf{P}_{\mathbf{w}_k}^- \\ &= \mathbf{P}_{\mathbf{w}_k}^- - \mathbf{P}_{\mathbf{w}_k \mathbf{z}_k}^- (\mathbf{R}_e + \mathbf{P}_z^-)^{-1} \mathbf{P}_{\mathbf{w}_k \mathbf{z}_k}^T \\ &= \mathbf{P}_{\mathbf{w}_k}^- - \mathbf{P}_{\mathbf{w}_k \mathbf{z}_k}^- (\mathbf{R}_e + \mathbf{P}_z^-)^{-1} (\mathbf{R}_e + \mathbf{P}_z^-) (\mathbf{R}_e + \mathbf{P}_z^-)^{-1} \mathbf{P}_{\mathbf{w}_k \mathbf{z}_k}^T \\ &= \mathbf{P}_{\mathbf{w}_k}^- - \mathbf{K} (\mathbf{R}_e + \mathbf{P}_z^-) \mathbf{K}^T , \end{aligned} \quad (4.84)$$

where we made the standard substitution for the Kalman gain in the last line, i.e. $\mathbf{K} = \mathbf{P}_{\mathbf{w}_k \mathbf{z}_k}^- (\mathbf{R}_e + \mathbf{P}_z^-)^{-1}$. The right hand side of Equation 4.84 is clearly again the standard SPKF measurement update of the parameter (state) covariance (see Equations 3.146 and 3.157). This implies by definition that $\check{\mathbf{P}}$ is the updated parameter (state) covariance, i.e.,

$$\check{\mathbf{P}} = \mathbf{P}_{\mathbf{w}_k} = \mathbf{P}_{\mathbf{w}_k}^- - \mathbf{K} (\mathbf{R}_e + \mathbf{P}_z^-) \mathbf{K}^T . \quad (4.85)$$

This equivalence for $\check{\mathbf{P}}$ will be used in Section 4.5.2. Furthermore, substituting these

⁹This can also be derived (and proved) explicitly (in the same manner as for the state update), by substituting Equation 4.77 into the definition of the MAP estimate covariance, $\mathbf{P}_{\mathbf{w}_k}^{MAP} = E \left[(\mathbf{w}_k - \hat{\mathbf{w}}_k^{MAP}) (\mathbf{w}_k - \hat{\mathbf{w}}_k^{MAP})^T \right]$, and carrying out a few pages more pages of algebraic manipulation (with ample use of the matrix inversion lemma).

¹⁰Matrix inversion lemma [156]: $[\mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T]^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{B}$, where \mathbf{B} is a square symmetric positive-definite (and hence invertible) matrix.

results back into Equation 4.77 (and simplifying) will again result in the standard SPKF parameter estimation measurement update of the state. This can be used as an alternative method to prove the equivalence of the MAP estimate to the SPKF estimate.

4.5.2 The Gauss-Newton Method for Nonlinear Least Squares

The *Gauss-Newton method* [126, 161, 132, 133] is used to find the iterative solution to a nonlinear least-squares problem of the form:

$$\min_{\mathbf{w}} \quad f(\mathbf{w}; \mathbf{z}) = \frac{1}{2} \|\mathbf{r}(\mathbf{w}; \mathbf{z})\|^2 , \quad (4.86)$$

where $r : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is twice differentiable, $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}}$ is the usual norm on $\mathbf{x} \in \mathbb{R}^m$, \mathbf{z} is the observed (noisy) data and \mathbf{w} is the vector of parameters. The Gauss-Newton method (GNM) for solving this problem applies a modified Newton's approach to the problem of finding a root of the gradient function

$$\mathbf{s}(\mathbf{w}; \mathbf{z}) = \nabla_{\mathbf{w}} f(\mathbf{w}; \mathbf{z}) = \mathbf{r}'(\mathbf{w}; \mathbf{z})^T \mathbf{r}(\mathbf{w}; \mathbf{z}) , \quad (4.87)$$

in order to solve

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} f(\mathbf{w}; \mathbf{z}) . \quad (4.88)$$

Modified Newton method iterations are given by

$$\hat{\mathbf{w}}_{i+1} = \hat{\mathbf{w}}_i + \mathbf{C}^{-1} \mathbf{s}(\mathbf{w}; \mathbf{z}) \Big|_{\mathbf{w}=\hat{\mathbf{w}}_i} , \quad (4.89)$$

where \mathbf{C} is some form of tractable approximation of the *observed information matrix* $\mathbf{I}(\mathbf{w})$ evaluated at $\mathbf{w} = \hat{\mathbf{w}}_i$ [132, 133]. When \mathbf{C} is set to

$$\begin{aligned} \mathbf{C} &= \mathbf{I}(\mathbf{w}; \mathbf{z}) \\ &= -\frac{\partial^2 f(\mathbf{w}; \mathbf{z})}{\partial \mathbf{w} \partial \mathbf{w}^T} , \end{aligned} \quad (4.90)$$

the true Hessian of the cost function evaluated at the maximum likelihood estimate of $\hat{\mathbf{w}}^{ML}$, the modified Newton method is equivalent to the Newton-Raphson method. However, when \mathbf{C} is set to the expected information matrix, also known as the *Fisher information matrix*,

i.e.,

$$\mathbf{C} = \mathcal{I}(\mathbf{w}) \quad (4.91)$$

$$= E[\mathbf{I}(\mathbf{w}; \mathbf{z})] \quad (4.92)$$

$$= E\left[-\frac{\partial^2 f(\mathbf{w}; \mathbf{z})}{\partial \mathbf{w} \partial \mathbf{w}^T}\right] \quad (4.93)$$

$$= E\left[-\frac{\partial f(\mathbf{w}; \mathbf{z})}{\partial \mathbf{w}} \left(\frac{\partial f(\mathbf{w}; \mathbf{z})}{\partial \mathbf{w}}\right)^T\right] \quad (4.94)$$

$$= E[\mathbf{s}(\mathbf{w}; \mathbf{z})\mathbf{s}(\mathbf{w}; \mathbf{z})^T] \quad (4.95)$$

(where the expectation is taken over all the data \mathbf{z} , and $\mathcal{I}(\mathbf{w})$ is evaluated at $\mathbf{w} = \hat{\mathbf{w}}_i$), the modified Newton method is known as *Fisher-scoring*. It is a well known fact that the Fisher information matrix evaluated at the maximum likelihood estimate is equal to the inverse of the Cramer-Rao lower bound of the estimate variance [104].

For IID data, we can apply the method of scoring (Fisher scoring [150, 50]) but instead of using $\mathcal{I}(\mathbf{w})$, we can employ the *empirical information matrix*, $\hat{\mathbf{I}}(\mathbf{w}; \mathbf{z})$, evaluated at the current estimate $\mathbf{w} = \hat{\mathbf{w}}_i$. In a seminal paper by Meilijson [133], it is shown how (for IID data) the empirical information matrix is given by

$$\hat{\mathbf{I}}(\mathbf{w}; \mathbf{z}) = \frac{1}{N} \sum_{k=1}^N \mathbf{s}(\mathbf{w}; \mathbf{z}_k)\mathbf{s}(\mathbf{w}; \mathbf{z}_k)^T - \left[\frac{1}{N} \sum_{k=1}^N \mathbf{s}(\mathbf{w}; \mathbf{z}_k) \right] \left[\frac{1}{N} \sum_{k=1}^N \mathbf{s}(\mathbf{w}; \mathbf{z}_k) \right]^T, \quad (4.96)$$

where the sums are taken over all N of the observed data points, is an consistent estimator of $\mathcal{I}(\mathbf{w})$. Meilijson then goes further showing how using the empirical information matrix for Fisher scoring is *equivalent* to the Gauss-Newton method for solving the nonlinear least-squares problem for IID data. For a complete (although lengthy) proof of this, see [133].

The Gauss-Newton method uses the following form for the \mathbf{C} matrix [12],

$$\mathbf{C} = -\mathbf{r}'(\mathbf{w}; \mathbf{z})^T \mathbf{r}'(\mathbf{w}; \mathbf{z}), \quad (4.97)$$

resulting in the following iterative formula for solving the nonlinear least squares cost

function:

$$\hat{\mathbf{w}}_{i+1} = \hat{\mathbf{w}}_i - [\mathbf{r}'(\mathbf{w}; \mathbf{z})^T \mathbf{r}'(\mathbf{w}; \mathbf{z})]^{-1} \mathbf{r}'(\mathbf{w}; \mathbf{z})^T \mathbf{r}(\mathbf{w}; \mathbf{z}) \Big|_{\mathbf{w}=\hat{\mathbf{w}}_i}. \quad (4.98)$$

Note that the traditional Gauss-Newton method for solving the nonlinear least-squares problem (as presented in Equation 4.98) is a *batch* algorithm. In other words, all of the observed data is first collected (for $k = 1 \dots N$) and then used as a complete batch in the GN iterations in order to iteratively find the best (maximum likelihood) estimate of \mathbf{w} . An *online version* of the Gauss-Newton method is also possible if the gradient, $\mathbf{r}'(\mathbf{w}; \mathbf{z})^T \mathbf{r}(\mathbf{w}; \mathbf{z})$, and the inverse of the empirical information matrix, $(\mathbf{r}'(\mathbf{w}; \mathbf{z}) \mathbf{r}'(\mathbf{w}; \mathbf{z})^T)^{-1}$, can be built up recursively as new observation data arrive online. We will now show how the SPKF achieves this by making use of the results of Section 4.5.1.

4.5.3 The SPKF Parameter Estimation Measurement Update as an Online Gauss-Newton Method

First we define the cost-functional $\mathbf{r}(\mathbf{w}_k)$ as

$$\mathbf{r}(\mathbf{w}_k) \doteq \begin{bmatrix} \mathbf{S}_{\tilde{\mathbf{e}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{\mathbf{w}_k}^- \end{bmatrix} \begin{bmatrix} \mathbf{z}_k - \mathbf{A}(\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^- \\ \mathbf{w}_k - \hat{\mathbf{w}}_k^- \end{bmatrix}, \quad (4.99)$$

where $\mathbf{S}_{\tilde{\mathbf{e}}}$ and $\mathbf{S}_{\mathbf{w}_k}^-$ are the normal matrix square-roots of $\mathbf{R}_{\tilde{\mathbf{e}}}^{-1}$ and $(\mathbf{P}_{\mathbf{w}_k}^-)^{-1}$ respectively, such that $\mathbf{S}_{\tilde{\mathbf{e}}}^T \mathbf{S}_{\tilde{\mathbf{e}}} = \mathbf{R}_{\tilde{\mathbf{e}}}^{-1}$ and $(\mathbf{S}_{\mathbf{w}_k}^-)^T \mathbf{S}_{\mathbf{w}_k}^- = (\mathbf{P}_{\mathbf{w}_k}^-)^{-1}$. The variables \mathbf{z}_k , \mathbf{A} , \mathbf{w}_k , $\hat{\mathbf{w}}_k^-$, $\hat{\mathbf{z}}_k^-$, $\mathbf{R}_{\tilde{\mathbf{e}}}$ and $\mathbf{P}_{\mathbf{w}_k}^-$ are used as defined in Section 4.5.1. Substituting this definition of $\mathbf{r}(\mathbf{w}_k)$ into Equation 4.86 gives us the following nonlinear least-squares cost function,

$$\begin{aligned} f(\mathbf{w}_k) &= \frac{1}{2} \mathbf{r}(\mathbf{w}_k)^T \mathbf{r}(\mathbf{w}_k) \\ &= \frac{1}{2} \left\{ [\mathbf{z}_k - \mathbf{A}(\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-]^T (\mathbf{S}_{\tilde{\mathbf{e}}}^T + (\mathbf{w}_k - \hat{\mathbf{w}}_k^-)^T (\mathbf{S}_{\mathbf{w}_k}^-)^T) \right\} \times \\ &\quad \times \{ \mathbf{S}_{\tilde{\mathbf{e}}} [\mathbf{z}_k - \mathbf{A}(\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-] + \mathbf{S}_{\mathbf{w}_k}^- (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) \} \\ &= \frac{1}{2} [\mathbf{z}_k - \mathbf{A}(\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-]^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} [\mathbf{z}_k - \mathbf{A}(\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-] + \\ &\quad + \frac{1}{2} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-)^T (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-), \end{aligned} \quad (4.100)$$

which is identical to Equation 4.74, the MAP log-likelihood cost function. In other words, the nonlinear least squares problem as defined above and the maximum posterior likelihood problem of the previous section optimizes the same objective (cost) function.

Given the definition of $\mathbf{r}(\mathbf{w}_k)$ in Equation 4.99, we next calculate its gradient function $\mathbf{s}(\mathbf{w}; \mathbf{z})$ (using Equation 4.87) as

$$\begin{aligned}
\mathbf{s}(\mathbf{w}; \mathbf{z}) &= \mathbf{r}'(\mathbf{w}; \mathbf{z})^T \mathbf{r}(\mathbf{w}; \mathbf{z}) \\
&= \left\{ \begin{bmatrix} \mathbf{S}_{\tilde{\mathbf{e}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{\mathbf{w}_k}^- \end{bmatrix} \begin{bmatrix} -\mathbf{A} \\ 1 \end{bmatrix} \right\}^T \begin{bmatrix} \mathbf{S}_{\tilde{\mathbf{e}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{\mathbf{w}_k}^- \end{bmatrix} \begin{bmatrix} \mathbf{z}_k - \mathbf{A}(\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^- \\ \mathbf{w}_k - \hat{\mathbf{w}}_k^- \end{bmatrix} \\
&= \begin{bmatrix} -\mathbf{S}_{\tilde{\mathbf{e}}}\mathbf{A} \\ \mathbf{S}_{\mathbf{w}_k}^- \end{bmatrix}^T \begin{bmatrix} \mathbf{S}_{\tilde{\mathbf{e}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{\mathbf{w}_k}^- \end{bmatrix} \begin{bmatrix} \mathbf{z}_k - \mathbf{A}(\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^- \\ \mathbf{w}_k - \hat{\mathbf{w}}_k^- \end{bmatrix} \\
&= -\mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} [\mathbf{z}_k - \mathbf{A}(\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \hat{\mathbf{z}}_k^-] + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) \\
&= \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^-) \\
&= [\mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1}] (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^-), \tag{4.101}
\end{aligned}$$

and the empirical information matrix as

$$\begin{aligned}
\mathbf{r}'(\mathbf{w}; \mathbf{z})^T \mathbf{r}'(\mathbf{w}; \mathbf{z}) &= \begin{bmatrix} -\mathbf{S}_{\tilde{\mathbf{e}}}\mathbf{A} \\ \mathbf{S}_{\mathbf{w}_k}^- \end{bmatrix}^T \begin{bmatrix} -\mathbf{S}_{\tilde{\mathbf{e}}}\mathbf{A} \\ \mathbf{S}_{\mathbf{w}_k}^- \end{bmatrix} \\
&= \mathbf{A}^T \mathbf{S}_{\tilde{\mathbf{e}}}^T \mathbf{S}_{\tilde{\mathbf{e}}} \mathbf{A} + (\mathbf{S}_{\mathbf{w}_k}^-)^T (\mathbf{S}_{\mathbf{w}_k}^-) \\
&= \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1}. \tag{4.102}
\end{aligned}$$

Substituting Equations 4.101 and 4.102 back into Equation 4.98, and noting that $\hat{\mathbf{w}}_k^- \doteq \hat{\mathbf{w}}_i$ and $\hat{\mathbf{w}}_k \doteq \hat{\mathbf{w}}_{i+1}$, we arrive at the following Gauss-Newton update:

$$\begin{aligned}
\hat{\mathbf{w}}_k &= \hat{\mathbf{w}}_k^- - [\mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1}]^{-1} [\mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1}] (\mathbf{w} - \hat{\mathbf{w}}_k^-) + \\
&\quad + [\mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1}]^{-1} \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^-) \Big|_{\mathbf{w}=\hat{\mathbf{w}}_i=\hat{\mathbf{w}}_k^-} \\
&= \hat{\mathbf{w}}_k^- + [\mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1}]^{-1} \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^-), \tag{4.103}
\end{aligned}$$

which is exactly the same as Equation 4.77, the MAP update. *Since we already proved in*

the previous section that the MAP update is equivalent to the SPKF measurement update for parameter estimation, by implication the SPKF update is also equivalent to the Gauss-Newton update.

By examining the exact form of the cost function gradient $\mathbf{s}(\mathbf{w}; \mathbf{z})$ and the \mathbf{C} matrix (approximation to the empirical information matrix, i.e. $\mathbf{C} = \mathbf{r}'(\mathbf{w}; \mathbf{z})^T \mathbf{r}'(\mathbf{w}; \mathbf{z})$), we see how these quantities are calculated *recursively*, forgoing the need to store all of the observed data: As is clear from Equation 4.101, the gradient of the cost function is a function not only of the previous estimate, $\hat{\mathbf{w}}_k^-$, but also of the prior estimate covariance, $\mathbf{P}_{\mathbf{w}_k}^-$. Likewise, the \mathbf{C} matrix (Equation 4.102) is also a function of $\mathbf{P}_{\mathbf{w}_k}^-$. Furthermore, by substituting the definition of the “effective observation noise”, $\mathbf{R}_{\tilde{\mathbf{e}}} = \mathbf{P}_{\boldsymbol{\varepsilon}} + \mathbf{R}_{\mathbf{e}}$, back into Equation 4.102, we can prove the following identity:

$$\begin{aligned}\mathbf{C} &= \mathbf{r}'(\mathbf{w}; \mathbf{z})^T \mathbf{r}'(\mathbf{w}; \mathbf{z}) \\ &= \mathbf{A}^T \mathbf{R}_{\tilde{\mathbf{e}}}^{-1} \mathbf{A} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} \\ &= \mathbf{A}^T (\mathbf{P}_{\boldsymbol{\varepsilon}} + \mathbf{R}_{\mathbf{e}})^{-1} \mathbf{A} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1}\end{aligned}\tag{4.104}$$

$$= \check{\mathbf{P}}^{-1} \tag{4.105}$$

$$= (\mathbf{P}_{\mathbf{w}_k})^{-1}, \tag{4.106}$$

where we made use of the definition $\check{\mathbf{P}}^{-1}$ (Equation 4.78) going from Equation 4.104 to 4.105, and the result of Equation 4.85 for the last step. In other words, the recursively updated state estimate covariance matrix, $\mathbf{P}_{\mathbf{w}_k}$ (as calculated by the SPKF parameter estimation filter), is also the recursively (online) calculated inverse of the empirical information matrix used by the Gauss-Newton equivalent optimization step. Since the posterior distribution of the parameters $p(\mathbf{w}_k | \mathbf{z}_{1:k})$ (summarized by the sufficient statistics $\hat{\mathbf{w}}_k$ and $\mathbf{P}_{\mathbf{w}_k}$), completely captures *all* of the information contained in all the observed data up to and including time k , the SPKF does indeed implement an *online/recursive* Gauss-Newton method for parameter estimation.

4.5.4 Discussion and Experimental Demonstration

It is a well known and published fact in the literature that the EKF also approximates an online modified Newton's method for parameter estimation [12, 13, 124, 143, 125, 15, 123]. One of the questions that remains to be answered regarding the use of the SPKF for parameter estimation is the following: *What is the significant difference between the implicit Gauss-Newton nature of the SPKF when compared to the implicit Gauss-Newton nature of the EKF?* We will attempt to answer this question next:

As we showed in the previous section, the SPKF (for parameter estimation) can be interpreted as an *online Gauss-Newton method* for solving the nonlinear least-squares problem with a cost function defined by Equation 4.100. Two important terms used in this cost function is the “Jacobian-like” matrix, \mathbf{A} , and the “effective observation noise” covariance, $\mathbf{R}_{\tilde{\mathbf{e}}}$. As we showed in Section 4.2.2, the sigma-point approach used in the SPKF implicitly calculates the \mathbf{A} matrix though the process of statistical regression (stochastic linearization) and that it can be interpreted as a *statistically averaged* Jacobian of the nonlinear observation function. In effect, an *expected* Jacobian is calculated with regard to the prior distribution of the state (in this case parameter) random variable. In Section 4.5.1 (Equations 4.71 and 4.72) we showed how the “effective observation noise” term is a combination of the true observation noise and the implicit *approximation error* made by the sigma-point approach. Likewise, the “effective observation noise” covariance is the sum of the true noise covariance and the variance of the approximation error.

Through the analysis in Section 4.5.3, we showed how the \mathbf{A} and $\mathbf{R}_{\tilde{\mathbf{e}}}$ matrices are used by the SPKF's implicit Gauss-Newton optimization step. To summarize: The SPKF recursively builds up the inverse of the curvature-metric term (empirical information matrix approximation of the \mathbf{C} matrix) through the Kalman measurement update of the state estimate covariance. Equation 4.102 shows how this term is implicitly updated by the weighted outer product of the WSLR calculated \mathbf{A} matrix, where the weighting factor is given by the inverse of the effective observation noise covariance, $\mathbf{R}_{\tilde{\mathbf{e}}}^{-1}$. These two terms are also used in the calculation of the gradient term (Equation 4.101).

Carrying out a similar analysis of the EKF (as done above for the SPKF), we find that

it minimizes the following nonlinear least-squares cost function

$$\begin{aligned} f(\mathbf{w}_k) = & \frac{1}{2} [\mathbf{z}_k - \mathbf{G} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \mathbf{g}(\hat{\mathbf{w}}_k^-)]^T \mathbf{R}_{\mathbf{e}}^{-1} [\mathbf{z}_k - \mathbf{G} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-) - \mathbf{g}(\hat{\mathbf{w}}_k^-)] + \\ & + \frac{1}{2} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-)^T (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} (\mathbf{w}_k - \hat{\mathbf{w}}_k^-), \end{aligned} \quad (4.107)$$

resulting in a recursive Gauss-Newton equivalent update of the following form [12]:

$$\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_k^- + \left[\mathbf{G}^T \mathbf{R}_{\mathbf{e}}^{-1} \mathbf{G} + (\mathbf{P}_{\mathbf{w}_k}^-)^{-1} \right]^{-1} \mathbf{G}^T \mathbf{R}_{\mathbf{e}}^{-1} (\mathbf{z}_k - \mathbf{g}(\hat{\mathbf{w}}_k^-)), \quad (4.108)$$

where \mathbf{G} is the true Jacobian (first partial derivatives matrix) of the nonlinear observation function evaluated at the prior parameter estimate, i.e.,

$$\mathbf{G} = \left. \frac{\partial \mathbf{g}(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w}=\hat{\mathbf{w}}_k^-}, \quad (4.109)$$

$\mathbf{R}_{\mathbf{e}}^{-1}$ is the true observation noise covariance, and $\mathbf{g}(\hat{\mathbf{w}}_k^-)$ is simply the nonlinear observation function evaluated at the prior parameter estimate. Given Equation 4.108 we can now compare and contrast the differences between the SPKF's implicit online Gauss-Newton method and that employed by the EKF, by focusing on the following three components:

1. **Jacobian term (\mathbf{A} vs. \mathbf{G}):** The EKF uses the standard Jacobian \mathbf{G} , evaluated in a single point (the prior estimate), when calculating the relevant terms of the Gauss-Newton iteration. The SPKF on the other hand uses the WSLR approximated expected Jacobian that more accurately (on average) reflects the gradient of the nonlinear observation function as evaluated over the prior spread (distribution) of the parameter random variable. As pointed out in Section 4.2.2 this is a much more robust linearization method than simply using the truncated first order Taylor series (as in the EKF). In Figure 4.2 we show the result of a parameter estimation experiment¹¹ that illustrates how the EKF, by relying on the local gradient and not the *expected (average) gradient* (as used by the SPKF), cause it to get stuck in local minima. The expected Jacobian, used by the SPKF, causes the optimization

¹¹Even though this experiment might seem a bit contrived, it does illustrate the relevant issues quite well.

process to respond more to the relevant average slope of the error surface, whereas the normal Jacobian by definition only looks at the slope at the prior estimate. By implication, if the local¹² slope of the error surface behaves significantly differently than the average slope of the error surface over the uncertainty region of the prior estimate, the resulting optimization step (direction and size) will be quite different for the EKF vs. the SPKF. The results of the experiment shown in Figure 4.2 clearly imply that the SPKF will tend to be more robust with regard to getting stuck in local minima. Another experimental verification of this claim is evident in the results of the inverted double pendulum parameter estimation experiment of Figure 3.11.

2. **Weighting term ($\mathbf{R}_{\tilde{\mathbf{e}}}^{-1}$ vs. $\mathbf{R}_{\mathbf{e}}^{-1}$):** Using the expected gradient is, however, only half of the story. Since the SPKF implicitly calculates the expected Jacobian by drawing sigma-points from the prior distribution of the parameter estimate, it is very important that this distribution stays consistent and not collapse (become overly peaked) prematurely (before a globally good solution has been found). The EKF is notoriously bad in this regard and one often finds the filter diverging due to covariance collapse. As pointed out in Figure 4.2, this collapse (for the EKF) is in part due to the failure to compensate for the variance loss caused by the linearization error when propagating the prior covariance through the nonlinear observation function. Looking at Equations 4.103 and 4.104 through 4.106, we see that the SPKF compensates for this by effectively “adding back” the variance of the linearization error through the use of the “effective observation” noise covariance as the Gauss-Newton weighting term, i.e.

$$\mathbf{R}_{\tilde{\mathbf{e}}} = \mathbf{P}_{\boldsymbol{\epsilon}} + \mathbf{R}_{\mathbf{e}} ,$$

where $\mathbf{P}_{\boldsymbol{\epsilon}}$, the WSLR (linearization) error covariance is given by Equation 4.21. As the SPKF estimates converge to the true solution, the resulting posterior covariance should also become more peaked (depending on the variance of the true observation

¹²By local we imply within an *epsilon* neighborhood around the prior estimate, i.e. $\forall \mathbf{w}_k$ such that $\lim_{\epsilon \rightarrow 0} \|\hat{\mathbf{w}}_k^- - \mathbf{w}_k\| \leq \epsilon$. For the EKF linearization to be well behaved the prior RV must be peaked up with most of its probability mass lying within this region. If this assumption is violated (which is often the case) and the nonlinearity is quite severe, the point wise Jacobian will be a bad approximation of the average slope.

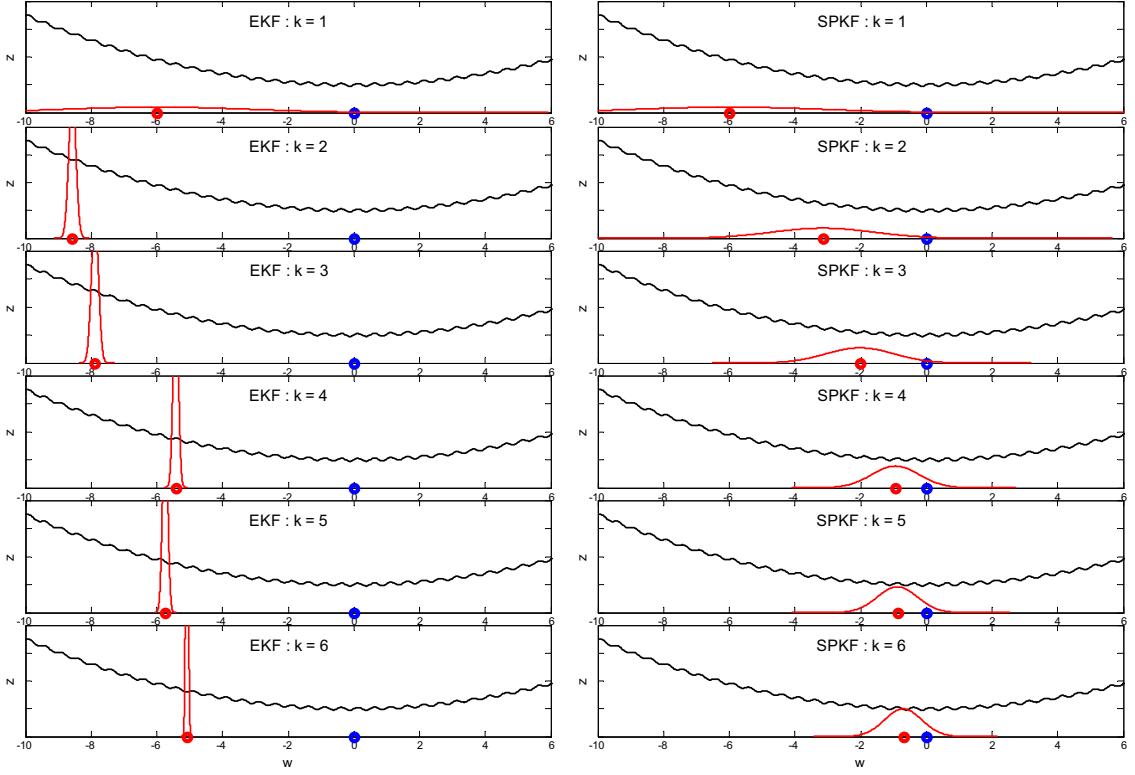


Figure 4.2: Optimization issues - local vs. average (expected) gradient of cost function: The results of a comparative (EKF vs. SPKF) parameter estimation experiment is presented here. Both filters were used to estimate the true value (blue dot) of an arbitrary parameter that was observed through a nonlinear observation function (curve shown in black). Globally (large scale) the nonlinear function has quadratic nature with a distinct minimum at the true value of the hidden parameter ($w^{true} = 0$). The small scale nature of the nonlinearity does however have numerous local minima as is evident from the plots. The plots on the left side show (from top to bottom) the first 6 iterations ($k = 1 \dots 6$) for the EKF and those on the right show the same steps for the SPKF. The shape (mean and covariance) of the posterior density function of the state (parameter) estimate is indicated in red for both filters. Both filters were initialized with identical wide prior Gaussian distributions. The EKF clearly is much more sensitive to the local (point wise) gradient of the nonlinearity. The first step of the EKF actually moves the estimate away from the true value of w since the local gradient has the opposite sign of the “average” gradient as measured over the region of w where the estimate’s distribution has significant probability mass. The distribution of the EKFs estimate also gets peaked up very rapidly, causing the algorithm to focus its “search” in a small area of the parameter space which is still far away from the optimal solution. The reason why the distribution gets so peaked early on is due (in part) to the fact that the EKF does not compensate for linearization error when propagating the prior covariance through the linearized observation function. The SPKF on the other hand, not only uses the expected (averaged) slope causing it to be less sensitive to the local gradient and more to the average global (over prior distribution of the estimate) slope, it also compensates for linearization error by using the effective observation noise covariance ($\mathbf{R}_{\bar{e}} = \mathbf{P}_e + \mathbf{R}_e$) that incorporates the variance of the linearization error. This keeps the covariance of the estimate to stay consistent, allowing for a more robust exploration of the parameter space. Both of these issues tend to help the SPKF to be more robust with regard to getting stuck in local minima, which the EKF clearly has fallen victim to in this experiment.

noise, \mathbf{R}_e). If the nonlinearity of the observation function becomes less severe (more linear) over the range of this “shrinking” covariance (of the parameter estimate), the variance of the linearization error, \mathbf{P}_ϵ , will also decrease. This will cause the effective observation noise covariance to approach the true observation noise covariance in the limit as the filter converges to a solution. One can thus think of the “effective observation noise” as an adaptive noise term that helps to regularize the SPKF especially in the early stages of the optimization process when the prior covariance of the parameter estimates still dominates the noisy observations and the filter estimate is far from the optimal solution. It is in these early stages that the EKF often behaves badly, especially if the filter was badly initialized. A typical *ad-hoc* method that is used to make the EKF more robust in these circumstances is to artificially add some “tuning noise” to the covariance estimate in order to enforce consistency. Although this might improve the robustness of the EKF, it usually results in a decrease in estimate accuracy.

3. **Predicted observation term ($\hat{\mathbf{z}}_k^-$ vs. $\mathbf{g}(\hat{\mathbf{w}}_k^-)$):** The last (and most obvious) difference between the EKF and SPKF algorithms is the manner in which the expected prediction of the nonlinear observation is calculated. As we already pointed out in Section 4.3.1, the SPKF achieves third order accuracy for the propagation of the mean of the prior Gaussian RV, through the use of the sigma-point approach, i.e.,

$$E[\mathbf{g}(\mathbf{w}_k)] \approx \hat{\mathbf{z}}_k^- = \sum_{i=0}^{2L} w_i \mathbf{g}(\mathcal{X}_i) ,$$

where the sigma-point set, $\{w_i; \mathcal{X}_i; i = 1 \dots 2L\}$ is calculated from the prior covariance of the parameter estimate, $\mathbf{P}_{\mathbf{w}_k^-}$. In contrast, the EKF only achieves first order accuracy through the first order truncation of the Taylor-series expansion of the nonlinear function of the prior RV, i.e.

$$E[\mathbf{g}(\mathbf{w}_k)] \approx \mathbf{g}(\hat{\mathbf{w}}_k^-) .$$

There is one caveat here that was hinted to in Section 3.5.2. If we are dealing with

a pure noise-less optimization problem where the prediction error can be driven to zero, the SPKF might converge to an estimate that might have a slightly higher error than the EKF. This is due to the fact that estimate covariance will only converge to a Dirac-delta function in the limit (i.e. collapse) and only if the process noise is annealed to zero (see Section 3.5.2). While the parameter estimate covariance still has a finite “spread” the SPKF will calculate the expected (averaged) best parameter estimate over this distribution, which might not lie “right in the bottom of the bowl” of the error surface. Under these circumstances, one might employ an equivalent method (once the estimate is close to the solution) to the EKF for calculating the prediction of the observation. This was referred to as *Option 2* in Section 3.5.2.

4.6 Summary of SPKF properties

Based in part on the analysis presented in this chapter, we now summarize the most salient properties of the SPKF:

- SPKF implicitly performs a *statistical linearization* of the nonlinear state transition (process) and state observation functions. Linearization error is compensated for by effectively including variance of error in calculation of posterior covariances.
- Mean and covariance of state estimate is calculated accurately to at least the second order (third order for true Gaussian priors) as opposed to the limited first order accuracy of the EKF.
- Equivalent computational complexity as EKF: $\mathcal{O}(L^3)$ in general, but a $\mathcal{O}(L^2)$ implementation is possible for parameter estimation.
- Computationally efficient and numerically robust implementations available through use of square-root forms.
- In contrast to EKF, no analytical derivatives (Jacobians or Hessians) need to be calculated: The utility of this is especially valuable in situations where the system at hand is a “black box” model for which the internal dynamic equations are unavailable. In order to apply an EKF to such systems, derivatives must be found either

from a principled analytic re-derivation of the system, or through costly and often inaccurate numerical methods (e.g., by perturbation). On the other hand, the SPKF relies on only functional evaluations (inputs and outputs) through the use deterministically drawn samples from the prior distribution of the state random variable. From a coding perspective, this also allows for a much more general and modular implementation.

- SPKF for parameter estimation is a new efficient online 2nd order optimization method. The SPKF (for parameter estimation) can be interpreted as an online recursive Gauss-Newton method utilizing an adaptive “effective observation noise” term and a prior parameter distribution averaged (expected) Jacobian of the error surface, to iteratively build up an approximation of the inverse of the empirical Fisher information matrix. This Hessian-like matrix serves as a curvature-metric to determine the (locally) optimal step size for a Fisher-scoring step in the parameter space. Based in part on these qualities, the SPKF seems to have better robustness properties, i.e., less likely to get stuck in non-optimal local minima.
- The SPKF consistently outperforms (or at least equals in some circumstances) the EKF for state, parameter and dual estimation. The SPKF generates estimates which are more accurate (lower estimation error) and estimates of the error covariance which tend to be more consistent, resulting in increased filter robustness and decreased likelihood of filter divergence.

4.7 Chapter Summary

In this chapter we analyzed certain theoretical characteristics of the sigma-point approach and the resulting sigma-point Kalman filter. We first presented an alternative interpretation of the sigma-point approach based on a technique called weighted statistical linear regression (statistical linearization), which allows for useful insight into why the SPKF performs better and more robustly than the EKF for Gaussian approximate nonlinear estimation.

In an analysis of the accuracy of the sigma-point approach, we showed how it attains at

least second order accuracy in calculation of the posterior statistics of a Gaussian random variable that undergoes an arbitrary nonlinear transformation. This was contrasted to the limited 1st order accuracy of the EKF which is achieved at the same computational cost as the SPKF.

A relationship was drawn between the sigma-point approach and the numerical integration technique called Gauss-Hermite quadrature. We showed how (at least for a limited scalar case) the SPKF is equivalent to a related algorithm called the Gauss-Hermite filter. The Gauss-Hermite filter, although theoretically superior to the SPKF with regard to accuracy of the posterior statistics calculation, was shown to be implementationally impractical for general estimation problems with more than a few system states.

One of the main contributions of this thesis was presented in this chapter, namely the analysis of the relationship between the SPKF for parameter estimation and other 2nd order optimization methods within the modified-Newton method family. We showed how the SPKF minimizes a MAP cost function (under a Gaussian assumption), which in turn can be interpreted as a novel adaptation of the Gauss-Newton optimization method.

The chapter was ended by summarizing the most salient features of the SPKF that has been revealed and demonstrated through the analysis in this chapter and the experimental verification of the previous chapter.

Chapter 5

SPKF Based UAV Autonomy

5.1 Introduction

Two large ongoing research projects within our group are the DARPA sponsored “*Model-Relative Control of Autonomous Vehicles*” [38, 105] and ONR sponsored “*Sigma-Point Kalman Filter Based Sensor Integration, Estimation and System Identification for Enhanced UAV Situational Awareness & Control*” [149, 203] projects. Both these overlapping research efforts focus on the autonomous control of an *unmanned aerial vehicle* (UAV), specifically, an autonomous small-scale helicopter as shown in Figures 5.2 and 5.3.

The core component of such a UAV is a high-performance digital computer based *guidance, navigation & control* (GNC) system as schematically depicted in Figure 5.1. The main subsystems of the GNC system is a control system (CS) and a guidance & navigation system (GNS). The GNS takes noisy avionics sensor measurements as input, and then fuses¹ them in a probabilistic sense with predictions from a vehicle dynamics model in order to calculate optimal vehicle navigation state solutions. These state estimates together with desired flight trajectories are then fed to the control system that computes some form of optimal control law to drive the flight surface actuators of the vehicle.

The current state-of-the-art probabilistic inference system used for such UAV guidance and navigation systems are usually EKF based [56, 58, 42]. Our experimental platform (see Section 5.2) which was developed by MIT [52], is indeed built around a EKF based GNS implementation. This EKF based estimator was adapted and “hand tuned” over a

¹The term “fuse” refers to the (Kalman) optimal *combination* of the prior state estimate with the new information contained in the sensor measurement. This is achieved through the Kalman measurement-update step.

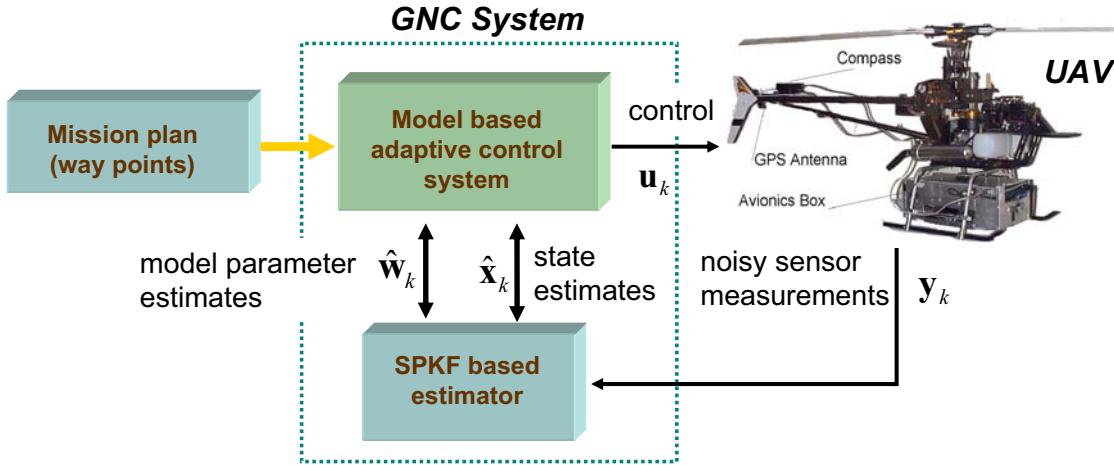


Figure 5.1: Schematic diagram unmanned aerial vehicle (UAV) guidance, navigation and control (GNC) system.

considerable amount of time in order to deliver high quality estimates for the specific UAV platform at hand. This system has been tested, debugged and verified extensively not only in simulation², but also through numerous real test-flights [60, 61].

As a real-world experimental verification of our research on the SPKF (as reported in the previous chapters of this thesis), we replaced the EKF in the UAV-GNC system by a SPKF and compared its performance to the existing EKF base-line system, with specific focus on:

- Improved six-degrees-of-freedom (6DOF) state estimation accuracy (relative to EKF).
- SPKF based compensation for GPS latency.
- Evaluation of improved control envelope due to use of better state estimator.
- SPKF based parameter estimation: Track certain UAV model parameters such as mass, moments of inertia, etc.
- SPKF based dual estimation: Simultaneously estimate 6DOF state, as well as secondary set of hidden dynamic model states (blade flapping angles, etc.) and attempt

²This includes “software only” as well as “hardware-in-the-loop” simulations. See Section 5.2 for more details on the different simulators.

to track a subset of vehicle model parameters (mass, etc.)

This brought together numerous theoretical and algorithmic aspects and served as a proof of concept for integrating the different separately developed SPKF technologies into a serious, real-world application. In this chapter we will cover the work related to this effort and report on our experimental findings.

5.2 Experimental Platform

In this section we briefly summarize the different component parts of the UAV platform. These include the vehicle airframe, the avionics subsystem, the mathematical dynamic state-space models used for simulator, controller and estimator design as well as the simulators themselves. For a complete in-depth exposition of the design, implementation, verification and testing of the complete UAV platform, please see [58].

Since the goals of the two above mentioned projects (DARPA & ONR) are research related to UAV autonomy (control and estimation) and not flight hardware development, we opted to subcontract out the manufacture of the experimental platform to the Aerial Robotics group at MIT's Laboratory for Information and Decision Systems [52]. The group at MIT (who was part of the multi-institute, multi-PI DARPA project [38]) already had a fully developed, debugged and operational UAV experimental platform built around the *X-Cell 90* radio controlled (R/C) helicopter [136]. This helicopter is outfitted with a state-of-the art avionics subsystem comprising a PC-104 based flight computer augmented by a high-grade IMU, GPS, digital magnetic compass and digital barometric altimeter. More detail on the avionics system is provided in Section 5.2.2. A large part of the MIT research effort was aimed at the development of a state-of-the-art high-fidelity nonlinear mathematical model for small helicopter dynamics. This model was provided as part of the sub-contracted deliverables such that we could readily leverage it in our own research. Section 5.2.3 provides further detail about this and other vehicle models used in our research.

The final component of the experimental platform is a high-fidelity *simulation environment* build around the previously mentioned high-fidelity helicopter model. The simulation



Figure 5.2: Close-up of instrumented X-Cell-90 helicopter in flight. The silver box between the helicopter's landing gear is the avionics box containing the flight computer, communication equipment, GPS, IMU and barometric altimeter. [photo courtesy of Vlad Gavrilets [58]].

environment consisted out of four separate simulators, operating at different levels of abstraction and fidelity. These are discussed in Section 5.2.5. We made extensive use of these simulation environments for the results reported in this chapter.

5.2.1 Vehicle Airframe

As mentioned above, an X-Cell 90 radio controlled hobby helicopter was chosen as the airframe around which the rest of our UAV experimental platform was built. The helicopter weighs about 10 lbs (empty) and carries a 7 lbs avionics payload and 1lbs of fuel. With a full fuel load, the helicopter can stay in the air for approximately 9-10 minutes. The X-Cell 90 has a hingeless main rotor that is equipped with a Bell-Hiller stabilizer bar [24], which provides lagged rate feedback and augments the servo torque with an aerodynamic moment used to change the cyclic pitch of the blades. The helicopter is powered by a single piston internal combustion engine running on a mixture of alcohol and oil. The helicopter



Figure 5.3: Instrumented X-Cell-90 helicopter in flight.

is equipped with an electronic governor that provides closed-loop engine speed control outside of the main (software controlled) vehicle control loop. The electronic governor, a Futaba-GV1, is a COTS³ device that measures the engine speed with a magnetic sensor and adjusts the engine throttle setting in order to maintain an operator commanded average setpoint (typically around 1600 rpm). This is a self-contained closed loop control system that is not accessible to the flight computer. The rotor speed (which is proportional to the engine speed via the gearbox ratio), although needed for precise and robust control of aggressive aerial maneuvers, is thus not accessible to the flight computer and must be estimated by the estimation system.

Figures 5.3 and 5.2 shows the helicopter in flight carrying the avionics box mounted on a custom suspension system, used to attenuate certain unwanted vibration inputs. High vibration levels significantly degrade performance of inertial sensors (IMU) leading to high drift rates, and can also increase the likelihood of electronic component failures such as broken solder joints and loose connectors. There are several sources of vibration in a helicopter such as the main rotor, the tail rotor and the engine, all of which can

³common off-the-shelf

excite lightly damped structural modes, e.g. the first bending mode of the tail boom. The MIT group has experimentally determined the primary vibration source as the once-per-revolution component coming from the main rotor with a nominal frequency of about 27 Hz [58]. The suspension system, consisting of four neoprene isolators, was designed to attenuate this specific frequency while still being stiff enough not to couple with the modes of interest needed for vehicle state estimation and control.

5.2.2 Avionics System

Autonomous aerobatics calls for a high-bandwidth control system, which in turn necessitates fast servo-mechanisms, low latencies in the flight computer and fast sensor responses. The actuators used (Futaba digital servos) were chosen to meet these strict load and bandwidth demands. See [58] for full detail on the servo design analysis. The sensor suite used on the X-Cell 90 platform consists of an IMU, a GPS receiver and a barometric altimeter. These will now be discussed in more detail:

The core of the avionics sensor suite is an inertial measurement unit (IMU) from Inertial Sciences (ISIS-IMU) [85], that provides measurements of the vehicle's 3D linear accelerations (in body frame), i.e.

$$\mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T , \quad (5.1)$$

as well as the vehicle's rotational rates in the body frame, i.e.,

$$\boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^T , \quad (5.2)$$

where p is the roll rate, q is the pitch rate and r is the yaw rate. All rates are measured in radians per second. (See Figures B.1 and B.2 in Appendix B for an illustration of the different frames of reference used in this and subsequent discussions.) The ISIS-IMU makes these measurements using three MEMS linear accelerometers and three temperature stabilized MEMS gyroscopes. The range of the gyroscopes was set to ± 300 deg/sec, and the range of the accelerometers to ± 5 g's, where 1 g is the normal gravity induced acceleration directed at the center of the earth, i.e., 9.81 m/s^2 . The helicopter with a full payload is

capable of achieving roll rates of up to 200 deg/sec, and yaw rates in excess of 1000 deg/sec. Higher full scale settings for the gyros lead to higher drift rates, and to avoid the latter, the yaw rate command was software limited to avoid sensor saturation and possible control instability. The IMU has internal power regulation, temperature compensation, a serial output update rate of 100Hz and internal first order analog anti-aliasing filters. The drift rates of the gyroscopes (which must be compensated for using the estimator subsystem) are in the order of 0.02 deg/sec, and for the accelerometers 0.02 g over the typical duration of a flight.

For direct absolute altitude measurements, $z = z^{alt}$, a digital barometric altimeter from Honeywell, the HPA200 [81], is used. This altimeter provides 0.6 meter resolution (0.001 psi) and has excellent stability. Due to the short flight durations, ambient pressure changes are negligible and hence ignored. The altimeter has internal power regulation, temperature compensation and a serial output which is sampled at 5 Hz. The altimeter and its pressure port are housed inside the avionics box in order to shield the measurements from pressure changes in the main rotor wake and from dynamic pressure changes induced by the vehicle motion and wind. This results in reliable altitude measurements once the helicopter is outside the ground effect zone (about 2 rotor diameters or 3 meters above the ground).

A high quality Global Positioning System (GPS) receiver, the G12 from Ashtech [130], is used to provide 3D measurements of vehicle position and velocity in the ECEF (Earth-centered Earth-fixed) reference frame⁴, i.e.,

$$\mathbf{p}_{gps} = \begin{bmatrix} x & y & z \end{bmatrix}_{ECEF}^T, \quad (5.3)$$

and

$$\mathbf{v}_{gps} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}_{ECEF}^T = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}_{ECEF}^T. \quad (5.4)$$

This GPS receiver has a 10 Hz update rate and a 50 ms latency. The latency implies that the measurements coming from the GPS relates to the vehicle state as it was 50 ms in the past. MIT's group deemed this latency to be short enough to have no discernible impact on

⁴See Appendix B for a definition of the different navigational reference frames.

their state estimators accuracy [59]. This combined with the fact that EKF based latency-compensation techniques are hard to implement and often result in no real improvement (if not degradation) in estimation accuracy, led them to ignore it completely and not compensate for it in their state estimator. Since a *SPKF based latency-compensation technique* is not only elegantly accommodated within the SPKF theoretical framework, but also implemented with relative ease, we decided to include latency compensation in our state estimator. Detail of this is provided in Section 5.3.

Other avionics devices include a flight control computer (TP400 from DSP design equipped with a Pentium MMX 300 Mhz compatible CPU, RAM, non-volatile FLASH memory and I/O subsystems), a wireless LAN transceiver for data and telemetry up- and downlinking from the base-station, a remote control radio receiver for pilot commands, and a custom-designed servo board for sampling pilot commands and driving the servo-mechanisms at 50Hz. The flight control processor runs the QNX 4.25 real-time operating systems (RTOS) [107] and all system software is implemented in the C programming language.

5.2.3 Nonlinear Vehicle Models

Accurate computational models of the helicopter dynamics and avionics sensors are needed to build accurate, high-fidelity simulators as well as high-performance control and estimation systems. Unfortunately, there is always a compromise between the fidelity of any given model and the computational resources needed to simulate it. Systems that by design are allowed to run off-line or at sub-real-time speeds, such as simulators, or systems that can run on high performance ground-based (e.g. in the ground station computer) assets, can make use of very high-fidelity models that demand significant computational resources. On the other hand, algorithms that have to run on-board the UAV's flight computer and meet stringent timing, CPU load and power requirements, often have to resort to using less complex (lower fidelity) approximate models. Since our main control loop in the flight computer runs at half the IMU rate (50 Hz), all of our processing (control step, estimator step, actuator command step) has to be completed within a single 20ms computation window. This places strict limitations on the complexity of the process model used inside

our estimation algorithm. For this reason we made use of two vehicle dynamic models in our research: For the simulator (used to test our algorithms), we used the full complexity high-fidelity nonlinear model developed by MIT (based on an original design by Draper Laboratories), which we call the *MIT-Draper-XCell-90 model*. For the online, on-board computations inside our SPKF based state estimator we used a reduced *kinematic model* driven directly by the high quality IMU inputs. We will now discuss these two different models in more detail. (Note: Please consult Appendix B for definitions of the different navigational reference frames used in the discussion of the helicopter models).

MIT-Draper-XCell-90 Model

The MIT-Draper-XCell-90 model is a high-fidelity quaternion based nonlinear model of small helicopter dynamics utilizing 43 states

$$\mathbf{x} = [\mathbf{p}^T \ \mathbf{v}^T \ \mathbf{e}^T \ \boldsymbol{\omega}^T \ \boldsymbol{\beta}^T \ \boldsymbol{\Omega}^T \ \boldsymbol{\omega}_{box}^T \ \dot{\boldsymbol{\omega}}_{box}^T \ \mathbf{s}_1^T \ \mathbf{s}_2^T \ \mathbf{s}_3^T \ \mathbf{s}_4^T]^T , \quad (5.5)$$

where $\mathbf{p} = [x \ y \ z]^T$ is the position vector in the NED reference frame, $\mathbf{v} = [u \ v \ w]^T$ is the velocity vector in the body-frame, $\mathbf{e} = [e_0 \ e_1 \ e_2 \ e_3]^T$ is the attitude quaternion vector, $\boldsymbol{\omega} = [p \ q \ r]^T$ is the rotational rate vector, $\boldsymbol{\beta} = [\beta_{lat} \ \beta_{lon}]$ is the flapping angle vector, $\boldsymbol{\Omega} = [\Omega_{mr} \ \Omega_{mrl}]^T$ is the main rotor rotation rate vector, $\boldsymbol{\omega}_{box} = [p_{box} \ q_{box} \ r_{box}]^T$ is the rotational rate vector of the suspended avionics box, $\dot{\boldsymbol{\omega}}_{box} = [\dot{p}_{box} \ \dot{q}_{box} \ \dot{r}_{box}]^T$ is the rotational acceleration vector of the suspended avionics box and $\mathbf{s}_i = [s_{i,0} \ s_{i,1} \ s_{i,2} \ s_{i,3} \ s_{i,4}]^T$ are the actuator state vectors for the four main vehicle servos ($i = 1, \dots, 4$). All of the state vector components are summarized in Table 5.1. The model is further parameterized by the following 70 dimensional parameter vector,

$$\mathbf{w} = [m \ I_{xx} \ I_{yy} \ I_{zz} \ \cdots \ Q_e^m]^T , \quad (5.6)$$

where the components are described in Table 5.2. The control input vector to the vehicle dynamic model

$$\mathbf{u}_k = \left[u_{lon} \ u_{lat} \ u_{col} \ u_{tcol} \right]_k^T , \quad (5.7)$$

Table 5.1: MIT-Draper-XCell-90 model state vector components

Variable	Description	Variable	Description
x	position in NED frame (north)	p	roll rate of vehicle
y	position in NED frame (east)	q	pitch rate of vehicle
z	position in NED frame (down)	r	yaw rate of vehicle
u	velocity in body frame (longitudinal)	p_{box}	roll rate of suspended avionics box
v	velocity in body frame (lateral)	q_{box}	pitch rate of suspended avionics box
w	velocity in body frame (vertical)	r_{box}	yaw rate of suspended avionics box
e_0	attitude quaternion (1st component)	\dot{p}_{box}	roll acceleration of suspended avionics box
e_1	attitude quaternion (2nd component)	\dot{q}_{box}	pitch acceleration of suspended avionics box
e_2	attitude quaternion (3rd component)	\dot{r}_{box}	yaw acceleration of suspended avionics box
e_3	attitude quaternion (4th component)	Ω_{mr}	main rotor rotational rate
β_{lon}	longitudinal flapping angle	Ω_{mrI}	integral of main rotor rotational rate
β_{lat}	lateral flapping angle		

corresponds to rotor blade pitch angles, i.e., main rotor longitudinal, lateral and collective inputs and tail rotor collective input. The model accurately accounts for rotor forces, torque, thrust, flapping dynamics, horizontal stabilizer, and vertical tail forces and moments, fuselage drag, and actuator states. This model has been found to be accurate as a basis for control design through pilot observations and flight-testing [58].

Without going into the exact detail⁵, the MIT-Draper-XCell-90 helicopter model can be described as a generic six-degree-of-freedom (6DOF) rigid body model with external forces and moments originating from the main and tail rotors, vertical and horizontal fins and fuselage drag [61]. All actuator responses are modeled by 5th order ARMAX models. The continuous time partial differential equations of the model is discretized using a fourth

⁵For a fully detailed derivation of the MIT-Draper-XCell-90 model, please consult [58].

Table 5.2: MIT-Draper-XCell-90 model parameters

Parameter	Description	Parameter	Description
m	helicopter mass	$\tilde{\mathbf{r}}_{hub}^{mr}, \tilde{\mathbf{r}}_{hub}^{tr}$	location of main and tail rotor hubs in body frame
I_{xx}, I_{yy}, I_{zz}	moments of inertia around rolling, pitching and yawing axes	$\tilde{\mathbf{r}}_{\rho_c}^h, \tilde{\mathbf{r}}_{\rho_c}^v$	location of center of pressure for horizontal and vertical fins in body frame
I_{xz}	roll-yaw cross moment of inertia	$\tilde{\mathbf{r}}_{gps}$	location of GPS antenna in body frame
Ω_{mr}^{nom}	nominal main rotor (m.r.) RPM	$\tilde{\mathbf{r}}_{imu}$	location of IMU c.g. in body frame
K_β	main rotor hub torsional stiffness	$\tilde{\mathbf{r}}_{lbg}$	location of left-back landing gear in body frame
K_μ	scaling factor of flap response to speed variation	$\tilde{\mathbf{r}}_{rgf}$	location of right-front landing gear in body frame
γ_{fb}	fly-bar lock number	$\tilde{\mathbf{r}}_{cp}^{fus}$	fuselage moment offset
K_{lon}, K_{lat}	longitudinal and lateral cyclic to flap gain	$\tilde{\mathbf{r}}_{cg}$	body axis coordinates of center of gravity (c.g.)
\tilde{r}_{mr}	main rotor radius	\tilde{r}_{tr}	tail rotor radius
\tilde{c}_{mr}	main rotor cord	\tilde{c}_{tr}	tail rotor cord
C_{la}^{mr}	main rotor lift curve slope	C_{la}^{tr}	tail rotor lift curve slope
C_{do}^{mr}	main rotor profile drag coefficient	C_{do}^{tr}	tail rotor profile drag coefficient
$C_{T_{max}}^{mr}$	main rotor maximum thrust coefficient	ϕ_o^{tr}	tail rotor pitch offset
I_b^{mr}	main rotor blade inertia	α_q^{tr}	tail rotor gear ratio
A_{vf}	vertical fin area	\tilde{A}_{hf}	horizontal fin area
C_{la}^{vf}	vertical fin	C_{la}^{hf}	horizontal fin
F_t^{vf}	vertical fin blockage factor	\tilde{A}_V^{fus}	fuselage vertical area
\tilde{A}_F^{fus}	fuselage frontal area	\tilde{A}_S^{fus}	fuselage side area
f_ϕ^{box}	suspension box pitch frequency	f_ψ^{box}	suspension box yaw frequency
f_θ^{box}	suspension box roll frequency	ζ^{box}	suspension damping ratio
P_{max}	maximum engine power	P_{idle}	engine idle power
K_{gov}^p	governor gain - proportional component	K_{gov}^i	governor gain - integral component
v_{wind}	wind speed in inertial frame	Q_e^m	mean engine torque
ψ_{wind}	wind direction in inertial frame		

order Runge-Kutta integration scheme with a 10ms integration step-size to generate a high-fidelity nonlinear discrete time model of the form

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k; \mathbf{w}) , \quad (5.8)$$

with the states, control inputs and parameters as defined above. In order to simulate this model, all of the above mentioned forces and moments need to be calculated as functions of the current state of the vehicle, the model parameters and certain external disturbances. These are computationally intensive calculations, some of which depend on iterating non-closed-form solutions. See [22, 58] for more detail on how this is accomplished. As stated earlier, this high-fidelity model forms the core of all of our simulation environments, but is too computationally complex to use onboard as the process model required by our SPKF based estimator. For this purpose, we make use of a IMU driven *kinematic model* of 3 dimensional holonomic movement. This model will now be introduced:

Kinematic Model

In order to derive a simpler 6DOF helicopter process model for use inside the state estimator, we need to resort to a *kinematic model* of three dimensional (3D) movement (translation and rotation). The main benefit of a kinematic model is that it is for the most part *vehicle agnostic*, in that it does not need to explicitly model the forces and moments that act upon the vehicle. It rather uses the outputs of an IMU (linear accelerations and rotational velocities) to directly drive the 6DOF kinematic differential equations. The IMU can be thought of as summarizing (through measurement and not modeling) all of the forces and moments acting upon the vehicle. For this reason kinematic models can easily be adapted for different vehicles. The only vehicle related parameters used by the model is the location (relative to the center of gravity of the vehicle) of where the IMU is mounted. This makes such models very appealing for “strap-down” type state estimators, like the one presented in this chapter. We will now give the specific detail of how the kinematic model is constructed:

For the purpose of the core state estimation filters (SPKF or EKF based) the following 16 dimensional state vector is used (the definition of the component variables are given in

Table 5.3: Kinematic model state vector components.

Variable	Description	Variable	Description
x	north position in NED frame	a_{b_x}	IMU x -accelerometer bias
y	east position in NED frame	a_{b_y}	IMU y -accelerometer bias
z	down position in NED frame	a_{b_z}	IMU z -accelerometer bias
v_x	north velocity in NED frame	w_{b_p}	IMU θ (roll) gyro rate bias
v_y	east velocity in NED frame	w_{b_q}	IMU ϕ (pitch) gyro rate bias
v_z	down velocity in NED frame	w_{b_r}	IMU ψ (yaw) gyro rate bias
e_0	attitude quaternion : first component	e_2	attitude quaternion : third component
e_1	attitude quaternion : second component	e_3	attitude quaternion : fourth component

Table 5.3):

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}^T & \mathbf{v}^T & \mathbf{e}^T & \mathbf{a}_b^T & \boldsymbol{\omega}_b^T \end{bmatrix} \quad (5.9)$$

$$= \begin{bmatrix} x & y & z & v_x & v_y & v_z & e_0 & e_1 & e_2 & e_3 & a_{x_b} & a_{y_b} & a_{z_b} & p_b & q_b & r_b \end{bmatrix}^T \quad (5.10)$$

where $\mathbf{p} = [x \ y \ z]^T$ is the NED frame⁶ position vector, $\mathbf{v} = [v_x \ v_y \ v_z]^T$ is the NED frame vehicle velocity vector, $\mathbf{e} = [e_0 \ e_1 \ e_2 \ e_3]^T$ is the unity norm vehicle attitude quaternion, $\mathbf{a}_b = [a_{x_b} \ a_{y_b} \ a_{z_b}]^T$ is the vector of IMU acceleration (drift) biases, and $\boldsymbol{\omega}_b = [p_b \ q_b \ r_b]^T$ is the IMU gyro rate bias vector. A quaternion (as opposed to Euler angle) attitude representation is used, since it is free of the trigonometric singularities⁷ associated with the Euler representation. This representation is however subject to a unit norm constraint [175].

Using the definition of the IMU accelerometer and gyro rate measurements (Equations 5.1 and 5.2) we can now write down the continuous time⁸ *kinematic* navigation

⁶See Appendix B for definitions of the different navigational reference frames.

⁷The well known “Gimbal Lock” problem. See Appendix B for more detail.

⁸For conciseness of notation we have dropped the explicit indication of time dependence of the vehicles state components, i.e., \mathbf{p} , \mathbf{v} , \mathbf{e} , \mathbf{a}_b and $\boldsymbol{\omega}_b$ implies \mathbf{p}_t , \mathbf{v}_t , \mathbf{e}_t , \mathbf{a}_{b_t} and $\boldsymbol{\omega}_{b_t}$. The same time dependency is implied for the acceleration and rotation rate terms, i.e., $\bar{\mathbf{a}}$, \mathbf{n}_a , $\mathbf{a}_{\tilde{r}_{imu}}$ and $\tilde{\boldsymbol{\Omega}}$ implies $\bar{\mathbf{a}}_t$, \mathbf{n}_{a_t} , $\mathbf{a}_{\tilde{r}_{imu,t}}$.

equations [58]:

$$\dot{\mathbf{p}} = \mathbf{v} \quad (5.11)$$

$$\dot{\mathbf{v}} = (\mathbf{T}_{b2i})(\bar{\mathbf{a}} - \mathbf{a}_{\tilde{\mathbf{r}}_{imu}}) + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T g \quad (5.12)$$

$$\dot{\mathbf{e}} = -\frac{1}{2}\tilde{\Omega}\mathbf{e} + j\lambda\mathbf{e} \quad (5.13)$$

$$\dot{\mathbf{a}}_b = \mathbf{0} \quad (5.14)$$

$$\dot{\boldsymbol{\omega}}_b = \mathbf{0} \quad (5.15)$$

where \mathbf{T}_{b2i} is the direction cosine matrix (DCM) transforming vectors from the body axis to the NED reference frame, given by

$$\mathbf{T}_{b2i} = 2 \begin{bmatrix} 0.5 - e_2^2 - e_3^2 & e_1e_2 - e_0e_3 & e_1e_3 + e_0e_2 \\ e_1e_2 + e_0e_3 & 0.5 - e_1^2 - e_3^2 & e_2e_3 - e_0e_1 \\ e_1e_3 - e_0e_2 & e_2e_3 + e_0e_1 & 0.5 - e_1^2 - e_2^2 \end{bmatrix}, \quad (5.16)$$

$\bar{\mathbf{a}}$ and $\bar{\boldsymbol{\omega}}$ are the bias and noise corrected IMU accelerometer and gyro rate measurements, i.e.

$$\bar{\mathbf{a}} = \tilde{\mathbf{a}} - \mathbf{a}_b - \mathbf{n}_a \quad (5.17)$$

$$\bar{\boldsymbol{\omega}} = \tilde{\boldsymbol{\omega}} - \boldsymbol{\omega}_b - \mathbf{n}_\omega, \quad (5.18)$$

where $\tilde{\mathbf{a}}$ and $\tilde{\boldsymbol{\omega}}$ are the measurements coming from the IMU, \mathbf{n}_a and \mathbf{n}_ω are the IMU acceleration and gyro-rate measurement noise terms, g is the gravitational acceleration component, $\lambda = 1 - \|\mathbf{e}\|^2$ is the deviation of the square of the quaternion norm from unity due to numerical integration errors, and j is the factor that determines the convergence speed of the numerical error. These factors serve the role of Lagrange multipliers ensuring that the norm of the quaternion remains close to unity [164]. The constraint on the speed of convergence for stability of the numerical solution is $j \cdot dt < 1$, where dt is the integration time step [58]. In Section 5.3 we introduce another SPKF centric method we used to further ensure this constraint. $\tilde{\Omega}$ is 4×4 skew-symmetric matrix [181] composed of the IMU gyro

measurements and gyro bias values, i.e.,

$$\tilde{\Omega} = \begin{bmatrix} 0 & \bar{\omega}_p & \bar{\omega}_q & \bar{\omega}_r \\ -\bar{\omega}_p & 0 & -\bar{\omega}_r & \bar{\omega}_q \\ -\bar{\omega}_q & \bar{\omega}_r & 0 & -\bar{\omega}_p \\ -\bar{\omega}_r & -\bar{\omega}_q & \bar{\omega}_p & 0 \end{bmatrix} \quad (5.19)$$

$$= \begin{bmatrix} 0 & p - p_b & q - q_b & r - r_b \\ -(p - p_b) & 0 & -(r - r_b) & q - q_b \\ -(q - q_b) & r - r_b & 0 & -(p - p_b) \\ -(r - r_b) & -(q - q_b) & p - p_b & 0 \end{bmatrix}, \quad (5.20)$$

and $\mathbf{a}_{\tilde{\mathbf{r}}_{imu}}$ is a rotation-moment-linear-acceleration coupling component due to the IMU not being located at the center of gravity of the vehicle and is given by

$$\mathbf{a}_{\tilde{\mathbf{r}}_{imu}} = \dot{\boldsymbol{\omega}} \times \tilde{\mathbf{r}}_{imu} + \bar{\boldsymbol{\omega}} \times (\bar{\boldsymbol{\omega}} \times \tilde{\mathbf{r}}_{imu}) , \quad (5.21)$$

where $\tilde{\mathbf{r}}_{imu}$ is the location vector of the IMU in the body frame (with origin at the center of gravity) and \times is the normal vector-cross-product.

Equation 5.11 of the kinematic model assumes that the time-derivative of the vehicle's position ($\dot{\mathbf{p}} = d\mathbf{p}_t/dt$) is equal to its velocity.

Equation 5.12 expresses the time derivative of the vehicle's velocity ($\dot{\mathbf{v}} = d\mathbf{v}_t/dt$) in terms of the IMU measured linear accelerations. We first correct these acceleration measurements by subtracting the current estimate of the IMU accelerometer bias vector, \mathbf{a}_b . This is reflected by Equation 5.17 which also shows the correction that need to be done for the IMU measurement noise, \mathbf{n}_a , that corrupts the acceleration measurements in an additive fashion. This noise term is usually modeled as a zero-mean Gaussian random variable with variance set according to the IMU manufacturer specified sensor accuracy. This noise term and the noise on the IMU gyro rate measurements, effectively becomes the process noise term that drives the system dynamics when the kinematic model is later embedded in a SPKF based state estimator.

Since the IMU acceleration measurements will also reflect the effect of gravitational

acceleration on the vehicle, we need to discount this component of the measurement in the NED frame. More specifically, when the vehicle is stationary, the downwards pointing component of the IMU acceleration measurement will be: $a_z = -g$. In order to discount this effect and accurately model the true acceleration of the vehicle, Equation 5.12 adds $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T g$ to the IMU measured accelerations once they are transformed into the NED frame from the body frame. We already discussed how the model further discounts the effect of rotation-to-acceleration coupled IMU measurements through the use of the $\mathbf{a}_{\tilde{\mathbf{r}}_{imu}}$ term. In the first experiment of Section 5.4 we show how this term can often be ignored if the magnitude of the IMU offset, $\tilde{\mathbf{r}}_{imu}$, is not too large. As stated earlier, the direction cosine matrix, \mathbf{T}_{b2i} , is constructed from the components of the attitude quaternion vector and used to transform the body-frame measured accelerations into the NED frame which is the reference frame in which the vehicles position and velocity states are expressed.

Equation 5.13 exploits one of the elegant properties of the quaternion framework for attitude representation. That is, the time derivative of the vehicle's attitude quaternion ($\dot{\mathbf{e}} = d\mathbf{e}_t/dt$) is given by a simple linear transformation of the current attitude quaternion where the 4-by-4 skew-symmetric transformation matrix $\tilde{\Omega}$ is determined by the rotational rates of the vehicle. These rates are directly measured by the IMU, compensated for estimated bias and measurement noise (Equation 5.18) and then “plugged” into the $\tilde{\Omega}$ matrix according to Equation 5.19.

The final components of the kinematic model, Equations 5.14 and 5.15, indicate that we assume the IMU biases are unknown but constant values with no defined dynamics driving their time evolution. In practice however, when using a SPKF to estimate their values, we model these terms as random walk processes, i.e.,

$$\mathbf{a}_{b_{k+1}} = \mathbf{a}_{b_k} + \mathbf{r}_{\mathbf{a}_{b_k}} \quad (5.22)$$

$$\boldsymbol{\omega}_{b_{k+1}} = \boldsymbol{\omega}_{b_k} + \mathbf{r}_{\boldsymbol{\omega}_{b_k}}, \quad (5.23)$$

where $\mathbf{r}_{\mathbf{a}_{b_k}}$ and $\mathbf{r}_{\boldsymbol{\omega}_{b_k}}$ are synthetic zero-mean Gaussian random variables. These noise terms help to increase the convergence speed of the estimator as well as increase the tracking robustness of the estimator in case these bias terms drift slowly over time. This bias

drifting is often caused by changing thermal conditions affecting the IMU in real vehicles. The variance of these noise terms can be annealed over time in a similar manner as for SPKF based parameter estimation (see Section 3.5.2).

Discretizing the kinematic model

We already showed at the end of the previous section how the bias components of the model can be treated in a discrete fashion. We will next discuss how the remaining components of the continuous time kinematic model is converted into a discrete time model for use within the SPKF framework.

The quaternion propagation equation can be discretized with an analytical calculation of the exponent of the skew-symmetric matrix given by Stevens [181]. The discrete-time update can be written as

$$\mathbf{e}_{k+1} = \exp\left(-\frac{1}{2}\tilde{\boldsymbol{\Omega}} \cdot dt\right) \mathbf{e}_k . \quad (5.24)$$

If we further denote

$$\Delta\phi = p \cdot dt \quad (5.25)$$

$$\Delta\theta = q \cdot dt \quad (5.26)$$

$$\Delta\psi = r \cdot dt , \quad (5.27)$$

as effective rotations around the (body frame) roll, pitch and yaw axes undergone by the vehicle during the time period dt , assuming that the angular rates p, q and r remained constant during that interval, we can introduce the 4×4 skew-symmetric matrix

$$\begin{aligned} \boldsymbol{\Phi}_\Delta &= \tilde{\boldsymbol{\Omega}} \cdot dt \\ &= \begin{bmatrix} 0 & \Delta\phi & \Delta\theta & \Delta\psi \\ -\Delta\phi & 0 & -\Delta\psi & \Delta\theta \\ -\Delta\theta & \Delta\psi & 0 & -\Delta\phi \\ -\Delta\psi & -\Delta\theta & \Delta\phi & 0 \end{bmatrix} . \end{aligned} \quad (5.28)$$

Using the definition of the matrix exponent and the skew symmetric property of $\boldsymbol{\Phi}_\Delta$, we

can write down the following closed-form solution:

$$\exp\left(-\frac{1}{2}\Phi_{\Delta}\right) = \mathbf{I} \cos(s) - \frac{1}{2}\Phi_{\Delta} \frac{\sin(s)}{s}, \quad (5.29)$$

where

$$\begin{aligned} s &= \frac{1}{2} \left\| \begin{bmatrix} \Delta\phi & \Delta\theta & \Delta\psi \end{bmatrix} \right\| \\ &= \frac{1}{2} \sqrt{(\Delta\phi)^2 + (\Delta\theta)^2 + (\Delta\psi)^2}. \end{aligned} \quad (5.30)$$

The proof for the closed-form solution presented in Equation 5.29 is given in Section B.4 of Appendix B. Equations 5.24 and 5.29 ensure (at least theoretically) that the updated quaternion \mathbf{e}_{k+1} has a unit norm. However, another small Lagrange multiplier term can be added to Equation 5.29 to further maintain numerical stability. The use of such a Lagrange multiplier term relaxes the requirement on the accuracy of computation of the trigonometric functions⁹, and allows for the use of truncated series approximations for $\cos(s)$ and $\sin(s)/s$. The resulting final solution for the time-update of the quaternion vector is given by

$$\mathbf{e}_{k+1} = \left[\mathbf{I} (\cos(s) + j \cdot dt \cdot \lambda) - \frac{1}{2}\Phi_{\Delta} \frac{\sin(s)}{s} \right] \mathbf{e}_k. \quad (5.31)$$

The position and velocity discrete-time updates are calculated by the following simple first-order Euler update

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \dot{\mathbf{p}}_k \cdot dt \quad (5.32)$$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \dot{\mathbf{v}}_k \cdot dt, \quad (5.33)$$

where $\dot{\mathbf{p}}_k$ and $\dot{\mathbf{v}}_k$ are calculated using Equations 5.11 and 5.12. More advanced integration techniques such as n th order *Runge-Kutta* methods can be used for this purpose as well, but we found the simpler approach to be more than sufficient (and accurate) for the calculation of the position and velocity updates.

⁹This can be important when the algorithm has to run on limited computational resources (i.e. embedded on-board CPU system) where trigonometric functions are often approximated via table lookups.

5.2.4 Sensor Models

The main sensor (observation) models are those for the *GPS* and the *barometric altimeter*. Unlike the full MIT-Draper-XCell-90 model, we do not include the IMU as part of the observation model. The reason for this was already mentioned when we introduced the kinematic model: Although the IMU is a sensor, for the kinematic model, it is used to summarize the forces and moments acting on the vehicle and hence is used directly as input to the kinematic differential equations. The measurement noise typically associated with observation sensors, now becomes *process noise* in the case of the IMU which directly drives the kinematic process model. This is convenient in that we can easily set the process noise variance levels according to the performance specifications of the IMU sensor, without having to resort to ad-hoc methods to determine the variance levels of “artificial” process noise, which is often the case in complex models.

We will now describe the nonlinear observation models used for the GPS and barometric altimeter:

GPS

A GPS receiver measures the position and velocity of the vehicle it is attached to, using precise timing signals from a constellation of orbital GPS satellites [17]. These measurements are normally¹⁰ relative to the inertial (NED) reference frame. The point of reference for the GPS’s measurements is the location of the GPS antenna, $\tilde{\mathbf{r}}_{gps}$, so one needs to compensate for the offset of this location from the vehicle’s center of gravity when building a mathematical observation model. Like all observation sensors, a GPS has an inherent accuracy determined by a variety of factors such as multi-path delay of the timing signal, signal strength, number of satellites in view and satellite location geometry [17]. Based on all of these factors, most GPS receivers calculate a “quality of service” type metric called the *position dilution of precision* (PDOP) signal that can be used as a scaling factor on the assumed accuracy of the resulting position and velocity estimates. For our purpose of building a robust kinematic model based state estimator, it is sufficient to lump all

¹⁰GPS receivers can also provide their measurements in the earth-centered-earth-fixed (ECEF) inertial reference frame.

of these GPS errors together into a single additive multivariate Gaussian noise term, \mathbf{n}_p , for the position measurements and \mathbf{n}_v for the velocity measurements. In order to model the time-varying accuracy of the GPS signals we then simply scale the variances of these Gaussian noise terms proportional to the PDOP signal.

Another effect that influences the GPS measurements is *latency*, i.e., the output provided at time t corresponds to a measurement made at time $t - t_{lag}$, where t_{lag} is the latency duration measured in seconds. Combining all of these effects results in the following GPS observation model:

$$\mathbf{p}_t^{GPS} = \mathbf{p}_{t-t_{lag}} + \mathbf{T}_{b2i}\tilde{\mathbf{r}}_{gps} + \mathbf{n}_{p_t} \quad (5.34)$$

$$\mathbf{v}_t^{GPS} = \mathbf{v}_{t-t_{lag}} + \mathbf{T}_{b2i}\boldsymbol{\omega}_{t-t_{lag}} \times \tilde{\mathbf{r}}_{gps} + \mathbf{n}_{v_t}, \quad (5.35)$$

where $\mathbf{p}_{t-t_{lag}}$ is the time-delayed 3D (NED) position of the helicopter as defined in Equations 5.9 and 5.10, $\mathbf{v}_{t-t_{lag}}$ is the time-delayed 3D (NED) velocity of the helicopter, \mathbf{T}_{b2i} is the body-to-NED direction cosine matrix as defined in Equation 5.16, $\tilde{\mathbf{r}}_{gps}$ is the location of the GPS antenna in the body frame, $\boldsymbol{\omega}_{t-t_{lag}}$ are the true rotational rates of the vehicle at time $t - t_{lag}$, and \mathbf{n}_{p_t} and \mathbf{n}_{v_t} are the Gaussian measurement noise terms as defined above. Here the noise terms are modeled as being time-dependent. This is due to the fact that the accuracy of observations vary over time according to the current PDOP value as discussed above.

Barometric Altimeter

Ambient air pressure provides an accurate source of sea-level altitude information out of the “ground effect”¹¹ zone. Important sources of error are sensor quantization and measurement noise. We used a high-end altimeter with $10^{-3}psi$ (0.6 meters) resolution. The measurement noise was assumed to be white Gaussian with a standard deviation of

¹¹Ground effect is due to the main rotor downwash being reflected upwards when the helicopter is close to the ground. This will have a nonlinear effect on the air pressure in the close vicinity of the vehicle.

6×10^{-4} psi. The observation model that incorporates these effects are:

$$\rho = \rho_0 \exp(G_{\Delta\rho} \cdot z_t) + n_{za} \quad (5.36)$$

$$\rho^q = \rho_0^q \text{floor}\left(\frac{\rho}{\rho_0^q}\right) \quad (5.37)$$

$$z_t^{alt} = -\frac{1}{G_{\Delta\rho}} \ln\left(\frac{\rho^q}{\rho_0}\right), \quad (5.38)$$

where ρ_0 is the nominal air pressure at sea-level, $G_{\Delta\rho}$ is the pressure decay rate with altitude constant (1.16603×10^{-4} psi/m), z_t is the current NED frame z-axis position of the vehicle, ρ_0^q is the air pressure quantization resolution of the altimeter (10^{-3} psi), z_t^{alt} is the altimeter output and $\text{floor}(\cdot)$ is the integer flooring function, e.g. $\text{floor}(3.7) = 3$.

5.2.5 Simulation Systems

The simulation environment comprises a family of four simulators to support rapid development of flight ready software. This environment was derived from a similar environment developed in MIT's Information Control Engineering program for the Aerial Robotics project (as mentioned in the introduction). These simulators include a Matlab implementation, a non-real-time, non-interactive (batch) simulator which runs on a Windows platform, an interactive, real-time "software-in-the-loop" simulator that includes simulations of sensor and servos, and a hardware-in-the-loop simulator. After control and estimation algorithms has undergone thorough testing in each of these environments, it is deemed flight ready. Common to all of these simulators is the high-fidelity computational flight dynamics model of the helicopter and its sensors (*MIT-Draper-XCell-90 model*: see Section 5.2.3), implemented in C.

The Matlab simulator is our preliminary design test bed. It allows control and estimation algorithms to be quickly implemented, compared and experimentally verified (tested). The Matlab simulator is linked via Matlab MEX functions to the C language high-fidelity helicopter model. The simulation is set up for either trajectory following, using a flight plan, or velocity tracking, following a set of pre-programmed stick inputs. All of the comparative experiments reported in this chapter were done within the Matlab simulator¹².

¹²For more detail on the other simulators, see [23, 22, 21, 20, 58].

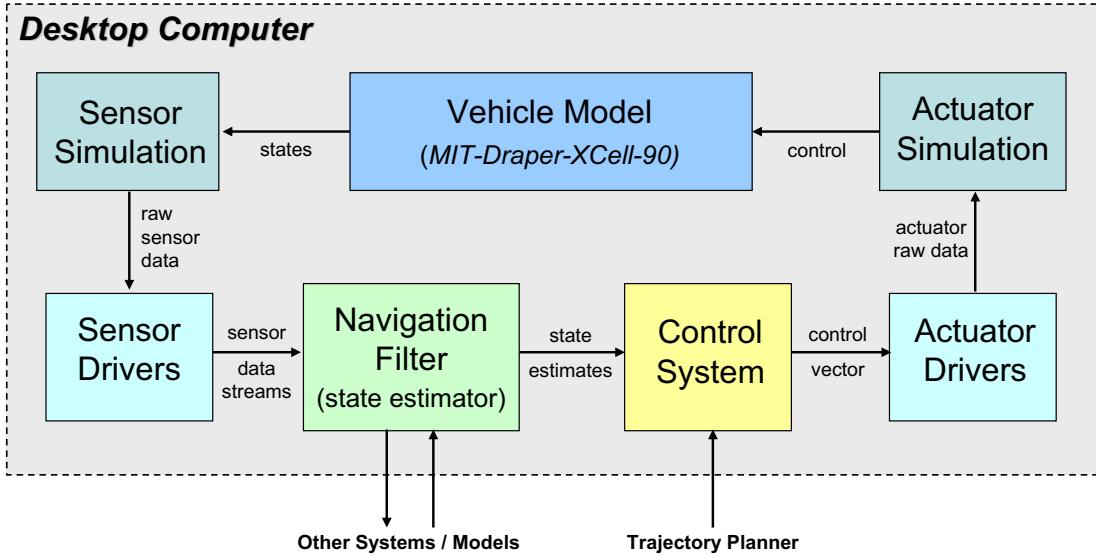


Figure 5.4: Schematic diagram of the software based UAV simulation system.

We chose to limit the experiments to this simulator for the following reasons: We wanted to make sure that measured differences in estimation accuracy was due to fundamental algorithmic (on a mathematical level) differences and not due to inefficient or incorrectly ported (from Matlab to C) implementations. Efficiently porting Matlab code to C in order to run in real-time under QNX in a real-time message passing fashion is not a trivial matter, and if done wrong, can easily affect the validity and accuracy of comparative (i.e. EKF vs. SPKF) experiments. Since we have, through the work reported in this chapter, verified the validity and superiority of our new SPKF based estimator, we are currently in the process of porting our code to C in order to test it on the hardware-in-the-loop simulator, as well as for real flight testing. However, since this second phase is still in progress it falls under the *future-work* category (see Chapter 7) and will not be covered in this thesis.

5.2.6 Control System

One of the main research efforts of our group, but not the focus of this thesis, is the development of a robust *state-dependent Riccati equation* controller utilizing nonlinear feedforward

neural-network compensation for the control of a small unmanned helicopter. This controller (which supersedes MIT's controller within our test platform) has reached a level of maturity such that it is now our controller of choice for all development work (in simulation and test-flights). We used this SDRE based controller for all of our estimation experiments comparing the performance of the SPKF to MIT's EKF based estimation subsystem. We now give only a brief overview of the SDRE control approach, for a more detailed discussion, see [21, 20, 30, 31].

SDRE Control: A brief overview

The SDRE approach [30] involves manipulating the vehicle dynamic equations

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \quad (5.39)$$

into a pseudo-linear form (SD-parameterization), in which system matrices are explicitly functions of the current state:

$$\mathbf{x}_{k+1} = \Phi(\mathbf{x}_k)\mathbf{x}_k + \Gamma(\mathbf{x}_k)\mathbf{u}_k . \quad (5.40)$$

A standard Riccati Equation can then be solved at each time step to design the state feedback control law on-line (a 50 Hz sampling rate is used in the flight experiments). Initial work on SDRE applications for helicopter control is described by Wan and Bogdanov[201, 22]. The SDRE regulator is specified as

$$\mathbf{u}_k = -\mathbf{R}^{-1}\Gamma^T(\mathbf{x}_k)\mathbf{P}(\mathbf{x}_k)(\mathbf{x}_k) \equiv -\mathbf{K}(\mathbf{x}_k)\mathbf{x}_k ,$$

where $\mathbf{P}(\mathbf{x}_k)$ is a steady state solution of the difference Riccati equation, obtained by solving the discrete-time algebraic Riccati equation (DARE)

$$\Phi^T [\mathbf{P} - \mathbf{P}\Gamma(\mathbf{R} + \Gamma^T\mathbf{P}\Gamma)^{-1}\Gamma^T\mathbf{P}] \Phi - \mathbf{P} + \mathbf{Q} = \mathbf{0} \quad (5.41)$$

using state-dependent matrices $\Phi(\mathbf{x}_k)$ and $\Gamma(\mathbf{x}_k)$, which are treated as being constant. For tracking problems with the desired state \mathbf{x}_k^{des} , the SDRE control can be implemented as

$$\mathbf{u}_k = -\mathbf{K}(\mathbf{x}_k)(\mathbf{x}_k - \mathbf{x}_k^{des}) \equiv -\mathbf{K}(\mathbf{x}_k)\mathbf{e}_k , \quad (5.42)$$

where the vector of controls are given by,

$$\mathbf{u}_k = \begin{bmatrix} u_{lon} & u_{lat} & u_{col} & u_{tcol} \end{bmatrix}^T , \quad (5.43)$$

corresponding to rotor blade pitch angles. The SDRE control generally exhibits greater stability and better performance than linear control laws, and thus is attractive in nonlinear control problems.

In order to calculate the SDRE based control law online, accurate estimates of the vehicle state and parameters are needed. As mentioned earlier, this requires the presence of an accurate state estimator to track the navigational state of the vehicle online, fusing the predictive output of the vehicle model with the noisy sensor observations in an probabilistically optimal fashion. We will next discuss how we designed and implemented a SPKF based state estimator for this purpose.

5.3 SPKF Based Estimation System

We replaced the MIT designed fine-tuned EKF with a SPKF based estimator that utilizes the nonlinear kinematic process model and observation models of the GPS and barometric altimeter as presented in Section 5.2.3. We chose to use a SR-CDKF SPKF formulation (Algorithm 11) due to its ease of implementation, intuitive choice of scaling parameters and numerical robustness.

A number of problem specific issues had to be dealt with in order to adapt the general SPKF framework to the UAV state estimation problem. The most important of these were.

- asynchronicity, differing sample-rates, and time-varying dimension of sensor observations

- filter initialization
- GPS latency
- quaternion unity norm constraint

We will next discuss how these issues were addressed within the SPKF algorithmic framework.

Measurement asynchronicity, varying sample rates and time-varying dimensions

Unlike well behaved synthetic laboratory experiments, the UAV avionics system (as well as the high-fidelity simulations thereof) operate on an asynchronous message passing principle. Although the main system (and filter) is clocked at the IMU rate (100Hz), the avionic sensors (GPS and altimeter) operate not only asynchronously from this clock, but their measurements are also provided at different sampling rates (10Hz and 20Hz respectively). This in turn implies that every filter cycle does not necessarily have both a time-update and a measurement-update step. Every time a new IMU measurement becomes available (roughly every 10ms), the filter undergoes a time-update using the IMU driven kinematic process model, but a measurement-update is only done if there are actually new sensor data available for processing. The flight-computer polls the navigational sensors (or their simulation equivalents) in order to determine if new data is available and if so, update a bank of sensor buffers with these new measurements. These measurements are accompanied by a unique time-stamp if new data was written into the sensor buffers.

Based on these time-stamps and prior knowledge of the different update rates of the sensors, the estimator system builds up an adaptive *event-map* of when to expect new data from the different sensors. This event-map (built up during the filter initialization stage), is important to deal with the GPS latency problem, which will be discussed in the next section.

Since the different sensors have different observation vector dimensions (GPS=6, altimeter=1) and operate at different rates, the SPKF observation model for any given measurement update must adapt to this time-varying total observation dimension: If both

sensors report new data, the effective observation model (and its measurement noise random variable) will be a concatenation of the individual sensor observation models, and sigma-points will be drawn from this augmented observation state (true 16D state + augmented noise state). It might also be the case though that only one sensor (either the GPS or altimeter) reports new data for a given measurement update, resulting in the observation model (and the related SPKF based measurement update sigma-point generation) reverting to the relevant single sensor model. This time-varying observation model requires careful book-keeping of sensor events in order to accurately adapt the filter to the time-varying nature of the sensor data stream.

Filter initialization

During the initialization of the navigation filter (state estimator), sensor data are observed and processed over a number of seconds while the UAV is sitting on the ground in a known position and attitude. This step is important to ensure robust performance of the filter during subsequent aggressive maneuvers.

The initial position estimate, $\hat{\mathbf{p}}_0 = \begin{bmatrix} \hat{x}_0 & \hat{y}_0 & \hat{z}_0 \end{bmatrix}^T$ in the NED frame is based solely on the averaged GPS position measurements for the first couple of seconds while the UAV is stationary. The initial NED frame velocity estimate, $\hat{\mathbf{v}}_0 = \begin{bmatrix} \hat{v}_{x_0} & \hat{v}_{y_0} & \hat{v}_{z_0} \end{bmatrix}^T$, is initialized to zero. The helicopter is orientated into a known yaw direction (either due north or due south) with the $x-y$ (horizontal) plane level (using two spirit levels) during the initialization phase. This allows the initial quaternion estimate, $\hat{\mathbf{e}}_0 = \begin{bmatrix} \hat{e}_{0_0} & \hat{e}_{1_0} & \hat{e}_{2_0} & \hat{e}_{3_0} \end{bmatrix}^T$, to be set to a known value such as $\hat{\mathbf{e}}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$. The averaged IMU accelerometer and gyro readings during the initialization period are used to initialize the estimates of the IMU bias variables. If the IMU is unbiased, the expected stationary accelerometer reading should be, $\mathbf{a} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T$ (where g is the local gravitational acceleration constant, i.e. $9.81m/s^2$) and the gyro rate readings (ignoring the rotation of the earth) should be zero. Any reading deviating from these expected values can then be used to initialize the bias estimates.

The final important house-keeping task performed during the initialization phase is to build up the “event map” for when to expect new data from the avionics sensors. This

is done by monitoring the time-stamps of the sensor stream buffers over a period of time (while the helicopter is stationary). A pre-specified number (≈ 10) of sensor cycles must be observed before the full event-map is robustly initialized. This in turn implies that the duration of this specific initialization phase is determined by the slowest sensor in the avionics system (in our case the GPS). As stated earlier, an accurate event map is also needed to accurately compensate for the inherent processing latency present in the GPS measurements.

Dealing with GPS latency

One of the big challenges in building a robust state estimator is dealing with the inherent measurement latency of the GPS sensor. As mentioned in Section 5.2.4, a GPS sensor has a finite processing delay between when a GPS satellite signal is received for processing and when the actual position and velocity measurement related to that signal becomes available. This implies that the current GPS reading actually corresponds to the position and velocity state of the vehicle at some point in the past. This time difference is called the measurement latency. For cheaper lower performance GPS systems this latency can be in the order of couple of seconds, causing serious problems when these measurements are fused (inside a Kalman filter) with the current prediction of the vehicle state. Figure 5.5 demonstrates this issue graphically for a linear DSSM given by

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{v}_k \quad (5.44)$$

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \mathbf{n}_k , \quad (5.45)$$

where $\mathbf{v}_k \sim N(\mathbf{0}, \mathbf{R}_v)$ and $\mathbf{n}_k \sim N(\mathbf{0}, \mathbf{R}_n)$. The state estimation filter, in general, receives measurements from a variety of sensors at each measurement-update step. Some measurements correspond to the system state at the current time, \mathbf{y}_k , given by Equation 5.45, while other latency-delayed measurements, \mathbf{y}_k^* , correspond to the system state at time $l = k - N$, i.e.,

$$\mathbf{y}_k^* = \mathbf{C}_l^* \mathbf{x}_l + \mathbf{n}_k^* ,$$

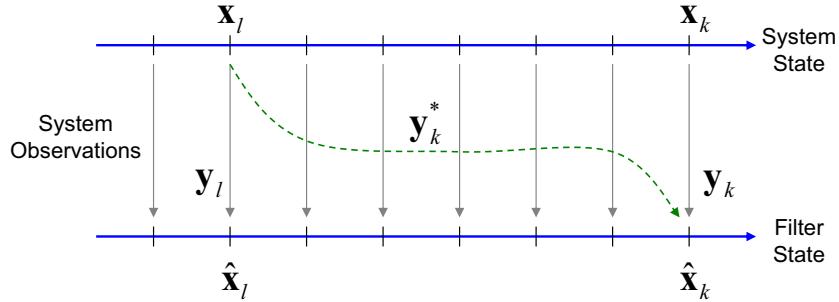


Figure 5.5: System with a delayed measurement due to sensor latency: At time k the state estimation filter receives two sets of sensor measurements from the system: A normal measurement \mathbf{y}_k corresponding to the system state at time k , as well as a delayed measurement \mathbf{y}_k^* corresponding to the system state at time $l = k - N$, where N is the sensor latency (measured in sample periods).

where N is the sensor latency measured in sample periods, \mathbf{C}_l^* is the measurement sensitivity matrix and \mathbf{n}_k^* is the observation noise for the delayed measurement with $\mathbf{n}_k^* \sim N(\mathbf{0}, \mathbf{R}_n^*)$. The challenge now is this: how do we optimally fuse these different sets of measurements with the current estimate of the system state?

A number of different solutions to the sensor latency problem has been suggested in the literature [2, 111, 8] for application to *linear* DSSMs through the use of modified linear Kalman filters. These solutions range from “*exact but impractical*” to “*trivial but highly inaccurate*” with a number of approximate solutions in between. We will now briefly review these different approaches.

The simplest to implement, but also the most inaccurate, solution is to simply ignore the fact that the sensor measurement is lagged. A normal Kalman measurement update is then performed to fuse the delayed sensor measurement with the current state estimate. This is the option that was used for the GPS sensor fusion in MIT’s EKF based state estimator [58]. Their rationale was that the 50ms latency of the Ashtech G12 GPS receiver was short enough to not cause significant errors if ignored [59]. Even though this assumption is probably reasonable for the specific time-delay at hand, we felt¹³ that valuable information in the measurements were still being discarded using this approach and that compensating for the latency will in fact result in a significant increase in estimation performance.

¹³This intuition is experimentally confirmed in Section 5.4.

Another approach used to compensate for measurement latency is to simply recalculate the complete time-trajectory of the filter through the delay period, when a lagged measurement is received. This requires all of the observations (both lagged and non-lagged) as well as the system state estimates to be saved for the complete latency duration, i.e. $\tilde{k} \in [k - N, k - N + 1, \dots, k - 1, k]$. Not only does this incur a large storage cost for any significant sensor latency, but the computational burden also increases rapidly as the recalculation period becomes longer. This has serious implications if the filter rate is such that new state estimates have to be calculated within a short period of time. The only advantage of this approach is that the resulting state estimates will be exact, i.e., no approximations with regard to the sensor latency have been made. Unfortunately, due to the large storage and computational cost, this method is almost never employed.

Another typical approximate solution used to deal with this latency problem is to store the value of the state estimate corresponding to the latency-lagged sensor measurement, i.e., $\hat{\mathbf{x}}_{k-N}$, in a buffer. When the lagged measurement \mathbf{y}_k^* then eventually becomes available at time k , the “prediction of the observation”, $\hat{\mathbf{y}}_k^{*-}$, is calculated using the correct lagged state estimate $\hat{\mathbf{x}}_{k-N}$, i.e.

$$\hat{\mathbf{y}}_k^{*-} = \mathbf{C}_{k-N}^* \hat{\mathbf{x}}_{k-N} . \quad (5.46)$$

The innovation based on this lagged prediction, $\mathbf{y}_k^* - \hat{\mathbf{y}}_k^{*-}$, is then fused using the standard Kalman measurement-update with the *current* predicted state, i.e.

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K} (\mathbf{y}_k^* - \hat{\mathbf{y}}_k^{*-}) . \quad (5.47)$$

Since the correct innovation is fused with the wrong state prediction, this method although better than the first approach, is still sub-optimal.

In [2], a method is derived where it suffices to calculate a correction term which is then added to the filter estimates when the latency-delayed measurements arrive. Referring to the standard Kalman filter equations (Algorithm 19) and Figure 5.5, the measurement \mathbf{y}_l should be fused at time $l = k - N$, causing a correction in the state estimate $\hat{\mathbf{x}}_l^-$ and a decrease in the state covariance $\mathbf{P}_{\mathbf{x}_l}^-$. As the Kalman gain is a function of this updated state covariance, the measurements occurring after this time ($k > l$) will all

be fused differently than if the measurement update for \mathbf{y}_k^* is omitted. If therefore the measurement \mathbf{y}_l is delayed by N samples (resulting in \mathbf{y}_k^*) and fused at time k , the data update should reflect the fact that the N data updates from time l to k , and therefore the state and covariance estimates, have all been affected by the delay in a complex manner [111]. Equations that account for this when fusing \mathbf{y}_k^* at time k were derived in [2] but are of such complexity that they are practically infeasible¹⁴ for most applications. This approach was reformulated in [111] into a practically implementable method (called *Larsen's method*) which does however require (at time l) the measurement sensitivity matrix \mathbf{C}_l^* and the observation noise covariance matrix \mathbf{R}_n^* to be known at time l (which is often the case). If these requirements are met, the filter covariance should be updated at time l as if the measurement \mathbf{y}_k^* is already available. This leads the measurements in the delay period to be fused as if \mathbf{y}_k^* had been fused at time l . At time k , when \mathbf{y}_k^* actually becomes available, incorporating the measurement (\mathbf{y}_k^*) correctly is then greatly simplified, by adding the following correction term after the non-lagged observation \mathbf{y}_k has been fused:

$$\delta\hat{\mathbf{x}}_k = \mathbf{M}_* \mathbf{K}_l (\mathbf{y}_k^* - \mathbf{C}_l^* \hat{\mathbf{x}}_l) \quad (5.48)$$

If the latency delay is zero, \mathbf{M}_* is the identity matrix. For $N > 0$, \mathbf{M}_* is given by:

$$\mathbf{M}_* = \prod_{i=0}^{N-1} \left(\mathbf{I} - \mathbf{K}'_{k-i} \mathbf{C}_{k-i} \right) \mathbf{A}_{k-i-1}, \quad (5.49)$$

where \mathbf{K}' signifies Kalman gain matrices that have been calculated based on a covariance matrix updated at time l with the covariance of the delayed measurement¹⁵. This implies that the covariance estimates of the filter will be wrong in a period of N samples (before the delayed measurement arrives), causing normal non-delayed measurements during this period to be fused sub-optimally. However, after the correction term (Equation 5.48) is added, the filter state and covariance will once again be optimal. Take note that Equations 5.49 and 5.48 assumes a linear DSSM. This method can be adapted to nonlinear

¹⁴In fact, the computational complexity is comparable to recalculating the complete Kalman filter through the complete delay of N_{lag} samples [111], which is equivalent to the previously discussed exact method for latency compensation.

¹⁵Since the covariance update only depends on \mathbf{C}_l^* and \mathbf{R}_n^* and not \mathbf{y}_k^* , it can be precalculated.

models using either EKF or SPKF based linearization, i.e., by replacing the sensitivity matrices \mathbf{C} and \mathbf{A} with either the process and observation model Jacobians (EKF), or the WSLR calculated “average” Jacobians (SPKF)¹⁶. See [8] for an application of this approach to a nonlinear robot localization problem using an EKF.

Larsen [111] extended the previous approach further to a method that provides optimally fused estimates not only at the instances when the delayed measurements arrive, but also during the interim period between these updates. This is achieved by running a second parallel Kalman filter that generates optimal estimates in the interim (between delayed measurements) period: At time l the first filter is updated according to Larsen’s first method (shown above) incorporating the covariance of the not-yet-received delayed measurement \mathbf{y}_k^* . This filter, will generate non-optimal estimates until the delayed measurement is actually received, but it will build up the correct terms needed for an optimal measurement update at that time. During this interim period the second filter, which was not pre-updated at time l , will generate optimal estimates when fusing non-delayed measurements. At time k , the correctly fused optimal estimate of the first filter (which incorporated the delayed measurement) and its covariance is used to reinitialize the second filter. Using this approach optimal estimates are available at all times. The downside is that two Kalman filters need to be run in parallel which will double the computational complexity.

Summary: In this section we discussed a number of existing techniques for Kalman filter based latency delayed sensor fusion. Of these approaches, *Larsen’s modified two-filter method* is the only approach that is both computationally practical and produces optimal states estimates at all time. The method is, however, only exact for linear systems. For nonlinear systems, the recursive calculation of the corrections terms (Equations 5.49 and 5.48) must be approximated through the use of linearized system matrices. If the linearization errors are severe, it is expected that large errors may accumulate over time resulting in sub-optimal corrections when the lagged measurements finally arrive.

Clearly, a new method for optimally fusing latency delayed sensor data in *general nonlinear* systems is needed.

¹⁶See Section 4.5.4 on how this WSLR averaged Jacobian can be calculated for the SPKF.

5.3.1 SPKF based Time Delayed Sensor Fusion

Based on discussions with, and a preliminary idea communicated by Julier [98], we implemented a novel SPKF based latency delayed sensor fusion technique [195] that has not yet been published elsewhere. We will next present this new method after “refreshing” some required background concepts from Chapter 2 and Appendix A, specifically as they apply to time delayed measurements.

Time Delayed Measurements Revisited: A Kalman Optimal Perspective

Kalman’s original optimal update for fusing a new observation with the current estimate of the system state is given by Equations A.1 and A.2 in Appendix A. These can easily be rewritten for the case where we want to optimally fuse the current state estimate at time k with a *time delayed* measurement $\mathbf{y}_k^* = \mathbf{y}_{k-n}$, where n is the number of samples by which the measurement is delayed (the latency period). Specifically, Equation A.1 becomes

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \tilde{\mathbf{y}}_{k-n} , \quad (5.50)$$

where the time delayed innovation is given by

$$\tilde{\mathbf{y}}_{k-n} = \mathbf{y}_k^* - \hat{\mathbf{y}}_k^* \quad (5.51)$$

$$= \mathbf{y}_{k-n} - \hat{\mathbf{y}}_{k-n} , \quad (5.52)$$

and \mathbf{K}_k is the Kalman gain term. The optimal form of the Kalman gain can again be found using the exact same derivation as presented in Section A.2 of Appendix A. However, the innovation and cross covariance terms are calculated between time step k and time step $l = k - n$. This results in the following form for the Kalman gain:

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_{k-n}} (\mathbf{P}_{\tilde{\mathbf{y}}_{k-n} \tilde{\mathbf{y}}_{k-n}})^{-1} . \quad (5.53)$$

The second term is the inverse of the innovation covariance which can easily be calculated at time $k - n$. For the EKF its value is given by

$$\mathbf{P}_{\tilde{\mathbf{y}}_{k-n} \tilde{\mathbf{y}}_{k-n}} = \mathbf{H}_{\mathbf{x}_{k-n}} \mathbf{P}_{\mathbf{x}_{k-n}}^- \mathbf{H}_{\mathbf{x}_{k-n}}^T + \mathbf{R}_n , \quad (5.54)$$

where $\mathbf{H}_{\mathbf{x}_{k-n}}$ is the observation model Jacobian evaluated at $\mathbf{x} = \hat{\mathbf{x}}_{k-n}^-$, $\mathbf{P}_{\mathbf{x}_{k-n}}^-$ is the time-updated state estimate covariance at time $k-n$ and \mathbf{R}_n is the measurement noise covariance (we assume the measurement noise is generated by a stationary process). However, the first term of Equation 5.53,

$$\mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_{k-n}} = E [(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{y}_{k-n} - \hat{\mathbf{y}}_{k-n}^-)^T] , \quad (5.55)$$

is not so easily calculated. For the EKF (under the assumption that the observation noise sequence is independent), this term can be approximated as

$$\mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_{k-n}} \approx E [(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_{k-n} - \hat{\mathbf{x}}_{k-n}^-)^T] \mathbf{H}_{\mathbf{x}_{k-n}}^T \quad (5.56)$$

The expectation in the equation above can be expanded and then approximated as [98],

$$E [(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_{k-n} - \hat{\mathbf{x}}_{k-n}^-)^T] = \mathbf{P}_{\mathbf{x}_k \mathbf{x}_{k-n}}^- \quad (5.57)$$

$$\approx \Phi_{k,k-n} \mathbf{P}_{\mathbf{x}_{k-n}}^- \quad (5.58)$$

where $\Phi_{k,k-n}$ is the approximate cumulative transformation matrix. Its value between any successive time step is

$$\Phi_{i,i-1} = (\mathbf{I} - \mathbf{K}_i \mathbf{H}_{\mathbf{x}_i}) \mathbf{F}_{\mathbf{x}_{i-1}} , \quad (5.59)$$

where $\mathbf{F}_{\mathbf{x}_{i-1}}$ is the process model Jacobian. Using the fact that $\Phi_{i,i} = \mathbf{I}$, this recursion relationship gives the same result as Larsen (Equations 5.48 and 5.49) [111, 98].

This section provided a derivation of the time delayed fusion method by approaching the problem from the “other way around” than Larsen. Rather than thinking about extrapolating a measurement forward in time, you can also view it as taking a cross correlation backwards through time. This idea will now be used to derive a new and better method to fuse the time-delayed measurement directly.

An Alternative Way to Maintain the Cross-Covariance

As pointed out above, the crucial part of correctly fusing the time delayed measurement is maintaining the cross-covariance term $\mathbf{P}_{\mathbf{x}_k \mathbf{x}_{k-n}}^-$ during the latency period. One approximate method to do this was shown above, based on recursively applying the linearized system model to propagate the needed covariance term. We will now present an alternative method to maintain the cross-covariance term through augmentation of the system state and redefinition of the process model.

First we define the augmented state variable,

$$\mathbf{x}_k^a = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_l \end{bmatrix}, \quad (5.60)$$

with process model

$$\mathbf{x}_{k+1}^a = \check{\mathbf{f}}(\mathbf{x}_k^a, \mathbf{v}_k) \quad (5.61)$$

$$= \begin{bmatrix} \mathbf{f}(\mathbf{x}_k, \mathbf{v}_k) \\ \mathbf{x}_l \end{bmatrix}, \quad (5.62)$$

where $\mathbf{f}(\cdot)$ is the original process model, \mathbf{x}_l is the original system state at time l with $l \leq k$, \mathbf{x}_k is the original system state at time k and \mathbf{v}_k is the original process noise term.

In other words, the augmented system state at time k (Equation 5.60) consists of the true system state at time k concatenated to the state of the system at some prior point in time, $l \leq k$. The augmented process model (Equations 5.61 and 5.62) will then update the first component of the augmented state using the original process model while keeping the second component, \mathbf{x}_l , constant. This approach will thus maintain the value of the system state at some prior point in time and update the current state of the system.

Lets now assume that at time $k-n$ the lagged sensor makes it's physical measurement, but will only output the result of this measurement N_{lag} samples later. So, at time $k-n$ the augmented system state is initialized with the current state of the system,

$$\hat{\mathbf{x}}_{k-n}^a = \begin{bmatrix} \hat{\mathbf{x}}_{k-n} \\ \hat{\mathbf{x}}_{k-n} \end{bmatrix}, \quad (5.63)$$

with covariance

$$\mathbf{P}_{\mathbf{x}_{k-n}^a} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_{k-n}} & \mathbf{P}_{\mathbf{x}_{k-n}} \\ \mathbf{P}_{\mathbf{x}_{k-n}} & \mathbf{P}_{\mathbf{x}_{k-n}} \end{bmatrix}. \quad (5.64)$$

Propagating the augmented state forward using the Kalman filter (SPKF) time-update (assuming that the process noise sequence is white, Gaussian and zero-mean), results in the following predicted state and covariance estimates:

$$\hat{\mathbf{x}}_{k-n+1}^{a-} = \begin{bmatrix} \hat{\mathbf{x}}_{k-n+1}^- \\ \hat{\mathbf{x}}_{k-n} \end{bmatrix} \quad (5.65)$$

$$\mathbf{P}_{\mathbf{x}_{k-n+1}^a}^- = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_{k-n+1}}^- & \mathbf{P}_{\mathbf{x}_{k-n+1}\mathbf{x}_{k-n}}^- \\ \mathbf{P}_{\mathbf{x}_{k-n}\mathbf{x}_{k-n+1}}^- & \mathbf{P}_{\mathbf{x}_{k-n}}^- \end{bmatrix}. \quad (5.66)$$

One remaining issue to be dealt with is how do we update the augmented state during the latency period when a normal (non time delayed) measurement arrives? For example: The system is still waiting for the lagged sensor, say a GPS, to output its measurement of the system state at time $k - n$, but it just received a non-lagged measurement from another sensor, say the altimeter, which must now be fused with the current estimate of the system state. How can this be accomplished within the augmented framework we're currently discussing?

One solution to this problem is through the use of the *Schmidt-Kalman filter* [172]. This alternate formulation of the Kalman filter has the property that certain states are marked as “ancillary” whose values are not updated. It uses the *Joseph form*¹⁷ [62] of the measurement update equations and a modified Kalman gain equation given by

$$\mathbf{K}_k = \mathbf{M} \mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_{k-n}} (\mathbf{P}_{\tilde{\mathbf{y}}_{k-n} \tilde{\mathbf{y}}_{k-n}})^{-1}, \quad (5.67)$$

where \mathbf{M} is the indicator matrix which indicates which of the states must be updated and which are “ancillary”. It is block diagonal with 1s in the diagonals for states which are updated and 0s for states which are not to be updated. For our specific case at hand, \mathbf{M}

¹⁷Both the Schmidt-Kalman filter and the Joseph form measurement update is discussed in Section A.5 of Appendix A.

is given by:

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (5.68)$$

Substituting Equations 5.67 and 5.68 back into the Schmidt-Kalman measurement update equations result in the following measurement updated system state and covariance:

$$\begin{aligned} \hat{\mathbf{x}}_{k-n+1}^a &= \begin{bmatrix} \hat{\mathbf{x}}_{k-n+1} \\ \hat{\mathbf{x}}_{k-n} \end{bmatrix} \\ \mathbf{P}_{\mathbf{x}_{k-n+1}^a} &= \begin{bmatrix} \mathbf{P}_{\mathbf{x}_{k-n+1}} & \mathbf{P}_{\mathbf{x}_{k-n+1}\mathbf{x}_{k-n}} \\ \mathbf{P}_{\mathbf{x}_{k-n}\mathbf{x}_{k-n+1}} & \mathbf{P}_{\mathbf{x}_{k-n}} \end{bmatrix}. \end{aligned}$$

In other words, the bottom right sub-block maintains the original covariance matrix, the top left sub-block is the covariance of the state as would be calculated by a standard (non latency compensated) Kalman filter, and the off diagonal sub-blocks are the important *cross covariance* estimates, needed to fuse the time-delayed measurement through implicit use of Equations 5.50, 5.51, 5.52 and 5.53.

The interesting thing is: *What happens when you don't use the Schmidt form of the measurement update?* In other words, perform a normal Kalman measurement update without the use of the indicator matrix \mathbf{M} . The result of this will be that the original state estimate, \mathbf{x}_l in the augmented state vector (Equation 5.60), will no longer stay constant throughout the latency period due to the measurement update performed on it. It will now be measurement-updated based on subsequent information that has been observed. This deviates from Larsen's method [111] and should give better results. In fact, what will happen is that the original state estimate \mathbf{x}_l (from which the delayed sensor measurement is derived), will be *smoothed* during the latency period using future non-delayed observations. We use the term "smoothing" here in a more general context than the typical algorithmic implementation of a smoothing filter as presented in Section 3.6.3. That is, we simply imply that past estimates of the system state can somehow be improved based on subsequent future observations of measurement data. This is a well known result from signal-processing filter theory [75]. Since we are waiting for the lagged sensor anyway during the latency period, we might just as well use data received in the interim period to

improve our estimate of the system state that the sensor will eventually be reporting on. When the delayed measurement then finally arrives and the innovation is calculated using Equations 5.51 and 5.52, the prediction of the delayed observation will be more accurate since it will now be calculated using the smoothed lagged state estimate. As long as this “smoothing” operation is done correctly, which will be the case for this unified Kalman framework, we will make better use of *all* of the information available to us, resulting in improved filter performance.

SPKF Implementation of New “Smoothed Delayed Sensor Fusion” Method

The approach proposed in the previous section can be directly extended to the SPKF through the use of the sigma-point approach: We simply expand our normal state vector (and its covariance) to the augmented form of Equations 5.60, 5.63 and 5.64. Sigma-points are then drawn from the enlarged state and its covariance using the standard SPKF formulation. These points are then propagated through the augmented state process model and updated using the standard SPKF equations and correctly constructed observation models that reflect the time varying nature of observation vectors (see Section 5.3). More specifically, using the augmented state vector $\mathbf{x}_k^a = \begin{bmatrix} \mathbf{x}_k^T & \mathbf{x}_l^T \end{bmatrix}^T$, the observation model for a non-time delayed measurement will be of the following form:

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k^a, \mathbf{n}_k) = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_{1,k}) , \quad (5.69)$$

where \mathbf{x}_k is the first (non-delayed) half of the augmented state vector and $\mathbf{n}_{1,k}$ is the observation noise term affecting the non-time delayed sensor measurement. Likewise, the observation model for the latency delayed sensor measurement is given by¹⁸:

$$\mathbf{y}_k^* = \mathbf{h}^*(\mathbf{x}_k^a, \mathbf{n}_k) = \mathbf{h}^*(\mathbf{x}_l, \mathbf{n}_{2,k}) , \quad (5.70)$$

where \mathbf{x}_l is the second (smoothed) half of the augmented state vector corresponding to time $l = k^* - N_{lag}$, k^* is the discrete time instance when a delayed measurement is received, N_{lag}

¹⁸Note, the “*” notation is used here simply as a function or variable identifier and does not imply the typical “complex-conjugate” mathematical operation.

is the latency period measured in samples and $\mathbf{n}_{2,k}$ is the observation noise term affecting the time delayed sensor measurement. If both delayed and non-delayed measurements are received at the same time the observation model is simply a concatenation of the two models shown above, with the observation and noise vector dimensions adjusted accordingly.

In order to accurately re-initialize the lagged (second) component of the augmented state vector every N_{lag} samples before the latency delayed observation will be received, the *sensor event-map* (which was built up during the initialization phase) is used. See Section 5.3.

The advantage of the SPKF formulation of the “smoothed delayed sensor fusion” approach is that the latency delayed measurements are incorporated using the exact same algorithmic framework used to recursively estimate the normal states. In other words, the delayed fusion benefits in the same way from the correct second-order accuracy of the sigma-point approach: not only for the propagation and calculation of the cross covariance terms but also for the correct treatment of the possibly nonlinear cumulative effect of the injected process noise. The disadvantage of this method, compared to some of the simpler approximate methods, is that the state dimension (and hence the number of sigma-points) doubles during the sensor latency period, resulting in an increase in computational complexity. However, in the next section we will show how the increased estimation accuracy benefit of this approach outweighs the moderate increase in computational cost.

5.3.2 SPKF based quaternion unity norm constraint enforcement

As mentioned earlier, the quaternion based attitude formulation has many benefits over an Euler-angle based formulation. These benefits do however come with the caveat that the unity norm of the estimated quaternion vector has to be maintained during the estimation process. We already discussed the Lagrange multiplier based direct approach we employ inside the kinematic model to enforce this constraint. There is however another elegant method available to enforce this unity constraint that is ideally suited to the SPKF framework.

We first define the following pseudo-observation of the quaternion sub-vector of the full

state vector

$$z_{\mathbf{e}} = \|\mathbf{e}\|^2 \quad (5.71)$$

$$= \mathbf{e}^T \mathbf{e} \quad (5.72)$$

$$= e_0^2 + e_1^2 + e_2^2 + e_3^2, \quad (5.73)$$

i.e., we directly observe the squared norm of the quaternion vector. This is a pseudo-observation however, since we don't have an actual sensor that can measure the norm of the true attitude quaternion. We do however know that the true attitude quaternion should always have a unity norm so we can simply force the observation at time k to be equal to one, i.e.,

$$z_{\mathbf{e},k} \equiv 1. \quad (5.74)$$

We can now calculate the difference (innovation signal) between the assumed known (observed) quaternion norm and that of the prediction of the observation based on the current estimate of the quaternion, i.e.,

$$\tilde{z}_{\mathbf{e},k} = z_{\mathbf{e},k} - \hat{z}_{\mathbf{e},k}^- \quad (5.75)$$

$$= 1 - E \left[\|\hat{\mathbf{e}}_k^-\|^2 \right] \quad (5.76)$$

$$= 1 - E \left[\hat{e}_{0,k}^2 + \hat{e}_{1,k}^2 + \hat{e}_{2,k}^2 + \hat{e}_{3,k}^2 \right], \quad (5.77)$$

and update the vehicle state using a standard Kalman framework measurement update, i.e.,

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_{\mathbf{e}} \left(z_{\mathbf{e},k} - \hat{z}_{\mathbf{e},k}^- \right). \quad (5.78)$$

This step can easily be incorporated into the the existing SPKF framework by simply augmenting the observation model with Equation 5.71 and concatenating the assumed unity observation to the end of the true observation vector. It is also convenient at this point to introduce a synthetic additive observation noise term, $n_{\mathbf{e}}$, into the pseudo-observation equation for the quaternion norm,

$$z_{\mathbf{e}} = \|\mathbf{e}\|^2 + n_{\mathbf{e}}, \quad (5.79)$$

where we model $n_{\mathbf{e}}$ as a zero-mean Gaussian random variable, i.e., $n_{\mathbf{e}} \sim \mathcal{N}(n_{\mathbf{e}}; 0, \sigma_{n_{\mathbf{e}}}^2)$, where $\sigma_{n_{\mathbf{e}}}^2$ is the variance. The magnitude of the variance of this synthetic noise term will directly affect the “weighting” that the quaternion norm unity constraint receives in relation to the other observations. If the variance is set to zero, the SPKF will put a large weight (importance) on ensuring that the constraint is met, possibly at the detriment of other components in the state vector estimate. On the other hand, if the variance is set very high, the filter will largely discount the pseudo observation, resulting in poor quaternion regulation. The exact setting of this parameter should thus be determined in relation to the variances of the other observation noise terms in order to find an acceptable trade-off between the two extreme conditions.

Due to the quadratic form of the pseudo-observation nonlinearity, the SPKF (in contrast to the EKF) is ideally suited for this specific method of quaternion norm regulation. As discussed in Section 4.3 and demonstrated in Section 4.3.3, the SPKF generates exact results for quadratic nonlinearities, whereas the EKF results for the same nonlinearity are highly biased and inconsistent. Furthermore, the right hand term of Equation 5.77, $E[\hat{e}_{0,k}^2 + \hat{e}_{1,k}^2 + \hat{e}_{2,k}^2 + \hat{e}_{3,k}^2]$, can be calculated accurately at almost no extra computational cost by noticing that

$$\begin{aligned} E[\hat{e}_{0,k}^2 + \hat{e}_{1,k}^2 + \hat{e}_{2,k}^2 + \hat{e}_{3,k}^2] &= E[\hat{e}_{0,k}^2] + E[\hat{e}_{1,k}^2] + E[\hat{e}_{2,k}^2] + E[\hat{e}_{3,k}^2] \\ &= \hat{\sigma}_{e_0}^2 + \hat{\sigma}_{e_1}^2 + \hat{\sigma}_{e_2}^2 + \hat{\sigma}_{e_3}^2 \end{aligned} \quad (5.80)$$

$$= \text{trace}\{\mathbf{P}_{\mathbf{e}_k}\}, \quad (5.81)$$

where $\hat{\sigma}_{e_i}^2$ is the variance of the i th component of the quaternion vector and $\mathbf{P}_{\mathbf{e}_k}$ is the sub-block of the estimated state covariance matrix that relates to the quaternion sub-vector. In other words

$$\mathbf{P}_{\mathbf{e}_k} = E[(\mathbf{e}_k - \hat{\mathbf{e}}_k)(\mathbf{e}_k - \hat{\mathbf{e}}_k)^T], \quad (5.82)$$

where the indexes of the sub-vector estimate (and its covariance) relative to the full state vector estimate and covariance are given by Equations 5.9 and 5.10. Based on this result, the quaternion norm pseudo-observation augmentation of the observation model can be implemented without the need to propagate any extra sigma-points for the calculation of

the predicted observation and its covariance. These values already exist encapsulated and pre-calculated within the prediction of the state and its covariance. This in turn implies that this SPKF-centric quaternion norm regularization method can be implemented at almost no extra computational cost.

5.4 State Estimation Experimental Results

In this section we report on numerous state estimation experiments performed using the above described kinematic model based SPKF state estimator. All of the experiments were performed using the high-fidelity MIT-Draper-XCell-90 model based UAV simulator platform. This allowed for a repeatable controlled experimental environment. For all the experiments we compared the performance of our new proposed SPKF approach to the performance of the built-in fine-tuned EKF based estimator as provided by MIT.

5.4.1 Experiment SE0: Determining effect of IMU offset term in vehicle dynamics

In Section 5.2.3 (Equation 5.12) we introduced a component of the vehicle kinematic model, the rotation-moment-linear-acceleration coupling term $\mathbf{a}_{\tilde{\mathbf{r}}_{imu}}$, which is used to couple the effect of the vehicle's rotation into the linear acceleration measurement as provided by the IMU accelerometers. This extra additive (measured) acceleration term is due to the IMU not being located at the center of gravity of the vehicle and is given by

$$\mathbf{a}_{\tilde{\mathbf{r}}_{imu}} = \dot{\boldsymbol{\omega}} \times \tilde{\mathbf{r}}_{imu} + \bar{\boldsymbol{\omega}} \times (\bar{\boldsymbol{\omega}} \times \tilde{\mathbf{r}}_{imu}) , \quad (5.83)$$

where $\tilde{\mathbf{r}}_{imu}$ is the location vector of the IMU in the body frame (with origin at the center of gravity) and \times is the normal vector-cross-product. This correction term is computationally costly to calculate and also depends on a variable that is not part of the kinematic-model state vector, $\dot{\boldsymbol{\omega}}$, (the rotational acceleration), which will have to be approximated by a pseudo-derivative of the gyro-rates. Such derivatives are typically quite noisy and not very accurate. For this reason, the correction term is often ignored if the IMU offset is not too far away from the center of gravity (c.g.) of the vehicle. For our specific situation, the IMU is located at a relative offset of $\tilde{\mathbf{r}}_{imu} = [0.100 \ 0.025 \ 0.120]^T$ from the helicopter's c.g. We performed a number of experiments using the high-fidelity simulator (See Section 5.2.5) to determine if we could safely drop the cross-coupling term from the kinematic equations without paying to large an accuracy penalty. Since the simulator is based on the high-fidelity MIT-Draper-XCell-90 model, it does include the affect of IMU offset on

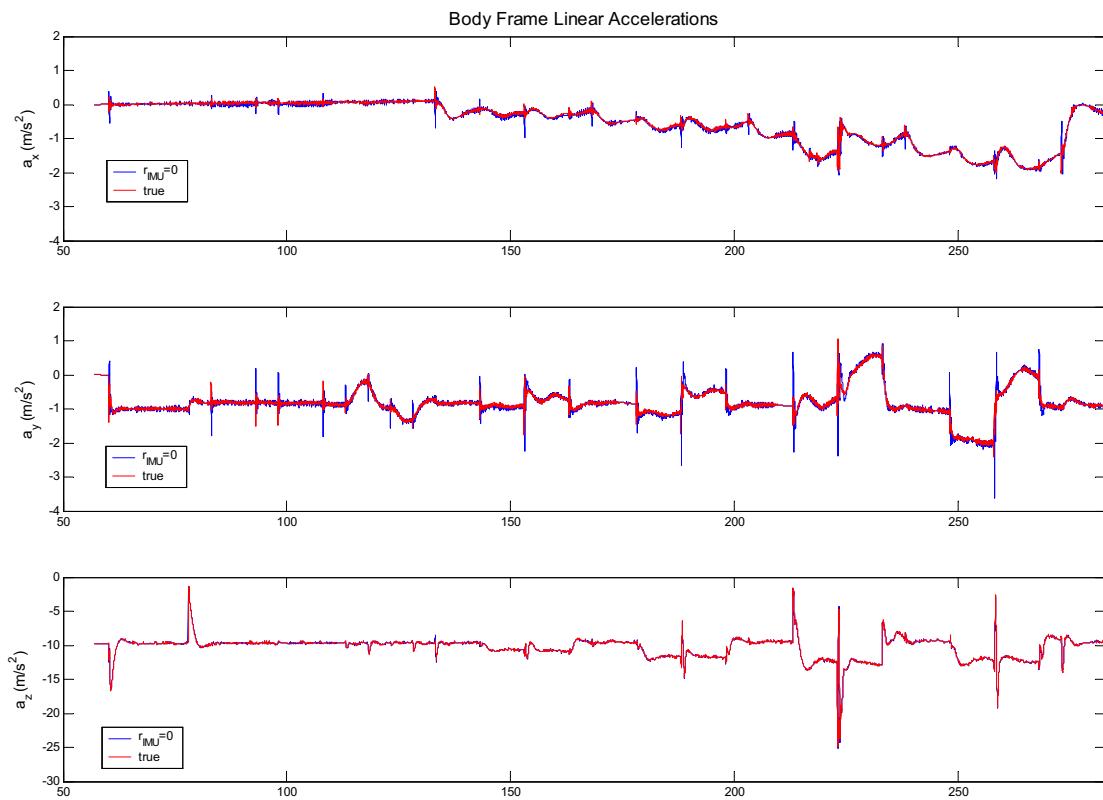


Figure 5.6: IMU offset coupling term impact determination experiment: The red trace in all the plots indicate the full (true) IMU accelerometer measurements for all three body axes. These measurements include the effect of the IMU-offset coupling term. The blue traces in the plot indicate accelerometer readings if the IMU-offset term is ignored. The average normalized MSE for the discrepancy error is 0.0372 (3.7%) with component contributions of $nMSE_{a_x} = 0.0197$, $nMSE_{a_y} = 0.0896$ and $nMSE_{a_z} = 0.0023$.

the accelerometer readings, allowing for an accurate comparison. We flew (in simulation) an aggressive maneuver while recording the IMU accelerometer readings (a_x, a_y, a_z), both with and without the effect of IMU offset. Figure 5.6 shows the result of this experiment. The red line in all the plots indicate the full (true) accelerometer measurements for all three body axes. These measurements include the effect of the IMU-offset coupling term. The blue lines in the plot indicate accelerometer readings if the IMU-offset term is ignored. Clearly there are some differences between the two sets of traces, but they do not dominate. The average normalized MSE for the discrepancy error is 0.0372 (3.7%) with component contributions of $nMSE_{a_x} = 0.0197$, $nMSE_{a_y} = 0.0896$ and $nMSE_{a_z} = 0.0023$. We furthermore determined that the difference in state estimation error between the two runs is less than 1%. For this reason it was decided to ignore the IMU-offset coupling term in the kinematic model of our SPKF based implementation, resulting in the use of the following equation for the velocity update: $\dot{\mathbf{v}} = \mathbf{T}_{b2i}\bar{\mathbf{a}} + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T g$.

5.4.2 Experiment SE1: EKF vs. SPKF (without latency compensation) vs. SPKF (with latency compensation)

The first experiment performed was used to determine if the proposed GPS latency compensation technique (see Section 5.3.1) offers a significant increase in performance compared to simply using a normal SPKF state estimator that fuses the GPS measurements as if they have no latency. The helicopter was flown (in simulation) along a complex trajectory that increased in “aggressiveness” over time. Figure 5.7 shows a 3D representation of this flight-plan trajectory with the helicopter’s true attitude superimposed at certain intervals. The flight plan included complex acrobatic maneuvers such as *rapid-rise-and-hover*, *figure-eights*, *split-s*, etc. The helicopter was commanded to sit on the landing pad for a couple of seconds while the different estimation algorithms were initialized before the flight commenced. The estimated navigational states of interest are: position, velocity and attitude. Even though the system state, kinematic model and estimation filters makes use of a quaternion representation of the vehicles attitude, we converted these values to their equivalent Euler angles for the sole purpose of reporting estimation error performance here. Internally the quaternion representation was used throughout due to its robustness

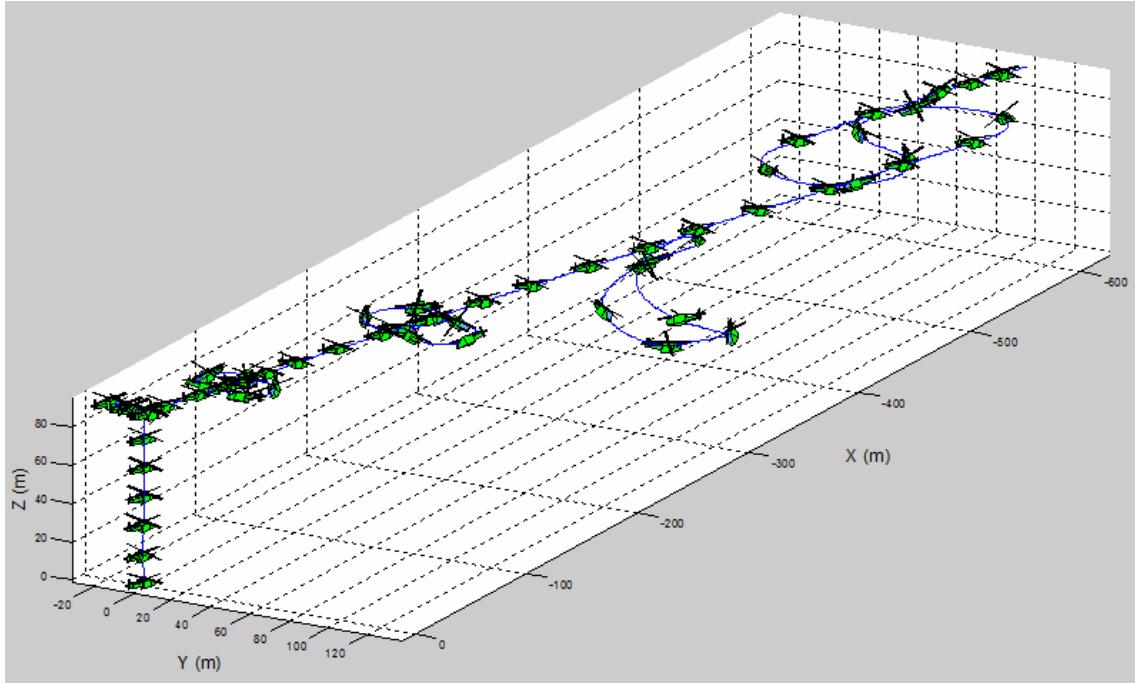


Figure 5.7: Simulator flight plan for UAV estimation experiments.

advantage as discussed earlier. For this experiment we did not “close the loop” for the flight control system. In other words, the SDRE controller used the true known states of vehicle for the online calculation of the control law. The SPKF or EKF estimated states was not fed back to the control system. This was done to ensure that the helicopter flew exactly the same flight profile when comparing the estimation performance of the different estimators. This “controlling the environment for repeatability” was one of the reasons why the high-fidelity simulation environment is so attractive when comparing different estimation approaches.

Table 5.4 compares the average root-mean-square (RMS) estimation errors for the three different state estimators. We also show (in brackets) the relative error reduction percentage for each of the two SPKF estimators compared to the EKF. These results are also presented graphically in Figure 5.8. The normal SPKF is able to reduce the 3D position and velocity estimation errors by about 10% and the roll and pitch angle estimation errors by about 20%. The biggest improvement over the EKF, 55%, is in the estimation of the yaw (heading) angle. The GPS latency compensated SPKF goes even further with

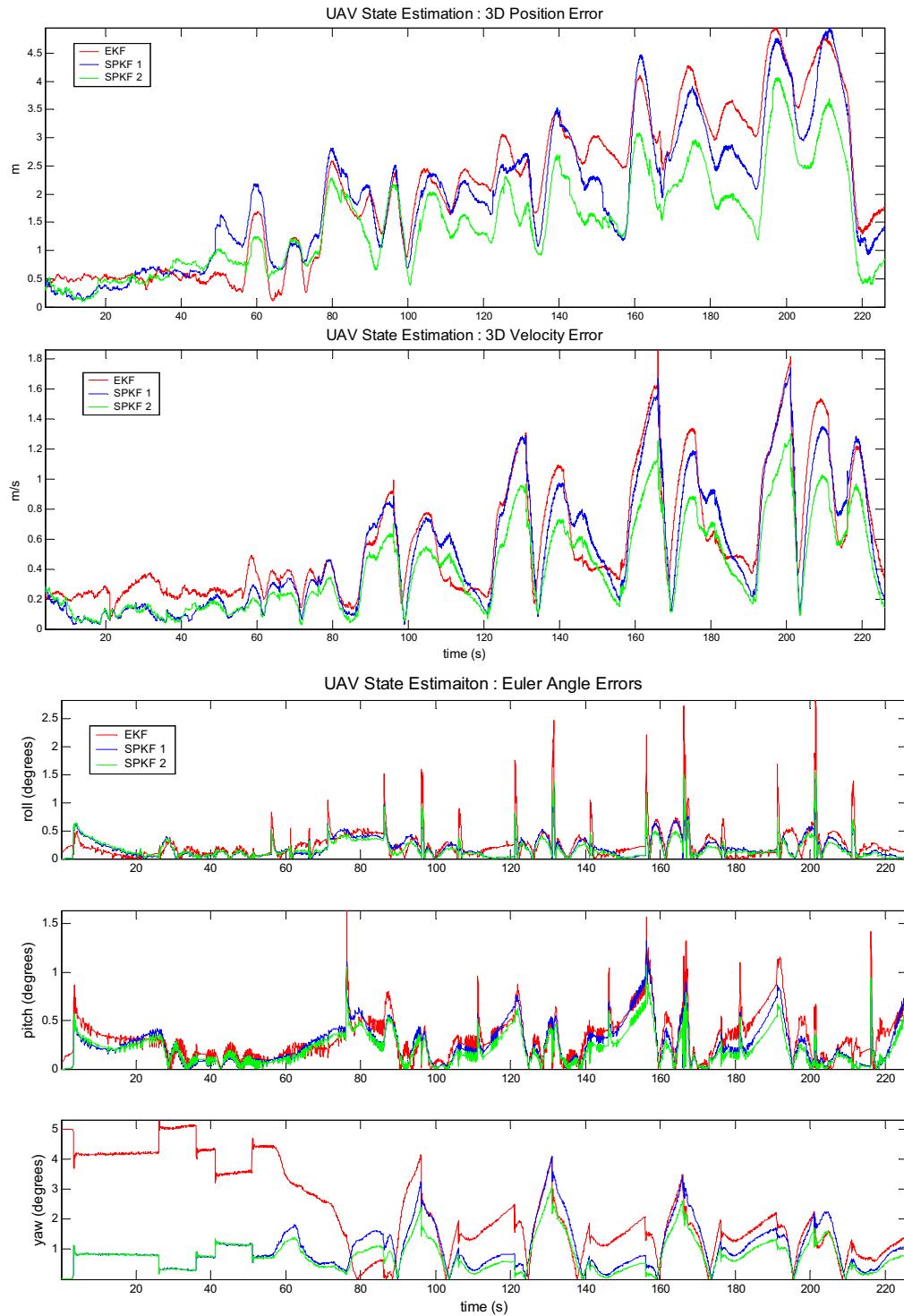


Figure 5.8: State estimation results: EKF vs. SPKF (*without* GPS latency compensation: SPKF 1) vs. SPKF (*with* GPS latency compensation: SPKF 2).

Table 5.4: UAV state estimation results : EKF vs. SPKF (with and without GPS latency compensation). The table reports average (over complete flight trajectory) root-mean-square (RMS) estimation errors for the EKF, SPKF (without GPS latency compensation) and SPKF (with GPS latency compensation) for the simulated flight shown in Figure 5.7. The estimation error reduction percentages are shown for all filters (relative to EKF).

Algorithm	Average RMS Error				
	position (m)	velocity (m/s)	Euler angles (degrees)		
			roll	pitch	yaw
EKF	2.1	0.57	0.25	0.32	2.29
SPKF - without LC	1.9 (10%)	0.52 (9%)	0.20 (20%)	0.26 (19%)	1.03 (55%)
SPKF - with LC	1.4 (32%)	0.38 (34%)	0.17 (32%)	0.21 (34%)	0.80 (65%)

a 33% reduction in position, velocity, roll angle and pitch angle errors. The yaw angle error reduction is again the highest at 65%. We repeated this experiment numerous times with different initializations and realizations of measurement noise as well as flying different flight trajectories and all of the results consistently confirmed the same relative performance between the different estimators as presented in this experiment. Clearly, even though the normal SPKF already outperforms the EKF (as expected), correctly accounting for GPS latency is well worth the extra effort. For this reason we chose the latency compensated SPKF as our preferred state estimation filter and it will be used for all of the remaining experiments presented in this chapter. Even though we will be referring to this estimator simply as the *SPKF* in all subsequent experiment descriptions, tables and plots, be aware that the use of the *GPS latency compensated SPKF* is implied.

In order to clearly illustrate the difference in estimation performance between the EKF and the (latency compensated) SPKF we present the results of another run of Experiment SE1, this time only showing the EKF and preferred SPKF implementation plots. The position and velocity estimation errors are shown in Figure 5.9 and the Euler angle estimation errors are shown in Figure 5.10. As before the SPKF clearly outperforms the EKF with the largest improvement again evident in the yaw (heading) angle estimation error. Figure 5.10 indicates how the EKF has a very large error in the yaw estimate for the first 80 seconds of the flight. This is due to a significant initial error in the underlying IMU bias estimate that is used to correct for systemic and temperature driven drift errors in the IMU

gyro and accelerometer readings. Even though the EKF and SPKF filters were initialized with exactly the same initial state estimates, the SPKF was able to converge to the true biases in the IMU measurements much quicker and then track them more accurately. This contributes (among other things) to more accurate Euler angle estimates. Although the average yaw estimate error improvement for the SPKF over the whole trajectory is 65%, this value does not accurately reflect the expected steady-state (after bias convergence) performance of the SPKF. If we calculate the average error improvement over the first 80 seconds, we find that the SPKF is actually realizing a 81% improvement. The average error improvement after bias convergence ($t > 80\text{s}$) are 43%. The steady-state error improvement of the SPKF over the EKF is thus 32%, 34% and 43% respectively for the roll, pitch and yaw angle estimates.

Another interesting performance characteristic to note from Figure 5.10 are the frequent high peaks in the EKF’s estimation error plots. These coincide with the onsets of aggressive maneuvers (banking, turns, rapid climbs, etc.) that pushes the vehicle into regimes of increased nonlinear response. The linearization errors of the EKF will therefore be more severe at these times resulting in poor estimation performance and increase estimation error. In contrast the SPKF is able to deal with these increased nonlinearities quite satisfactorily.

5.4.3 Experiment SE2: Closed loop control and estimation performance

In this experiment we “close the loop” in the GNC system by feeding the estimated states back to the SDRE control system. In other words, the vehicle control commands will now be a function of the estimates generated by either the EKF or SPKF estimator and not of the “true” vehicle states. This mimics (in simulation) the true interdependency between the estimation and control system as would occur in the real flight hardware during a fully autonomous flight. This experiment will thus not only indicate the difference in estimation performance between the different filters, but also how that translates into improved or worsened control-performance. As a measure of control performance we calculate the average *linear quadratic (LQ) control cost*, J_{LQ} , over the duration of an aerial acrobatic maneuver. The LQ control cost is a function of the *difference* between the flight-plan

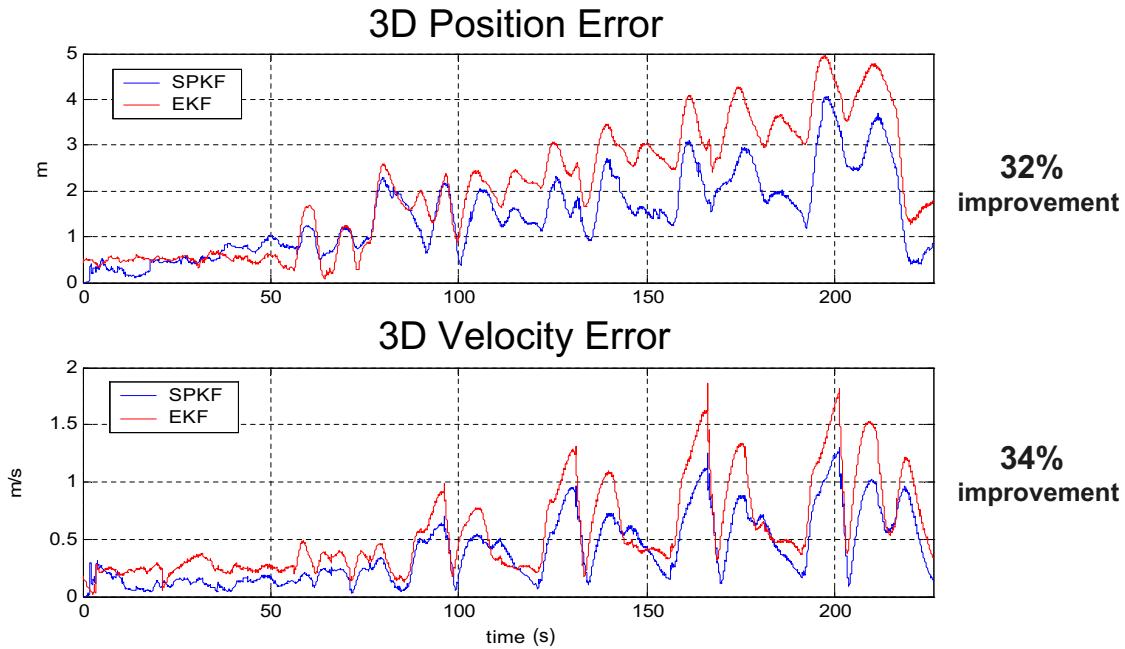


Figure 5.9: State estimation results (position and velocity error) : SPKF vs. EKF

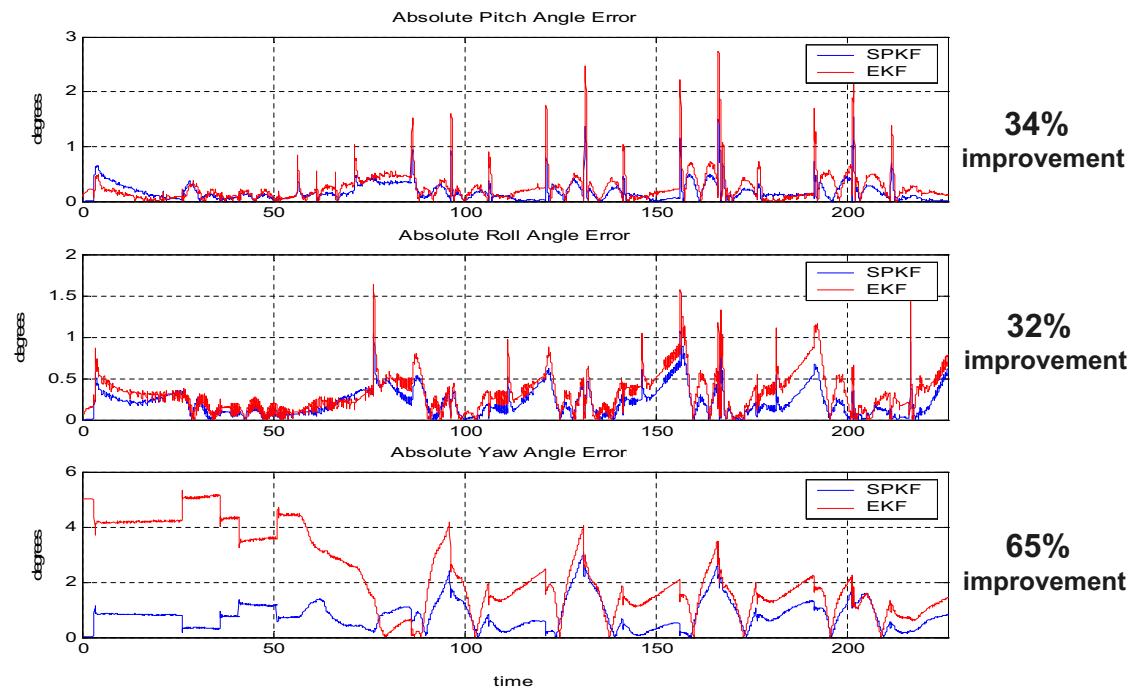


Figure 5.10: State estimation results (Euler angle errors) : SPKF vs. EKF

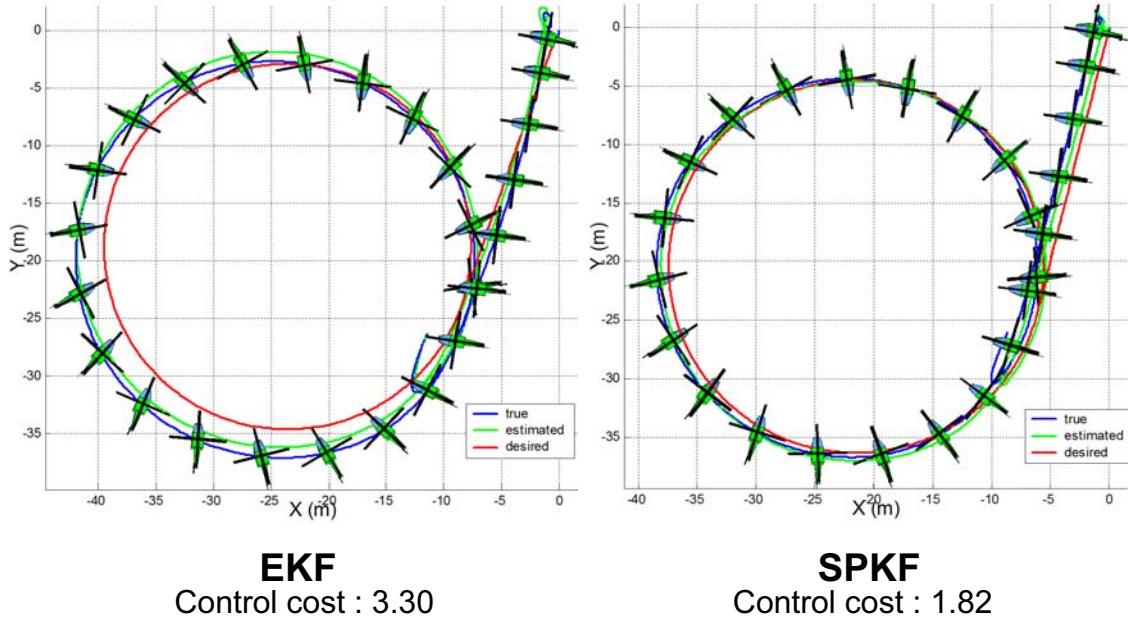


Figure 5.11: Comparative closed loop flight trajectories using SDRE controller and either EKF (left) or SPKF (right) state estimation results. Using the SPKF estimator has reduced the LQ control cost by 45%. Not only does the SPKF based system more closely track the desired trajectory (shown in red), it also results in a much better yaw (heading) tracking. This is evident from the fact that the nose of the UAV should point toward the center of the trajectory circle.

determined *desired* state trajectory (using all navigational states) and the actually realized *true* state trajectory. Again, performing this type of experiment *in simulation* allows for the calculation of the true control cost,

$$J_{LQ} = \sum_{k=0}^N c \left(\mathbf{x}_k^{des} - \mathbf{x}_k \right) , \quad (5.84)$$

where \mathbf{x}_k^{des} is the desired state (as determined by the flight-planning subsystem), \mathbf{x}_k is the true system state (as determined by the high-fidelity simulator) and $c(\cdot)$ is some error function, e.g., the squared-norm. In a real flight experiment, only the estimated control cost

$$\hat{J}_{LQ} = \sum_{k=0}^N c \left(\mathbf{x}_k^{des} - \hat{\mathbf{x}}_k \right) , \quad (5.85)$$

can be calculated where $\hat{\mathbf{x}}_k$ is the EKF or SPKF estimated states.

For this experiment we commanded the helicopter to perform an aggressive high speed

nose-in turn. This maneuver requires the helicopter to fly along an imaginary circular trajectory while constantly pointing its nose towards the exact center of the circle. Accurate position, velocity and especially yaw angle estimates are needed to accurately follow the desired flight plan with the desired attitude. Figure 5.11 shows the results of this experiment for both the EKF and SPKF. The desired flight trajectory is indicated by the red curve, the true realized trajectory in blue and the estimated trajectory in green. The true attitude of the helicopter is indicated by periodic renderings of the vehicle itself along the flight path. The left hand plot show the results for the EKF and the right hand plot those of the SPKF. Clearly for the SPKF case the estimated trajectory is not only close to the true trajectory (small estimation error), but the true trajectory is close to the *desired* trajectory which indicated good control performance. The EKF plots clearly shows worse performance according to both these criteria. Also evident from the plots is the much improved yaw angle tracking performance of the SPKF system compared to the EKF system: Not only does the helicopter renderings on the left indicate that their noses are not pointing at the true center of the desired circle, they don't even point to the same point. The SPKF system on the other hand does much better in estimating and realizing the correct yaw attitude for this maneuver. Finally, the average control cost of the EKF system for this maneuver was calculated as $J_{LQ} = 3.30$, compared to the $J_{LQ} = 1.82$ of the SPKF based system. This corresponds to a 45% reduction in control cost. These results again confirm the superior performance of the SPKF over the EKF.

5.4.4 Experiment SE3: IMU degradation experiments

The single most expensive component of our flight system is the high-performance IMU manufactured by Inertial Sciences [85]. At a cost of about \$10,000, this sensor is more expensive than the rest of the flight hardware combined (including the helicopter itself). From a cost-minimization perspective it would thus be beneficial if our SPKF based estimator can still perform satisfactory and robustly if the IS-IMU is replaced by a lower performance, and hence cheaper IMU. This is an important requirement for one of the continuing research goals of the ONR sponsored “*UAV Autonomy*” [149, 203] projects, viz. *providing high-quality navigational and situational awareness to low unit cost, expendable*

UAVs.

In order to test the hypothetical estimation performance of our SPKF based estimator when driven by a lower quality IMU, we performed a number of state estimation experiments in simulation where we artificially degraded the quality of the IMU measurements (both the accelerometer and gyro rate sensor readings). Using Equations 5.17 and 5.18 from Section 5.2.3, we can write the IMU observation model as

$$\tilde{\mathbf{a}}_k = \mathbf{a}_k + \mathbf{a}_{b_k} + \mathbf{n}_{\mathbf{a}_k} \quad (5.86)$$

$$\tilde{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_k + \boldsymbol{\omega}_{b_k} + \mathbf{n}_{\boldsymbol{\omega}_k} \quad (5.87)$$

where $\mathbf{a}_k = [a_x \ a_y \ a_z]_k^T$ and $\boldsymbol{\omega}_k = [p \ q \ r]_k^T$ are the true linear accelerations and rotational rates of the vehicle, $\tilde{\mathbf{a}}_k$ and $\tilde{\boldsymbol{\omega}}_k$ are the measured (noisy/corrupted) IMU readings, \mathbf{a}_{b_k} and $\boldsymbol{\omega}_{b_k}$ are the long term bias/drift terms, and $\mathbf{n}_{\mathbf{a}_k}$ and $\mathbf{n}_{\boldsymbol{\omega}_k}$ are additive noise terms affecting the instantaneous accuracy of the measurements. Here we assumed the IMU is located at the center of gravity (c.g.) of the vehicle. See Section 5.2.3 for a discussion of how this model should be amended if the IMU is located at a finite offset relative to the c.g. Given this observation model, the IMU signal can be degraded in following four ways:

1. Increasing the variance of the *additive noise*
2. Increasing the measurement *bias*.
3. Increasing the *drift-rate* of the measurements
4. Reducing the *update rate* of the IMU

The first factor (additive noise variance) simply affects the absolute accuracy of the IMU measurements. The second factor (bias) affects the initial bias of the measurements from their nominal (true) value. The magnitude of this bias term depends on the ambient operating temperature of the IMU. The third factor (drift-rate) affects the long-term stability of the IMU measurements. Due to the integration of the initial bias error over time, the IMU measurements will drift away from their true values over time, even if the vehicle

is motionless. This is one of the reasons why “dead reckoning” navigation systems which are purely based on IMU integration without corrective measurement updates (from GPS, altimeter, etc.) are notoriously inaccurate over long intervals. The magnitude (rate) of this drift process is related (proportional) to the ambient operating temperature of the IMU. Within the MIT-Draper-XCell90 model based simulator, the second and third factor are combined into a single *bias/drift-rate* error terms, \mathbf{a}_{b_k} and $\boldsymbol{\omega}_{b_k}$ in Equations 5.86 and 5.2. These terms are modeled as [58],

$$\mathbf{a}_{b_k} = \alpha^k \bar{\mathbf{a}}_b^s + \bar{\mathbf{a}}_b^l \quad (5.88)$$

$$\boldsymbol{\omega}_{b_k} = \alpha^k \bar{\boldsymbol{\omega}}_b^s + \bar{\boldsymbol{\omega}}_b^l, \quad (5.89)$$

where $\bar{\mathbf{a}}_b^s$ and $\bar{\boldsymbol{\omega}}_b^s$ are the short term *bias* terms, and $\bar{\mathbf{a}}_b^l$ and $\bar{\boldsymbol{\omega}}_b^l$ are the long-term *drift* terms. These terms ($\bar{\mathbf{a}}_b^s$, $\bar{\boldsymbol{\omega}}_b^s$, $\bar{\mathbf{a}}_b^l$ and $\bar{\boldsymbol{\omega}}_b^l$) are set during the simulator initialization by drawing (sampling) them from zero-mean Gaussian random variables. The magnitude of the variance of the Gaussian densities they are sampled from determines the magnitude of the respective *bias* and *drift-rate* effect as modeled by Equations 5.88 and 5.89. In other words, these terms are sampled once during initialization according to:

$$\bar{\mathbf{a}}_b^s \sim \mathcal{N}(\mathbf{a}; \mathbf{0}, \sigma_{\mathbf{a}_b^s} \mathbf{I}) \quad (5.90)$$

$$\bar{\mathbf{a}}_b^l \sim \mathcal{N}(\mathbf{a}; \mathbf{0}, \sigma_{\mathbf{a}_b^l} \mathbf{I}) \quad (5.91)$$

$$\bar{\boldsymbol{\omega}}_b^s \sim \mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \sigma_{\boldsymbol{\omega}_b^s} \mathbf{I}) \quad (5.92)$$

$$\bar{\boldsymbol{\omega}}_b^l \sim \mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \sigma_{\boldsymbol{\omega}_b^l} \mathbf{I}), \quad (5.93)$$

where the scalar multipliers $\sigma_{\mathbf{a}_b^s}$, $\sigma_{\mathbf{a}_b^l}$, $\sigma_{\boldsymbol{\omega}_b^s}$ and $\sigma_{\boldsymbol{\omega}_b^l}$ determine the magnitude of the respective bias and drift-rate effects. Once sampled, these terms are kept constant for a specific experimental run (simulation) and the resulting time-varying combined bias/drift terms are calculated according to the models of Equations 5.88 and 5.89. α is a bias filter-constant with magnitude less than one, i.e., $0 < \alpha < 1$. Typically $\alpha \approx 0.9999$ is used. This results in an exponential decay in the initial short-term bias terms ($\bar{\mathbf{a}}_b^s$ and $\bar{\boldsymbol{\omega}}_b^s$) which models the IMU temperature dependent transient response during initialization or “warm up”. Figure 5.12 shows two typical time trajectories of these bias/drift terms as generated

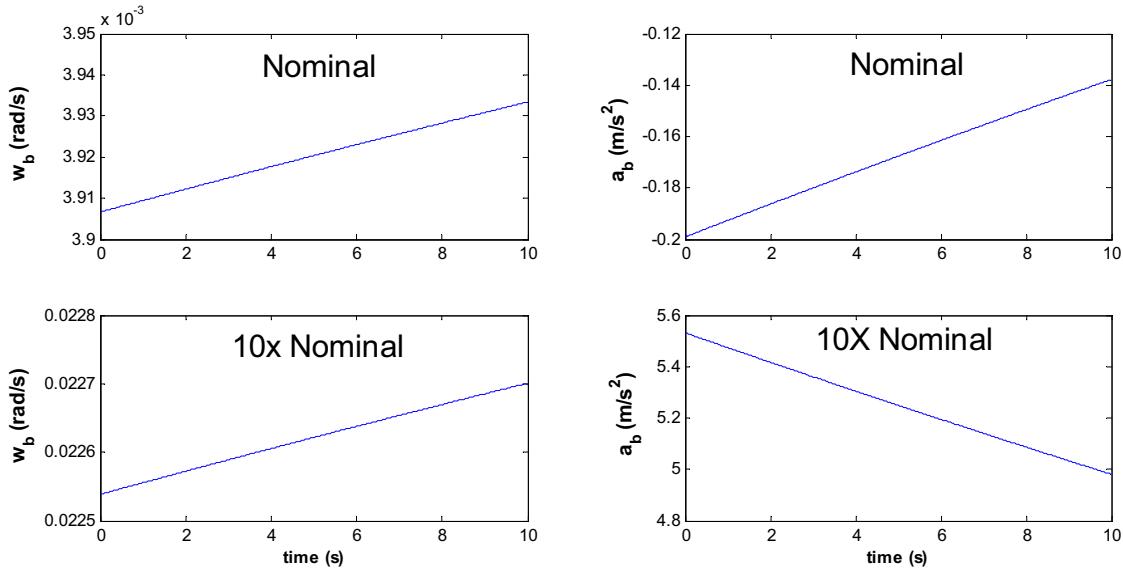


Figure 5.12: Typical bias/drift-rate plots for IMU simulation. The top two plots show the nominal (modeling the true ISIS-IMU) response for a specific realization of the accelerometer and gyro-rate bias/drift-rate components. The bottom two plots show another realization where the quality of the IMU was synthetically degraded by increasing the variance of the bias/drift-rate noise sources (Equations 5.90 - 5.93) ten fold.

by the MIT-Draper-XCell-90 simulator. The top two plots of the figure show the nominal (modeling the true ISIS-IMU) response for a specific realization of the accelerometer and gyro-rate bias/drift-rate components. The bottom two plots show another realization where the quality of the IMU was synthetically degraded by increasing the variance of the bias/drift-rate noise sources (Equations 5.90 - 5.93) ten fold.

The fourth and final factor, IMU update rate, determines at what rate the IMU can provide new data to the flight computer. Since the IMU directly drives the kinematic model, the filter update rate is directly linked to the IMU rate. A reduction of the update rate is thus expected to negatively impact the estimation performance of the system.

For the IMU degradation experiments reported here, we simulated an aggressive take-off followed by the same “nose-in turn” as used in Experiment SE2. We varied each of the above mentioned four IMU “performance degraders” separately while calculating average estimation error as well as average control cost over the complete flight profile. As already mentioned, the simulator combines the *bias* and *drift-rate* factors into a single term. We

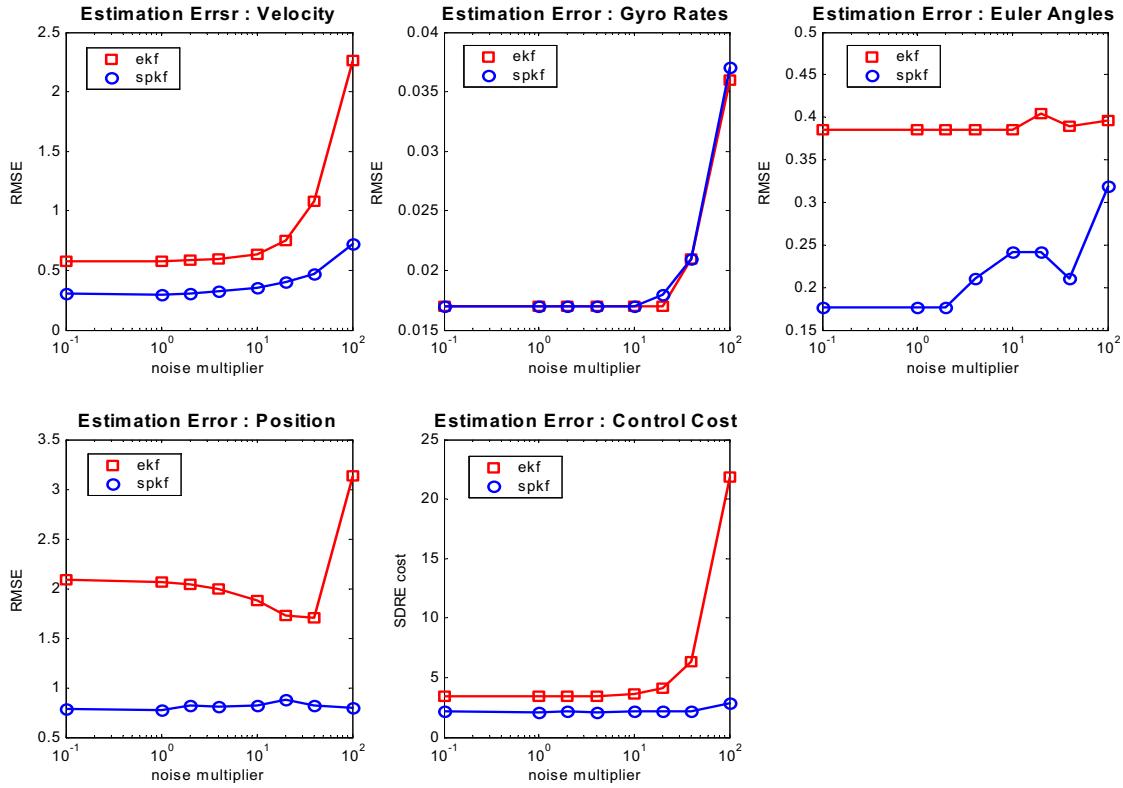


Figure 5.13: IMU degradation experiment: additive noise

varied the total magnitude of this factor in the same manner as was demonstrated in Figure 5.12, i.e., we scaled all of the relevant nominal variance terms using a single *noise-multiplier*. Although varying these factors separately would have been preferable, the combined approach was dictated by the current design of the simulator. We plan to upgrade the simulator in the future allowing for more accurate IMU modeling.

Figure 5.13 shows the results for the *additive noise* variation experiment. As the first figure indicates, the SPKF state estimator robustly maintains a high level of state estimate accuracy as well as a resulting low control cost over a wide range of additive noise levels. Only at very high noise levels does the performance start to degrade significantly. Over the whole range the SPKF estimator also consistently outperforms the EKF based system and also exhibit better degradation characteristics.

For the next set of experiments the variance of the bias/drift-rate noise terms were scaled relative to their nominal values using a single scalar noise-multiplier factor. All

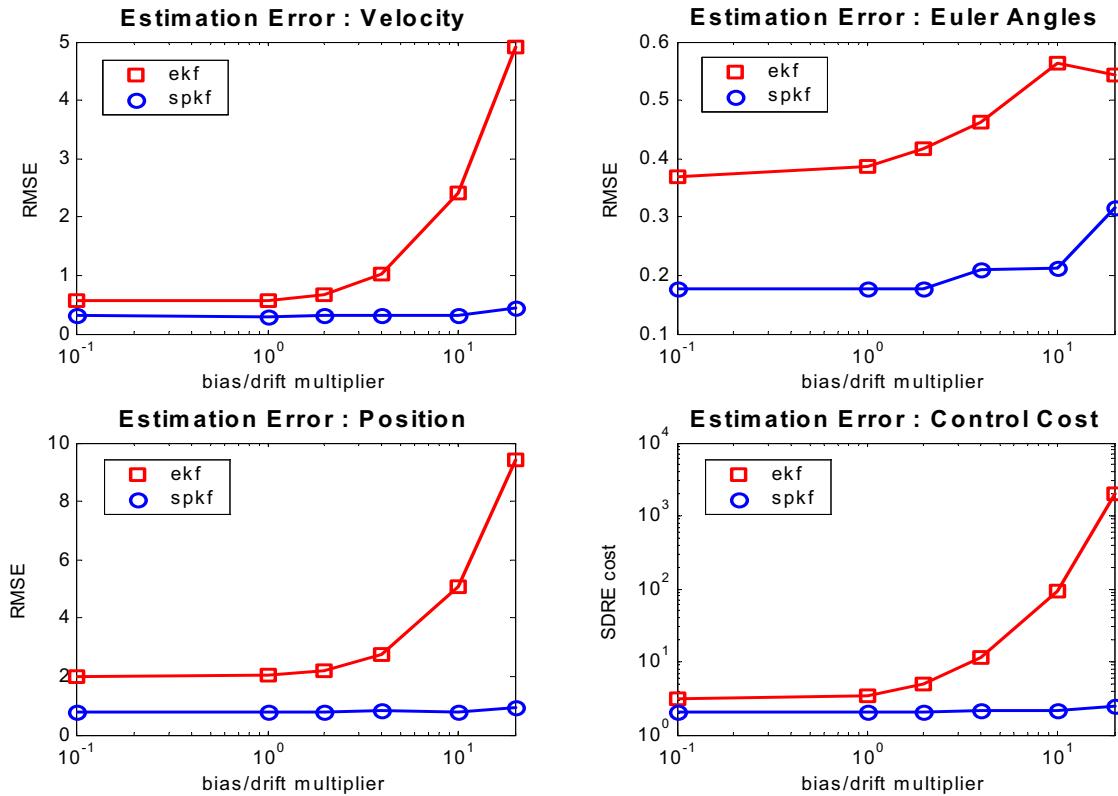


Figure 5.14: IMU degradation experiment: drift/bias

other experiment parameters were kept constant. The same flight profile as the previous experiment is used. Again estimation errors and control costs where averaged over the complete trajectory. Also note that for all of the different runs, nothing was changed with regard to filter/estimator implementations (EKF and SPKF), i.e., the filters were not fine tuned to the different bias/drift levels through noise variance adjustments on the “filter side”. Only the levels in the simulator was changed. This was done to determine how robustly the different estimators can handle observation model mismatch. Figure 5.14 shows the results for this sensor *bias/drift* variation experiment. The SPKF again consistently outperform the EKF and maintains near nominal performance over the whole variance range. Only at high bias/drift-rate levels does the performance begin to degrade and even then the degradation is very gradual.

For the final IMU degradation test, we performed a number of simulation experiments where we artificially reduced the *update rate* of the IMU. The actual used were: (1) 100Hz

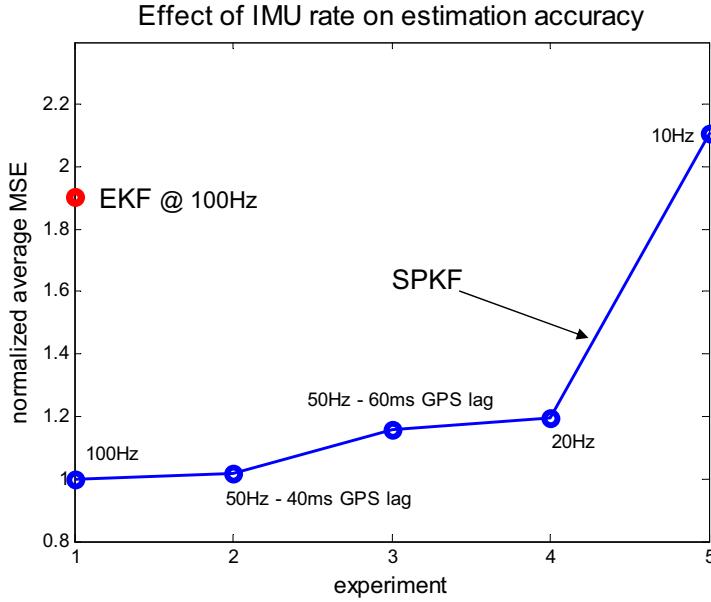


Figure 5.15: IMU degradation experiment: IMU update rate .

(nominal value) - 50ms GPS latency; (2) 50Hz - 40ms GPS latency setting; (3) 50Hz - 60ms GPS latency setting; (4) 20Hz - 50ms GPS latency; (5) 10Hz - 50ms GPS latency.

Since we still need to deal with GPS latency, which is 50ms for our Ashtech G12 GPS receiver [130], the 50Hz IMU update rate experiment posed a problem: At a 50Hz rate each update window is 20ms long, which is not an integer divisor of the GPS latency. This means that we have to either assume a 40ms or a 60ms latency window for the GPS in the SPKF based latency compensated state estimator. We thus repeated the 50Hz experiment twice: once for the 40ms latency setting and once for the 60ms setting. Unfortunately for this experiment we could not calculate comparative performance relative to the EKF due to implementational constraints: The MIT designed EKF is tightly integrated into the flight simulation software, both for the software-in-the-loop and hardware-in-the-loop systems which has been highly optimized for the specific rates of the different components (sensors, controller, estimator, etc.). Unlike our external Matlab implemented SPKF estimator, accommodating a different IMU rate is thus not easily achieved without significantly changing certain parts of their already optimized source code. For this reason we simply compared the performance of our SPKF estimator at the different IMU rates relative to

its own performance at the optimal 100Hz rate. The normalized (with respect to the nominal 100Hz estimator) average MSE values are shown in Figure 5.15. In comparison, the nominal EKF performance at 100Hz is also shown (normalized to SPKF @ 100Hz data point).

Clearly the SPKF based estimator still performs quite well even when the IMU (and filter) is running at half of its nominal rate (50Hz). It also seems like the 40ms approximation for the GPS latency gives better performance especially with regard to position and velocity estimates. This indicated that underestimating the latency seems to be less detrimental to filter performance than over estimation. In general the performance with regard to IMU rate will most likely be a function of the severity (aggressiveness) of the vehicles maneuvers. In other words, if the roll, pitch and yaw rate as well as linear acceleration bandwidths increase, one would expect the requirements on a fast IMU update rate to become more strict. Also note how, even at low update rates, the SPKF still outperforms the nominal EKF performance running at the full 100Hz IMU rate.

Based on the results of the earlier state estimation experiments and the IMU degradation robustness experiments reported above, we feel confident that our GPS latency compensated SPKF state estimator provides a robust, highly accurate, stand-alone navigation solution for UAV platforms. Since this system is mostly vehicle agnostic, we are confident that it can easily be adapted to work on a variety of related aircraft platforms utilizing avionics sensors of varying performance. This is an ongoing research issue which is addressed further in Chapter 7.

5.5 Parameter Estimation Experimental Results

In Section 3.5.2 of Chapter 3 we showed how the SPKF can be used for parameter estimation (system identification) given a general nonlinear DSSM. In this section we will show how we used such a SPKF based parameter estimation system to determine the true values of the parameters of the MIT-Draper-XCell90 model.

As pointed out earlier, one of the advantages of the 6DOF IMU driven kinematic model (as used in our SPKF state estimator) is that it is for the most part “vehicle agnostic” and

that it has very few parameters. The SDRE control system however, makes use of the full nonlinear MIT-Draper-XCell90 model when calculating the optimal control law. This requires it to also known the true values of the parameters of this model. Table 5.2 list the ≈ 70 parameters that are used by this model. If these values are not accurately known *a-priori*¹⁹, they can be determined (or at least fine-tuned) on-line using a parameter estimation system. As discussed in Chapter 3, system parameters can be estimated by such a system if we have access to the clean system states. Since we almost *never* have access to the clean states, most practical parameter estimation systems are in fact *dual estimation* systems, i.e., we need to estimate the states and the parameters. Such a system is discussed in detail in Section 5.6.

In this section we will thus determine if, in the best (if not practical) case scenario where we do have access to the true system states, we can in fact determine some of the systems model parameters. A number of parameter estimation experiments using a SPKF applied to the full MIT-Draper-XCell90 model is presented as a “proof of concept”.

5.5.1 Implementation Issues

Since the operational stability of any vehicle control system is of high importance, certain safeguards must be put in place to ensure robustness and limit the “worst case scenario” that might occur especially when real-time adaptive systems are used. This has direct implications on any parameter estimation system which is used to set the values of the system parameters as used by the control system. For this reason, the parameters are often not estimated directly, but rather lower magnitude “correction terms” are estimated which are then added to some nominal a-priori value of the parameter vector. In other words, if \mathbf{w} is the true value of the parameter vector, the parameter estimation system makes use of the following decomposition:

$$\mathbf{w} = \tilde{\mathbf{w}}_0 + \boldsymbol{\delta}\mathbf{w}, \quad (5.94)$$

¹⁹Typically a variety of complex off-line experimental methods are used to accurately estimate certain “observable” system parameters. For detail on how this was done for our specific UAV platform, please consult [58].

where $\tilde{\mathbf{w}}_0$ is the nominal value of the parameters and $\delta\mathbf{w}$ is the relative offset vector. The parameter estimation system is now used to rather estimate $\delta\mathbf{w}$, i.e., $\hat{\delta\mathbf{w}}$, than $\hat{\mathbf{w}}$. To further prevent divergence of the estimates one can use a nonlinear “squashing function” to limit the absolute magnitude of the estimated correction term, i.e.,

$$\mathbf{w} = \tilde{\mathbf{w}}_0 + \tanh(\delta\mathbf{w}) . \quad (5.95)$$

This will prevent the estimation system calculating parameter estimates that might make mathematical sense, but not physical sense, such as negative moments of inertia or negative masses.

If the relative magnitude of the parameters to be estimated differs significantly, it can possibly lead to numerical instability (covariance matrices becoming ill-conditioned) as well as slow convergence [75]. For this reason, we pre-multiply the limited correction term by a scaling gain factor, i.e.,

$$\mathbf{w} = \tilde{\mathbf{w}}_0 + \mathbf{g}_{\mathbf{w}} \cdot \tanh(\delta\mathbf{w}) . \quad (5.96)$$

The gain vector term is given by,

$$\mathbf{g}_{\mathbf{w}} = \alpha_w |\tilde{\mathbf{w}}_0| , \quad (5.97)$$

where α_w is a scalar value usually set to $0.1 \leq \alpha_w \leq 0.4$ and $|\tilde{\mathbf{w}}_0|$ is the absolute value of the nominal parameter values vector. Using this formulation, the actual parameter vector estimated by the SPKF parameter estimation filter is given by,

$$\hat{\mathbf{w}}_{\delta} = \hat{\delta\mathbf{w}} = \arctan[(\hat{\mathbf{w}} - \tilde{\mathbf{w}}_0) ./ \mathbf{g}_{\mathbf{w}}] , \quad (5.98)$$

where $./$ is the per-element vector division operator, i.e. $\mathbf{a}./\mathbf{b} = [a_1/b_1 \ a_2/b_2 \ \dots \ a_n/b_n]$. We found this formulation to result in improved convergence speed and numerically well behaved estimates and covariances.

All of the parameter estimation experiments presented below made use of the SR-CDKF implementation of the SPKF framework.

5.5.2 Experiment PE1: Static system parameter identification

In this experiment we flew the helicopter through a number of aggressive maneuvers while running an incorrectly initialized SPKF based parameter estimation filter on a subset of the MIT-Draper-XCell-90 model parameter vector. Specifically, we estimated the following parameters: *mass* (m), *moments of inertia* ($I_{xx}, I_{yy}, I_{zz}, I_{xz}$), *main rotor hub torsional stiffness* (K_β), *main rotor lift curve slope* (C_{la}^{mr}), *main rotor profile drag coefficient* (C_{do}^{mr}), and the *tail rotor pitch offset* ($\tilde{\phi}_0^{tr}$). The helicopter sat on the ground for 5 seconds (for filter initialization) after which a rapid take-off was commanded. Figure 5.16 shows the estimation results for the first 30 seconds of the flight. The dashed red lines indicate the true values of the different model parameters (as used in the simulator) and the solid blue lines indicate the trajectory of the estimates generated by the SPKF. The large ($\pm 20\%$) initialization error for the different parameters are clearly visible over the first 5 seconds of all the plots. Clearly the filter quickly converges to the true values of the underlying system parameters. For this experiment we used the Robbins-Monro method (see Section 3.5.2 for detail) to adapt the covariance of the artificial process noise on-line.

5.5.3 Experiment PE2: Dynamic system parameter identification and tracking

One of the ultimate aims of our SPKF based parameter estimation system is not only to identify but to *track* changes in system parameters on-line. Many real-world flight events such as payload deployment or fault conditions²⁰ can result in changes in the flight model parameters which, if not compensated for in the GNC system, can lead to suboptimal behavior or even system failure.

In this experiment we used the SPKF based parameter estimation system to track the *mass* of the helicopter during the course of a flight. We dynamically changed the true mass of the helicopter over time, simulating the effect of (linear) fuel consumption. We assumed the helicopter weighs 8kg when fully fueled and that it will consume 1.75kg of fuel over the

²⁰Fault conditions can include anything from stuck or broken actuators to significant flight surface damage caused by impacts, weather, etc.

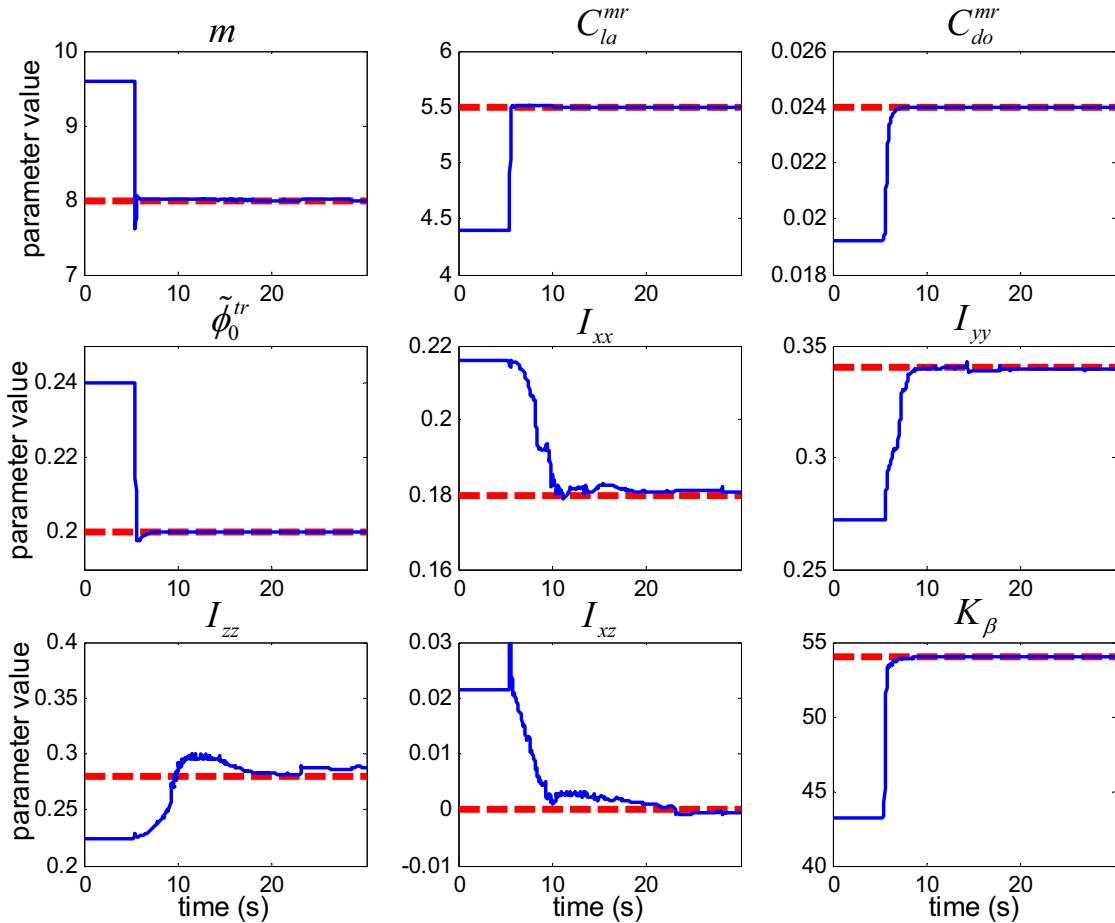


Figure 5.16: Parameter estimation using full nonlinear MIT-Draper-XCell-90 model and known states. The dashed red lines indicate the true value of the system parameters and the blue lines indicates the estimates generated by the SPKF based parameter estimation system.

first 30 seconds of the flight²¹. At the 25 second mark of the flight, we further simulated a discrete 2kg step in the true value of the helicopter’s mass, corresponding to a payload deployment²². For this experiment we again used a SR-CDKF SPKF implementation as well as the Robbins-Monro process noise adaptation technique.

Figure 5.17 shows the SPKF’s tracking results for this experiment. Clearly the estimation system is quite capable of accurately tracking the true underlying system parameter, with only a little bit of overshoot right after the “step” at 25 seconds.

Unfortunately, as we already pointed out earlier, these parameter estimation experiments makes use of the known (non estimated) states of the system as inputs to the SPKF estimator. Therefore, the excellent tracking performance we witnessed in this experiment is at most an upper limit on performance. Using a more realistic (and practically implementable) dual estimation system, we expect the estimation and tracking performance to go down. This is further investigated in the next section where we repeat this experiment using a dual estimation framework.

5.6 Dual Estimation Experimental Results

In this final section of the chapter, we bring together the kinematic model based SPKF state estimator and the full MIT-Draper-XCell-90 model based parameter estimator in an attempt to perform dual (state and parameter) estimation on the UAV system. Although we already showed how the kinematic model based SPKF state estimator delivers very high quality estimates of the navigational state of the vehicle (as defined by Equation 5.9), there are further auxiliary state variables which are important from a *control system design perspective* that are not estimated by this estimator [23, 22, 21, 20]. These states are however contained in the 43 dimensional state vector of the full MIT-Draper-XCell-90 model (see Equation 5.5). In this section we will describe a SPKF based dual estimation framework we implemented to estimate some of these auxiliary states as well as estimate and track certain model parameters.

²¹Even though this “artificially low” fuel efficiency is more akin to that of a SUV than a UAV, it does allow for an interesting parameter tracking problem.

²²For a UCAV (unmanned combat aerial vehicles) this might correspond to the firing/releasing/dropping of a munition.

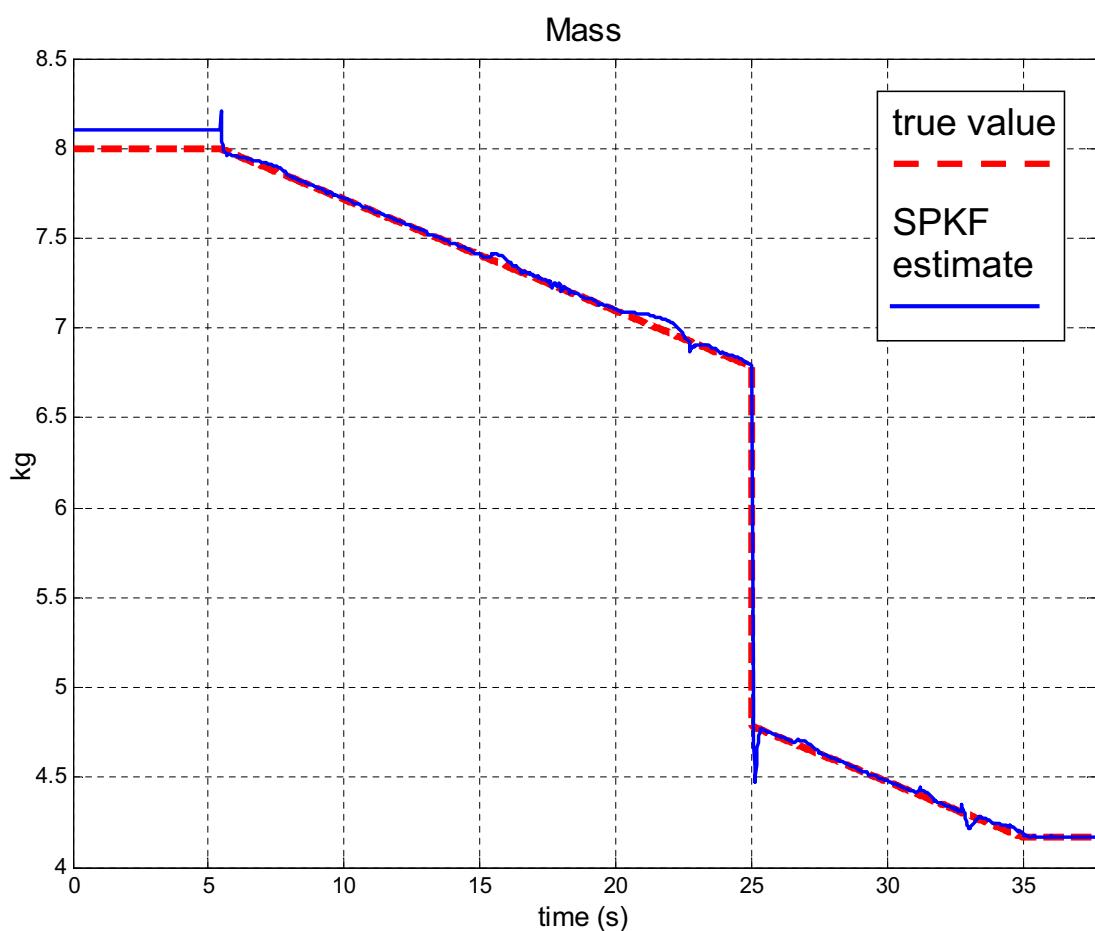


Figure 5.17: Tracking a time-varying system parameter (vehicle mass in this case).

5.6.1 System Implementation

We make use of a *joint SPKF* framework (see Section 3.5.3) with the joint state defined as the concatenation of the 26 dimensional state of the MIT-Draper-XCell-90 model with whatever subset of parameters (from the same model) we wish to estimate.

Joint Process Model

The joint process model is given by the concatenation of the *full* nonlinear MIT-Draper-XCell-90 model with the standard random walk parameter estimation model as presented in Section 3.5.3. The full C-code implemented MIT-Draper-XCell-90 model (as used in the high-fidelity simulator) was wrapped in a Matlab MEX file layer [187] so that it can be called directly from the SPKF based joint estimator. The process noise vector dimension is scaled accordingly and adapted using the Robbins-Monro method.

Joint Observation Model

For our observation model of the joint state, we make use of the standard “direct noisy form” given by

$$\mathbf{y}_k = \mathbf{x}_k^{sub} + \mathbf{n}_k^{sub}, \quad (5.99)$$

were \mathbf{x}_k^{sub} is the subset of the full state vector that is being observed and \mathbf{n}_k^{sub} is the corresponding observation noise term. For part of our observation vector we use the output of the *kinematic state estimator* as the direct (but noisy) partial observation of joint system state, i.e.,

$$\mathbf{y}_k^a \equiv \hat{\mathbf{x}}_k^{kse}, \quad (5.100)$$

where $\hat{\mathbf{x}}_k^{kse} = \begin{bmatrix} \hat{\mathbf{p}}_k^T & \hat{\mathbf{v}}_k^T & \hat{\mathbf{e}}_k^T \end{bmatrix}^T$ is a 10×1 sub-vector composed of the first 10 components of the state estimate vector as calculated by the kinematic model based SPKF estimator, described in Section 5.3.1. The time-varying covariance of this estimate, $\mathbf{P}_{\hat{\mathbf{x}}_k^{kse}}$, is used as

the observation noise covariance of the first part of the observation, i.e.,

$$\begin{aligned}
\mathbf{R}_{\mathbf{n}_k^a} &= E \left[\mathbf{n}_k^a (\mathbf{n}_k^a)^T \right] \\
&= E \left[(\mathbf{y}_k^a - \mathbf{x}_k^{kse}) (\mathbf{y}_k^a - \mathbf{x}_k^{kse})^T \right] \\
&= E \left[(\hat{\mathbf{x}}_k^{kse} - \mathbf{x}_k^{kse}) (\hat{\mathbf{x}}_k^{kse} - \mathbf{x}_k^{kse})^T \right] \\
&= \mathbf{P}_{\mathbf{x}_k^{kse}} .
\end{aligned} \tag{5.101}$$

In other words, we ran the previously described *kinematic model based SPKF state estimator* (with GPS latency compensation) in parallel with the currently discussed *dual estimator* to generate part of the observation vector online. One can think of the kinematic model based state estimator as a complex higher-level observation sensor that generates partial measurements of the larger MIT-Draper-XCell-90 model based system for the joint SPKF estimator.

We further augmented the observation vector with the bias-corrected IMU gyro rate measurements, i.e.,

$$\mathbf{y}_k^b \equiv \bar{\omega}_k , \tag{5.102}$$

where $\bar{\omega}_k$ is defined by Equation 5.18. The noise covariance of this term, $\mathbf{R}_{\mathbf{n}_k^a}$, is set according to the noise specifications of the IS-IMU [85]. The final combined observation model is thus given by,

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{y}_k^a \\ \mathbf{y}_k^b \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_k^{kse} \\ \bar{\omega}_k \end{bmatrix} , \tag{5.103}$$

with measurement noise covariance given by,

$$\mathbf{R}_{\mathbf{n}_k^{sub}} = \begin{bmatrix} \mathbf{R}_{\mathbf{n}_k^a} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\mathbf{n}_k^b} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_k^{kse}} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\bar{\omega}_k} \end{bmatrix} . \tag{5.104}$$

Using the process and observation models described above within a standard joint SPKF framework (utilizing a SR-CDKF core), we performed the following dual estimation experiments as a preliminary proof of concept. As we will further discuss in Chapter 7 this

is an ongoing research effort within the larger UAV project.

5.6.2 Experiment D1: Joint estimation of auxiliary states and dynamic tracking of vehicle mass

We ran a number of experiments where we estimated the full 26 dimensional state of the MIT-Draper-XCell-90 model, while at the same time trying to estimate and track a certain subset of time-varying parameters. The specific auxiliary states we are interested in are the *longitudinal* and *lateral flapping angles* of the main rotor blades (β_{lat} and β_{lon}), as well as the rotational rate states of the main engine shaft (Ω_{mr} and Ω_{mrI}). These states estimates provide useful information which can be used to improve the design and robustness of the SDRE controller [20, 21]. At the same time we tracked the helicopter's *mass* parameter using the same "fuel-consumption/payload-deployment" simulation of Experiment PE2. The dual estimation results of this experiment are shown in Figures 5.18 and 5.19.

Figure 5.18 shows how the joint SPKF is capable of accurately tracking the flapping angles and engine RPM states of the helicopter. Figure 5.19 shows how the estimation system tracks the time varying mass of the helicopter: Again, a large initial mis-adjustment of the mass estimate was quickly corrected. The estimator response to the discrete change in mass (due to payload deployment) at the 25 second mark is clearly slower (over damped) compared to the response of the pure parameter estimator (Section 5.5: Experiment PE2). The system is capable however of accurately tracking the subsequent linear decrease in mass due to fuel consumption.

It turned out that full dual estimation (all states and parameters) for the complete UAV system is an extremely hard problem; not so much in estimating the auxiliary states, but in accurately identifying the full set of system parameters using only the noisy observations (as defined in Section 5.6.1) as input. We found that the joint estimator were able to converge to the true values of some of the parameters, while sometimes getting stuck in local minima for the other parameters, depending on the type and aggressiveness of the maneuver being flown. This is probably related to the notion of *persistently exciting inputs* [124, 122], i.e., not all modes of the helicopter's dynamics are sufficiently excited by any given maneuver such that the navigational state estimates generated by the kinematic

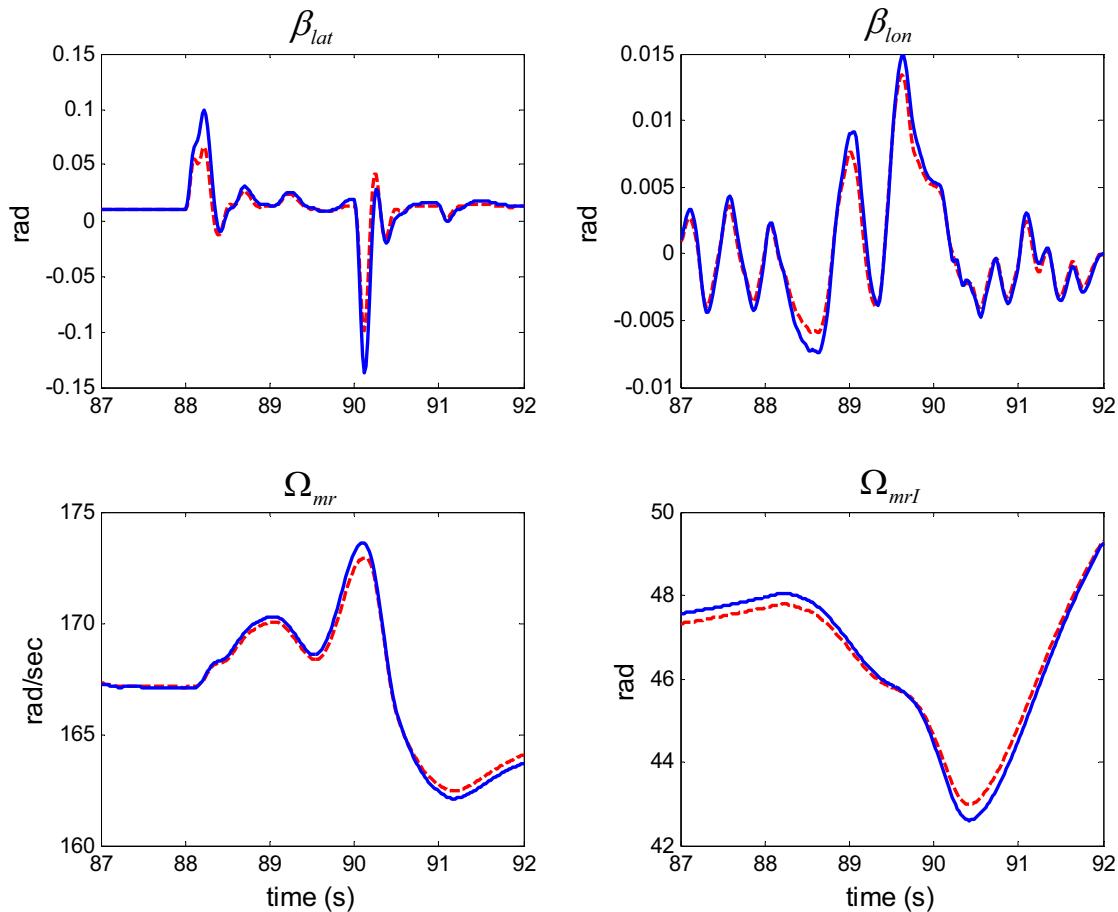


Figure 5.18: Joint estimation : Auxiliary hidden system states

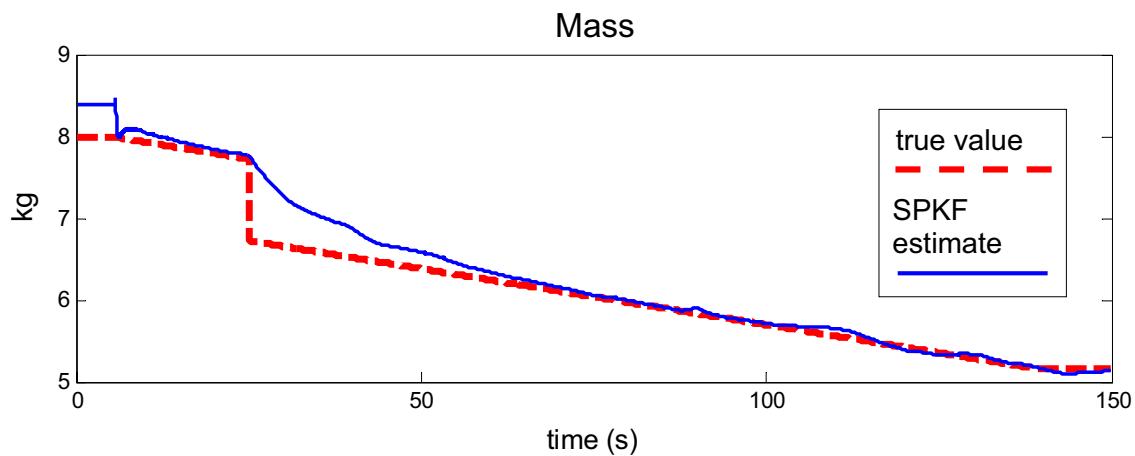


Figure 5.19: Joint estimation : tracking time-varying system mass

model based estimator can accurately observe the effect of all model parameters at all time. A *dual SPKF* implementation (as opposed to the *joint SPKF*) exhibited the same difficulties in uniquely identifying all parameters. This is an ongoing research issue.

5.7 Chapter Summary

In this chapter we covered in detail how a SPKF based integrated guidance & navigation system (GNS) for UAV autonomy was developed, implemented and tested. This application served as a testing bed and real-world verification of many of the theoretical and algorithmic concepts developed and presented in the earlier chapters. Although this specific application is an ongoing research effort under the earlier mentioned DARPA [38, 105] and ONR [149, 203] sponsored research projects, with a number of milestones still outstanding (such as testing on real hardware and real flight-data with further refinement of the dual estimation system), the preliminary results achieved so far and reported here is very encouraging. To our judgment we have shown that the SPKF is not only a viable, but in fact, a preferred alternative to the industry standard EKF currently used in most UAV GNC systems. All of the experiments showed how the SPKF based estimator consistently outperforms the hand-tuned state-of-the-art EKF based system developed by MIT for this exact platform. Furthermore, a novel new sensor latency technique that directly leverages the sigma-point approach of the SPKF framework was introduced. We demonstrated the increased estimation accuracy benefit of this approach through experimental verification. We are currently in the process of comparing this new method to other published latency compensation techniques. The results of this investigation will be published in due course [195].

The next step in the real-world validation of the system presented in this chapter, is testing it on real flight-data (as opposed to simulated data). We are currently in the process of recording the needed telemetry using our XCell-90 UAV platform for this purpose as well as migrating the Matlab implemented SPKF code to C for real-time implementation of the flight hardware.

Chapter 6

Non-Gaussian Bayesian Estimation: Sequential Monte Carlo / SPKF Hybrids

6.1 Introduction

The SPKF, like the EKF, still assumes a Gaussian posterior which can fail in certain nonlinear non-Gaussian problems with multi-modal and/or heavy tailed posterior distributions. The Gaussian sum filter (GSF) [3] addresses this issue by approximating the posterior density with a finite Gaussian mixture and can be interpreted as a parallel bank of EKFs. Unfortunately, due to the use of the EKF as a subcomponent, it also suffers from similar shortcomings as the EKF. Recently, particle based sampling filters have been proposed and used successfully to recursively update the posterior distribution using *sequential importance sampling and resampling* [45]. These methods (collectively called *particle filters*) approximate the posterior by a set of weighted samples without making any explicit assumption about its form and can thus be used in general nonlinear, non-Gaussian systems. In this chapter, we present hybrid methods that utilizes the SPKF to augment and improve the standard particle filter, specifically through generation of the *importance proposal distributions*. We will review the background fundamentals necessary to introduce particle filtering and then discuss the two extensions based on the SPKF, the *Sigma-Point Particle Filter* (SPPF) and the *Gaussian-Mixture Sigma-Point Particle Filter* (GMSPPF).

6.2 Particle Filters: Monte Carlo Simulation and Sequential Importance Sampling

Particle filtering is based on Monte Carlo simulation with sequential importance sampling (SIS) [44, 118]. The overall goal is to directly implement optimal recursive Bayesian estimation (see Section 1.4) by recursively approximating the complete posterior state density.

6.2.1 Perfect Monte Carlo Simulation

In Monte Carlo simulation, a set of weighted particles (samples), drawn from the posterior distribution, is used to map integrals to discrete sums. More precisely, the posterior filtering density can be approximated by the following empirical estimate:

$$p(\mathbf{x}_k | \mathbf{y}_{1:k}) \approx \hat{p}(\mathbf{x}_k | \mathbf{y}_{1:k}) = \frac{1}{N} \sum_{i=1}^N \delta\left(\mathbf{x}_k - \mathbf{x}_k^{(i)}\right), \quad (6.1)$$

where the random samples $\{\mathbf{x}^{(i)}; i = 1, 2, \dots, N\}$, are drawn from $p(\mathbf{x}_k | \mathbf{y}_{1:k})$ and $\delta(\cdot)$ denotes the Dirac delta function. The posterior *filtering* density, $p(\mathbf{x}_k | \mathbf{y}_{1:k})$, is a marginal of the *full* posterior density given by $p(\mathbf{x}_{0:k} | \mathbf{y}_{1:k})$. Consequently, any expectations of the form

$$E[\mathbf{g}(\mathbf{x}_k)] = \int \mathbf{g}(\mathbf{x}_k) p(\mathbf{x}_k | \mathbf{y}_{1:k}) d\mathbf{x}_k, \quad (6.2)$$

can be approximated by the following estimate:

$$E[\mathbf{g}(\mathbf{x}_k)] \approx \tilde{E}[\mathbf{g}(\mathbf{x}_k)] = \frac{1}{N} \sum_{i=1}^N \mathbf{g}\left(\mathbf{x}_k^{(i)}\right). \quad (6.3)$$

For example, letting $\mathbf{g}(\mathbf{x}_k) = \mathbf{x}_k$ yields the optimal MMSE estimate $\hat{\mathbf{x}}_k \approx E[\mathbf{x}_k | \mathbf{y}_{1:k}]$. The particles $\mathbf{x}_k^{(i)}$ are assumed to be independent and identically distributed (i.i.d.) for the approximation to hold. According to the law of large numbers, as N goes to infinity the estimate converges to the true expectation almost surely, i.e.,

$$\tilde{E}[\mathbf{g}(\mathbf{x}_k)] \xrightarrow[N \rightarrow \infty]{\text{a.s.}} E[\mathbf{g}(\mathbf{x}_k)]. \quad (6.4)$$

Moreover, if the posterior variance of $\mathbf{g}(\mathbf{x}_k)$ is bounded, that is

$$\text{var} [\mathbf{g}(\mathbf{x}_k)] = E \left[\mathbf{g}(\mathbf{x}_k) \cdot \mathbf{g}(\mathbf{x}_k)^T \right] < \infty , \quad (6.5)$$

then the following central limit theorem holds [45]:

$$\sqrt{N} \left(\tilde{E} [\mathbf{g}(\mathbf{x}_k)] - E [\mathbf{g}(\mathbf{x}_k)] \right) \xrightarrow[N \rightarrow \infty]{} \mathcal{N}(\mathbf{0}, \text{var} [\mathbf{g}(\mathbf{x}_k)]) , \quad (6.6)$$

where $\xrightarrow[N \rightarrow \infty]{}$ denotes convergence in distribution.

Sampling from the filtering posterior is only a special case of Monte Carlo simulation which in general deals with the complete posterior density, $p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$. We will use this more general form to derive the particle filter algorithm.

6.2.2 Bayesian Importance Sampling

As mentioned in the previous section, one can approximate the posterior distribution with a function on a finite discrete support. Consequently, it follows from the strong law of large numbers that as the number of samples N increases, expectations can be mapped into sums. Unfortunately, it is often impossible to sample directly from the posterior density function. However, we can circumvent this difficulty by making use of *importance sampling* and alternatively sampling from a known, easy-to-sample, *proposal distribution* $\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$. The exact form of this distribution is a critical design issue and is usually chosen in order to facilitate easy sampling. The details of this is discussed later. Given this proposal distribution we can make use of the following substitution,

$$\begin{aligned} E [\mathbf{g}(\mathbf{x}_{0:k})] &= \int \mathbf{g}(\mathbf{x}_{0:k}) \frac{p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) d\mathbf{x}_{0:k} \\ &= \int \mathbf{g}(\mathbf{x}_{0:k}) \frac{p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k}) p(\mathbf{x}_{0:k})}{p(\mathbf{y}_{1:k}) \pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) d\mathbf{x}_{0:k} \\ &= \int \mathbf{g}(\mathbf{x}_{0:k}) \frac{w_k(\mathbf{x}_{0:k})}{p(\mathbf{y}_{1:k})} \pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) d\mathbf{x}_{0:k} , \end{aligned} \quad (6.7)$$

where the variables $w_k(\mathbf{x}_{0:k})$ are known as the *unnormalized importance weights*, and are given by

$$w_k(\mathbf{x}_{0:k}) = \frac{p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k}) p(\mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} . \quad (6.8)$$

We often drop the explicit argument $(\mathbf{x}_{0:k})$ for notational convenience, i.e., $w_k \equiv w_k(\mathbf{x}_{0:k})$. Take note that the weights w_k are still a function of the state variable $\mathbf{x}_{0:k}$. We can get rid of the generally unknown or hard to calculate normalizing density $p(\mathbf{y}_{1:k})$ in Equation 6.7 as follows:

$$\begin{aligned} E[\mathbf{g}(\mathbf{x}_{0:k})] &= \frac{1}{p(\mathbf{y}_{1:k})} \int \mathbf{g}(\mathbf{x}_{0:k}) w_k(\mathbf{x}_{0:k}) \pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k}) d\mathbf{x}_{0:k} \\ &= \frac{\int \mathbf{g}(\mathbf{x}_{0:k}) w_k(\mathbf{x}_{0:k}) \pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k}) d\mathbf{x}_{0:k}}{\int p(\mathbf{y}_{1:k} | \mathbf{x}_{0:k}) p(\mathbf{x}_{0:k}) \frac{\pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k})}{\pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k})} d\mathbf{x}_{0:k}} \\ &= \frac{\int \mathbf{g}(\mathbf{x}_{0:k}) w_k(\mathbf{x}_{0:k}) \pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k}) d\mathbf{x}_{0:k}}{\int w_k(\mathbf{x}_{0:k}) \pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k}) d\mathbf{x}_{0:k}} \\ &= \frac{E_\pi[w_k(\mathbf{x}_{0:k}) \mathbf{g}(\mathbf{x}_{0:k})]}{E_\pi[w_k(\mathbf{x}_{0:k})]}, \end{aligned} \quad (6.9)$$

where the notation $E_\pi[\cdot]$ has been used to emphasize that the expectations are taken over the proposal distribution $\pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k})$. Hence, by drawing samples from the proposal distribution function $\pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k})$, we can approximate the expectations of interest by the following estimate:

$$\begin{aligned} E[\mathbf{g}(\mathbf{x}_{0:k})] \approx \tilde{E}[\mathbf{g}(\mathbf{x}_{0:k})] &= \frac{\frac{1}{N} \sum_{i=1}^N \mathbf{g}\left(\mathbf{x}_{0:k}^{(i)}\right) w_k\left(\mathbf{x}_{0:k}^{(i)}\right)}{\frac{1}{N} \sum_{i=1}^N w_k\left(\mathbf{x}_{0:k}^{(i)}\right)} \\ &= \sum_{i=1}^N \tilde{w}_k^{(i)} \mathbf{g}\left(\mathbf{x}_{0:k}^{(i)}\right), \end{aligned} \quad (6.10)$$

where the *normalized importance weights* $\tilde{w}_k^{(i)}$ are given by

$$\tilde{w}_k^{(i)} = \frac{w_k\left(\mathbf{x}_{0:k}^{(i)}\right)}{\sum_{j=1}^N w_k\left(\mathbf{x}_{0:k}^{(j)}\right)} \quad (6.11)$$

$$\equiv w_k^{(i)} / \sum_{j=1}^N w_k^{(j)}, \quad (6.12)$$

where we again used notational shorthand for Equation 6.12.

The estimate of Equation 6.10 is biased since it is itself calculated as the ratio of estimates. However, it is possible to obtain asymptotic convergence and a central limit

theorem for $\tilde{E}[\mathbf{g}(\mathbf{x}_{0:k})]$ under the following assumptions [44, 63]:

1. $\mathbf{x}_{0:k}^{(i)}$ corresponds to a set of i.i.d. samples drawn from the proposal distribution; the support of the proposal distribution includes the support of the true posterior distribution and $E[\mathbf{g}(\mathbf{x}_{0:k})]$ exists and is finite.
2. The expectations of \tilde{w}_k and $\tilde{w}_k \mathbf{g}^2(\mathbf{x}_{0:k})$ over the posterior distribution exist and are finite.

A sufficient condition to verify the second assumption is to have bounds on the variance of $\mathbf{g}(\mathbf{x}_{0:k})$ and on the importance weights [63, 34]. Thus, as N tends to infinity, the posterior density function can be approximated arbitrarily well by the empirical point-mass estimate

$$\hat{p}(\mathbf{x}_{0:k} | \mathbf{y}_{1:k}) = \sum_{i=1}^N \tilde{w}_k^{(i)} \delta \left(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^{(i)} \right) . \quad (6.13)$$

6.2.3 Sequential Importance Sampling

In order to compute a sequential estimate of the posterior distribution at time k without modifying the previously simulated states $\mathbf{x}_{0:k-1}^{(i)}$, proposal distributions of the following form can be used,

$$\pi(\mathbf{x}_{0:k} | \mathbf{y}_{1:k}) = \pi(\mathbf{x}_{0:k-1} | \mathbf{y}_{1:k-1}) \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) . \quad (6.14)$$

Here we are making the assumption that the current state is not dependent on future observations. This is consistent with the 1st order Markov nature of the DSSMs we consider in this thesis (see Chapter 1). It needs to be emphasized that more general proposals, which modify previously simulated trajectories, might be necessary in some scenarios [154]. This issue is, however, beyond the scope of what is presented here. Under our assumptions that the states correspond to a Markov process and that the observations are conditionally independent given the states, we get

$$p(\mathbf{x}_{0:k}) = p(\mathbf{x}_0) \prod_{j=1}^k p(\mathbf{x}_j | \mathbf{x}_{j-1}) \quad \text{and} \quad p(\mathbf{y}_{1:k} | \mathbf{x}_{0:k}) = \prod_{j=1}^k p(\mathbf{y}_j | \mathbf{x}_j) . \quad (6.15)$$

By substituting Equations 6.14 and 6.15 into Equation 6.8, a recursive estimate for the importance weights can be derived as follows:

$$w_k = \frac{p(\mathbf{y}_{1:k} | \mathbf{x}_{0:k}) p(\mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k-1} | \mathbf{y}_{1:k-1}) \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \quad (6.16)$$

$$= \frac{p(\mathbf{y}_{1:k} | \mathbf{x}_{0:k}) p(\mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k-1} | \mathbf{y}_{1:k-1}) \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \times \frac{p(\mathbf{y}_{1:k-1} | \mathbf{x}_{0:k-1}) p(\mathbf{x}_{0:k-1})}{p(\mathbf{y}_{1:k-1} | \mathbf{x}_{0:k-1}) p(\mathbf{x}_{0:k-1})} \quad (6.17)$$

$$= \frac{p(\mathbf{y}_{1:k-1} | \mathbf{x}_{0:k-1}) p(\mathbf{x}_{0:k-1}) p(\mathbf{y}_{1:k} | \mathbf{x}_{0:k}) p(\mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k-1} | \mathbf{y}_{1:k-1}) p(\mathbf{y}_{1:k-1} | \mathbf{x}_{0:k-1}) p(\mathbf{x}_{0:k-1}) \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \quad (6.18)$$

$$= w_{k-1} \frac{p(\mathbf{y}_{1:k} | \mathbf{x}_{0:k}) p(\mathbf{x}_{0:k})}{p(\mathbf{y}_{1:k-1} | \mathbf{x}_{0:k-1}) p(\mathbf{x}_{0:k-1}) \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \quad (6.19)$$

$$= w_{k-1} \frac{\left[\prod_{j=1}^k p(\mathbf{y}_j | \mathbf{x}_j) \right] \left[p(\mathbf{x}_0) \prod_{j=1}^k p(\mathbf{x}_j | \mathbf{x}_{j-1}) \right]}{\left[\prod_{j=1}^{k-1} p(\mathbf{y}_j | \mathbf{x}_j) \right] \left[p(\mathbf{x}_0) \prod_{j=1}^{k-1} p(\mathbf{x}_j | \mathbf{x}_{j-1}) \right] \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \quad (6.20)$$

$$= w_{k-1} \frac{p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{x}_{k-1})}{\pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \quad (6.21)$$

Equation 6.21 provides a mechanism to sequentially update the importance weights, given an appropriate choice of proposal distribution, $\pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})$. The exact form of this distribution is a critical design issue and is usually approximated in order to facilitate easy sampling. The details of this is discussed in the next section. Since we can sample from the proposal distribution,

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) \quad i = 1 \dots N, \quad (6.22)$$

and evaluate the likelihood $p(\mathbf{y}_k | \mathbf{x}_k^{(i)})$ and transition probabilities $p(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1})$, all we need to do is generate a prior set of samples (particles) and iteratively compute the importance weights. The initial set of samples (particles) are equally weighted, i.e.,

$$w_0^{(i)} = \frac{1}{N} \quad i = 1 \dots N. \quad (6.23)$$

This procedure, known as *sequential importance sampling (SIS)*, allows us to obtain the

type of estimates described by Equation 6.10, i.e.,

$$E[\mathbf{g}(\mathbf{x}_k)] \approx \sum_{i=1}^N \tilde{w}_k^{(i)} \mathbf{g}\left(\mathbf{x}_k^{(i)}\right). \quad (6.24)$$

Since we are focusing on the *filtering* and not *smoothing*, we do not need to keep the whole history of the sample trajectories. For this reason $\mathbf{g}(\mathbf{x}_{0:k})$ and $\mathbf{x}_{0:k}$ in Equation 6.10 was replaced by $\mathbf{g}(\mathbf{x}_k)$ and \mathbf{x}_k in Equation 6.24. This is the standard form of the estimator used for sequential importance sampling based filtering. Finally, as N tends to infinity, the posterior *filtering* density can be approximated arbitrarily well by the following empirical point-mass estimate:

$$\hat{p}(\mathbf{x}_k | \mathbf{y}_{1:k}) = \sum_{i=1}^N \tilde{w}_k^{(i)} \delta\left(\mathbf{x}_k - \mathbf{x}_k^{(i)}\right). \quad (6.25)$$

These point-mass estimates can approximate any general distribution arbitrarily well, limited only by the number of particles used and how well the earlier-mentioned importance sampling conditions are met. In contrast, the posterior distribution calculated by the EKF is a minimum-variance Gaussian approximation of the true posterior distribution, which inherently cannot capture complex structure such as multimodalities, skewness, or other higher-order moments.

Choice of Proposal Distribution

The choice of proposal function is one of the most critical design issues in importance sampling algorithms and forms the main issue addressed in this chapter. Doucet shows in [43] that the optimal proposal distribution minimizes the variance of the proposal weights. He goes on to prove [46] that the proposal distribution

$$\pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) \stackrel{\circ}{=} p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k) \quad (6.26)$$

minimizes the variance of the importance weights conditional on $\mathbf{x}_{0:k-1}$ and $\mathbf{y}_{1:k}$. This choice of proposal distribution has also been advocated by other researchers [108, 120, 208] and is referred to in general as the *optimal proposal distribution* [46]. Nonetheless, the

distribution

$$\pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) \doteq p(\mathbf{x}_k | \mathbf{x}_{k-1}), \quad (6.27)$$

(the transition prior) is the most popular choice¹ of proposal distribution [7, 10, 68, 87, 106]. Although it results in higher Monte Carlo variation than the optimal proposal $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k)$, as a result of it not incorporating the most recent observations, it is usually easier to implement [16, 44, 118]. This “ease of implementation” can be seen by substituting Equation 6.27 back into the importance weight expression (Equation 6.21):

$$\begin{aligned} w_k &= w_{k-1} \frac{p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{x}_{k-1})}{\pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \\ &= w_{k-1} \frac{p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{x}_{k-1})}{p(\mathbf{x}_k | \mathbf{x}_{k-1})} \\ &= w_{k-1} p(\mathbf{y}_k | \mathbf{x}_k). \end{aligned} \quad (6.28)$$

In other words, if we choose the transition prior as our proposal distribution to sample from, the importance weights are easily updated by simply evaluating the observation likelihood density $p(\mathbf{y}_k | \mathbf{x}_k)$ for the sampled particle set and multiplying with the previous weights. The transition prior is defined in terms of the probabilistic model governing the system’s states’ evolution (Equation 1.15) and the process noise statistics. For example, if an additive Gaussian process noise model is used, the transition prior is simply,

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{0}), \mathbf{R}_{\mathbf{v}_{k-1}}), \quad (6.29)$$

where $\mathbf{f}(\cdot)$ is the nonlinear process model and $\mathbf{R}_{\mathbf{v}_{k-1}}$ is the process noise covariance (we assumed the process noise \mathbf{v}_k is zero mean).

Even though it is easy and convenient to use the transition prior as proposal distribution, it can possibly lead to problems. Unlike the optimal proposal distribution, the transition prior is not conditioned on the observed data, including the latest observation. As illustrated in Figure 6.1, if we fail to use the latest available information contained in current observation to propose new values for the states, only a few particles will have significant importance weights when their likelihood are evaluated. *It is therefore of paramount*

¹ $A \doteq B$ implies that we *choose* B to approximate A.

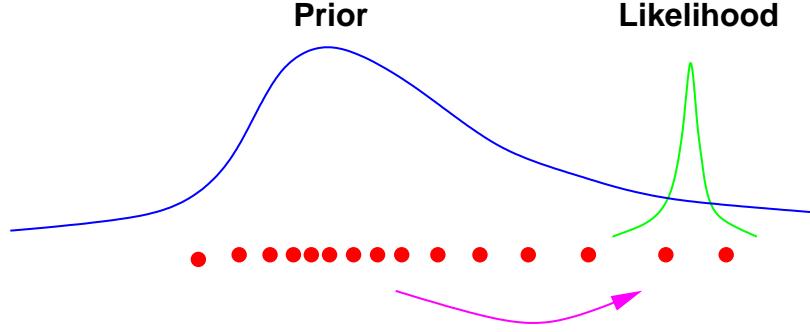


Figure 6.1: Including the most current observation into the proposal distribution, allows us to move the samples in the prior to regions of high likelihood. This is of paramount importance if the likelihood happens to lie in one of the tails of the prior distribution, or if it is too narrow (highly peaked due to low measurement error). This situation often occurs when very high quality, low noise sensors are used.

importance to move the particles towards the regions of high likelihood. This problem also arises when the likelihood function is too narrow compared to the prior. In Section 6.3 we describe several algorithms, based on linearization and the SPKF, to approximate the optimal importance function.

Degeneracy of the SIS Algorithm

The SIS algorithm discussed so far has a serious limitation: *the variance of the importance weights increases stochastically over time* [46]. In order to show this we begin by expanding Equation 6.8,

$$\begin{aligned}
 w_k(\mathbf{x}_{0:k}) &= \frac{p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k})p(\mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \\
 &= \frac{p(\mathbf{y}_{1:k}, \mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \\
 &= \frac{p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})p(\mathbf{y}_{1:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \\
 &\propto \frac{p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}
 \end{aligned} \tag{6.30}$$

The ratio in the last line² of Equation 6.30 is called the *importance ratio* and it can be shown that its variance increases over time. For a proof of this, see [108] and [47]. To

²The proportionality in the last line of the equation follows from the fact that $p(\mathbf{y}_{1:k})$ is a constant.

understand why such a variance increase poses a problem, suppose that we want to sample from the posterior. In that case, we want the proposal density to be very *close* to the posterior density. Closeness is defined (in a probabilistic sense) over the full support of the true posterior [39]. This implies that the best possible (but not practical) choice for the proposal is $\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) = p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$, the true posterior. So, in the *closeness* limit as the proposal distribution approaches the true posterior, we obtain the following results for the mean and variance (see [43] for a proof):

$$E_\pi \left[\frac{p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \right] = 1 , \quad (6.31)$$

and

$$\text{var}_\pi \left[\frac{p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \right] = E_\pi \left[\left(\frac{p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} - 1 \right)^2 \right] = 0 . \quad (6.32)$$

In other words, we want the variance to be close to zero in order to obtain reasonable estimates. Therefore, a variance increase has a harmful effect on the accuracy of the simulations. In practice, the degeneracy caused by the variance increase can be observed by monitoring the importance weights. Typically, what we observe is that, after a few iterations, one of the normalized importance weights tends to 1, while the remaining weights tend to zero. A large number of samples are thus effectively removed from the sample set because their importance weights become numerically insignificant. The next section presents a strategy to reduce (but not fully eliminate) this *degeneration* or depletion of samples.

6.2.4 Mitigating SIS Degeneracy : Resampling

To address the rapid degeneracy of the SIS simulation method, a selection (*resampling*) stage may be used to eliminate samples with low importance weights and multiply samples with high importance weights. A selection scheme associates to each particle $\mathbf{x}_k^{(i)}$ a number of “children”, N_i , such that $\sum_{i=1}^N N_i = N$. Several selection schemes have been proposed in the literature, including *sampling-importance resampling (SIR)* [48, 168, 180], *residual resampling* [118, 78] and *minimum variance sampling* [45].

Sampling-importance resampling (SIR) involves mapping the Dirac random measure

$$\left\{ \mathbf{x}_k^{(i)}, \tilde{w}_k^{(i)}; i = 1, \dots, N \right\}, \quad (6.33)$$

into an equally weighted random measure

$$\left\{ \mathbf{x}_k^{(j)}, \frac{1}{N}; j = 1, \dots, N \right\}. \quad (6.34)$$

In other words, we produce N new samples all with equal weighting $1/N$. This can be accomplished by sampling uniformly from the discrete set $\{\mathbf{x}_k^{(i)}; i = 1, \dots, N\}$ with probabilities $\{\tilde{w}_k^{(i)}; i = 1, \dots, N\}$. Figure 6.2 gives a graphical representation of this process. This procedure effectively replicates the original $\mathbf{x}_k^{(i)}$ particle N_i times (N_i may be zero).

In *residual resampling*, a two step process is used which makes use of SIR. In the first step, the number of children are deterministically set using the floor function,

$$N_i^a = \left\lfloor N \tilde{w}_k^{(i)} \right\rfloor. \quad (6.35)$$

Each $\mathbf{x}_k^{(i)}$ particle is replicated N_i^a times. In the second step, SIR is used to select the remaining

$$\bar{N}_k = N - \sum_{i=1}^N N_i^a \quad (6.36)$$

samples, with new weights

$$\check{w}_k^{(i)} = \bar{N}_k^{-1} \left(\tilde{w}_k^{(i)} N - N_i^a \right). \quad (6.37)$$

These samples form a second set, N_i^b , such that $\bar{N}_t = \sum_{i=1}^N N_i^b$, and are drawn as described previously. The total number of children of each particle is then set to $N_i = N_i^a + N_i^b$. This procedure is computationally cheaper than pure SIR and also has lower sample variance. For this reason residual resampling is used for all subsequent particle filter implementations (See Sections 6.2.5, 6.3.1 and 6.3.2). In general we have found that the specific choice of resampling scheme does not significantly affect the performance of the particle filter.

After the selection/resampling step at time k , we obtain N particles distributed marginally approximately according to the posterior distribution. Since the selection step favors the

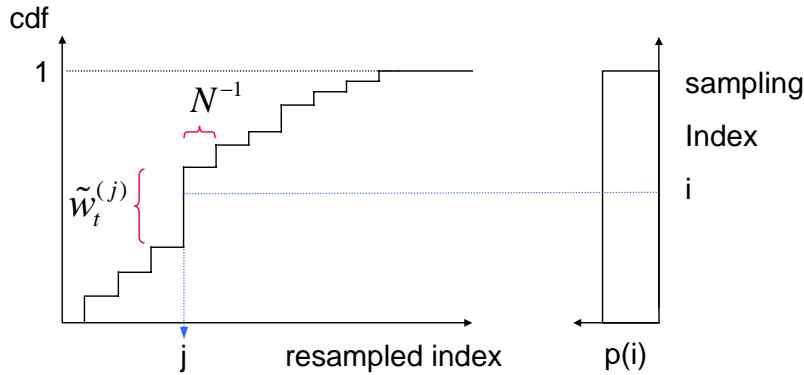


Figure 6.2: Resampling process, whereby a random measure $\{\mathbf{x}_k^{(i)}, \tilde{w}_k^{(i)}; i = 1, \dots, N\}$ is mapped into an equally weighted random measure $\{\mathbf{x}_k^{(j)}, \frac{1}{N}; j = 1, \dots, N\}$. The index i is drawn from a uniform distribution.

creation of multiple copies of the “*fittest*” particles, many particles may end up having no children ($N_i = 0$), whereas others might end up having a large number of children, the extreme case being $N_i = N$ for a particular value i . In this case, there is yet again a severe depletion of samples. Therefore, and additional procedure is often required to introduce sample variety after the selection step without affecting the validity of the approximation they infer. This is achieved by performing a single *Markov chain Monte Carlo*³ (MCMC) step on each particle. The basic idea is that if the particles are already (approximately) distributed according to the posterior $p(\mathbf{x}_k | \mathbf{y}_{1:k})$ (which is the case), then applying a Markov chain transition kernel with the same invariant distribution to each particle results in a set of *new* particles distributed according to the posterior of interest. However, the new particles may move to more interesting areas of the state-space. Details on the MCMC step are given in [189]. For our experimental work in the rest of this chapter we found the need for a MCMC step to be unnecessary. However, this cannot be assumed in general.

6.2.5 The Particle Filter Algorithm

The pseudo-code of a generic particle filter (PF) is presented in Algorithm 16. As discussed in Section 6.2.3, the optimal proposal distribution (which minimizes the variance on the

³For an excellent tutorial paper on MCMC methods for machine learning, see [5].

importance weights) is given by [44, 118, 46],

$$\pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) \stackrel{\circ}{=} p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k) , \quad (6.38)$$

i.e., the true conditional state density given the previous state and current observation. Sampling from this is, of course, impractical for arbitrary densities (recall the motivation for using importance sampling in the first place). Consequently the transition prior is the most popular choice of proposal distribution for the reasons already mentioned in Section 6.2.3 [45], i.e.,

$$\pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) \stackrel{\circ}{=} p(\mathbf{x}_k | \mathbf{x}_{k-1}) . \quad (6.39)$$

For this choice of proposal distribution the generic particle filter is also known as the *Condensation Algorithm* [87]. This form of the particle filter was also used by Gordon, Salmond and Smith in their seminal 1993 paper [68]. They named their approach the *bootstrap (particle) filter*. The effectiveness of this proposal distribution approximation depends on how close the transition prior proposal distribution is to the true posterior distribution. If there is not sufficient overlap, only a few particles will have significant importance weights when their likelihood are evaluated.

In all of the experimental work presented in the rest of this chapter, the generic particle filter will be used as the baseline system for most comparisons.

Algorithm 16 : The Particle Filter (PF)

- *Initialization: $k=0$*
 1. For $i = 1, \dots, N$, draw (sample) particle $\mathbf{x}_0^{(i)}$ from the prior $p(\mathbf{x}_0)$.
- For $k = 1, 2, \dots$
 1. *Importance sampling step*
 - For $i = 1, \dots, N$, sample $\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)})$.
 - For $i = 1, \dots, N$, evaluate the importance weights up to a normalizing constant:

$$w_k^{(i)} = w_{k-1}^{(i)} p(\mathbf{y}_k | \mathbf{x}_k^{(i)}) \quad (6.40)$$

- For $i = 1, \dots, N$, normalize the importance weights: $\tilde{w}_k^{(i)} = w_k^{(i)} / \sum_{j=1}^N w_k^{(j)}$.

2. *Selection step (resampling)*

- Multiply/suppress samples $\mathbf{x}_k^{(i)}$ with high/low importance weights $\tilde{w}_k^{(i)}$, respectively, to obtain N random samples approximately distributed according to $p(\mathbf{x}_k | \mathbf{y}_{1:k})$.
- For $i = 1, \dots, N$, set $w_k^{(i)} = \tilde{w}_k^{(i)} = N^{-1}$.
- (optional) Do a single MCMC (Markov chain Monte Carlo) move step to add further ‘variety’ to the particle set without changing their distribution.

3. *Output:* The output of the algorithm is a set of samples that can be used to approximate the posterior distribution as follows: $\hat{p}(\mathbf{x}_k | \mathbf{y}_{1:k}) = \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)})$. From these samples, any estimate of the system state can be calculated, such as the MMSE estimate,

$$\hat{\mathbf{x}}_k = E[\mathbf{x}_k | \mathbf{y}_{1:k}] \approx \frac{1}{N} \sum_{i=1}^N \mathbf{x}_k^{(i)}.$$

Similar expectations of the function $\mathbf{g}(\mathbf{x}_k)$ (such as MAP estimate, covariance, skewness, etc.) can be calculated as a sample average.

- Note: This particle filter algorithm is also known as the *sampling-importance-resampling* (SIR) particle filter (SIR-PF) [39], the *bootstrap filter* [68], or the *CONDENSATION algorithm* [87].
-

6.2.6 Demonstration of Nonlinear Non-Gaussian Inference

We will now present a brief demonstration of a simple nonlinear non-Gaussian inference problem where Gaussian approximate methods such as the EKF or SPKF fail, due to their inability to represent highly non-Gaussian posteriors. In particular, this problem is bimodal in nature, i.e., there is an inherent and unresolvable ambiguity in determining the true underlying state value that created a certain observation. The DSSM for this problem is given by:

$$x_{k+1} = x_k + v_k \quad (6.41)$$

$$y_k = \alpha x_k^2 + n_k, \quad (6.42)$$

where x_k is the scalar hidden state variable, v_k is an additive Gaussian process noise term with a very small variance ($\sigma_v^2 = 1e-6$), α is a scaling term (for this experiment $\alpha = 10$

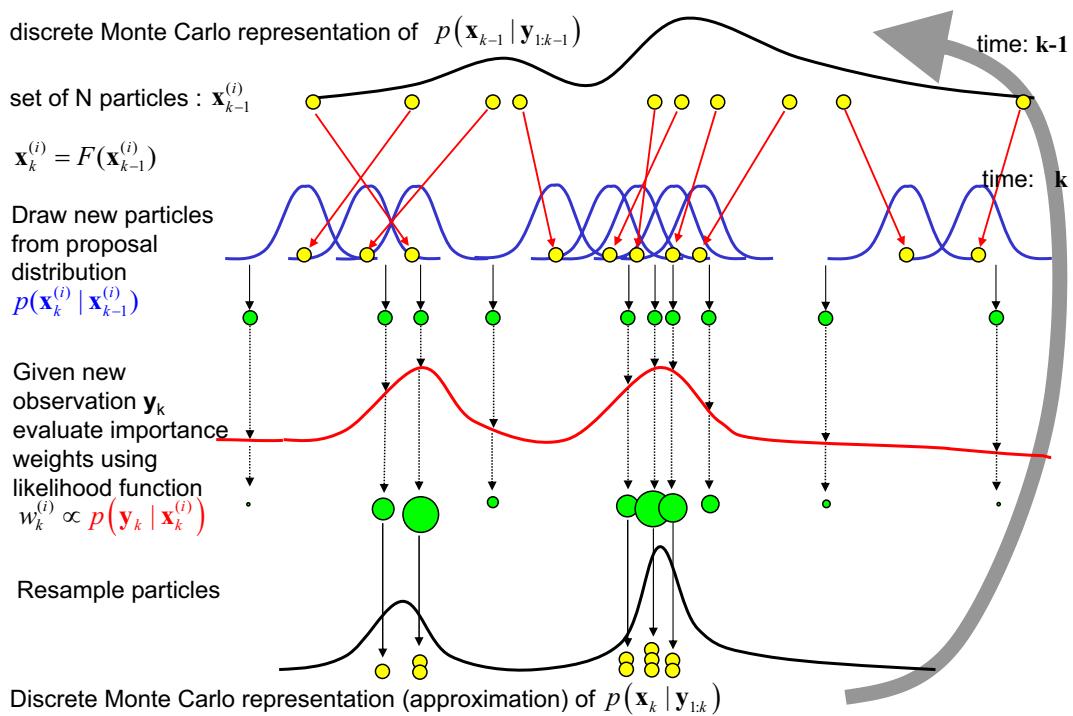


Figure 6.3: Schematic diagram of a generic particle filter (SIR-PF)

was used), y_k is the scalar observation and n_k is an additive Gaussian measurement noise term ($\sigma_n^2 = 1$). Due to the symmetric-around-zero nonlinearity in the observation function (simple quadratic), it is impossible to disambiguate based on the observations alone the sign of the underlying true value of the state variable. In other words, if the true value of the underlying state is $x = x_*$, then the posterior $p(x_k|y_{1:k})$ will be bimodal with the one mode centered at $x = -x_*$ and the other mode at $x = x_*$.

Clearly in this case, a simple Gaussian approximate solution will not suffice to accurately approximate and track the true posterior of the system. In order to test this hypothesis, we generated a sequence of 40 noisy observations based on the DSSM presented above, with the true value of the state set to $x_* = -0.5$. The true posterior state density should thus consist of two Gaussian modes, one situated at $x = -0.5$ and the other at $x = 0.5$. We then used an EKF, a SPKF and a generic particle filter (PF) to try and estimate the underlying state variable, based on the sequence of observations. Figure 6.4 shows the results of this estimation experiment. For each filter we show the time evolution of the estimated posterior state density. All three filters were initialized with a very wide (uncertain) initial distribution which covered all possible true underlying states. The bottom plot shows the results for the particle filter. The filter quickly converges to the true bimodal posterior with peaks at $x = \pm 0.5$. Given a posterior of this nature, using the conditional mean as estimate of the underlying state, i.e.,

$$\hat{x}_k = E[x_k|y_{1:k}] , \quad (6.43)$$

is clearly a bad choice since it will result in an estimate which lies in the middle ($\hat{x} \approx 0$) between the two modes (center of mass) of the distribution. A better estimator will be a MAP estimator which will pick the position of the mode with the “highest likelihood”. For this reason a simple Gaussian approximate estimator such as the EKF or SPKF will result in suboptimal results. This is evident when we look at the posterior distribution estimates generated by the EKF (top plot) and the SPKF (middle plot).

Clearly the EKF and SPKF have to approximate the posterior by a single Gaussian distribution. The EKF starts out with a very broad (uncertain) Gaussian posterior but

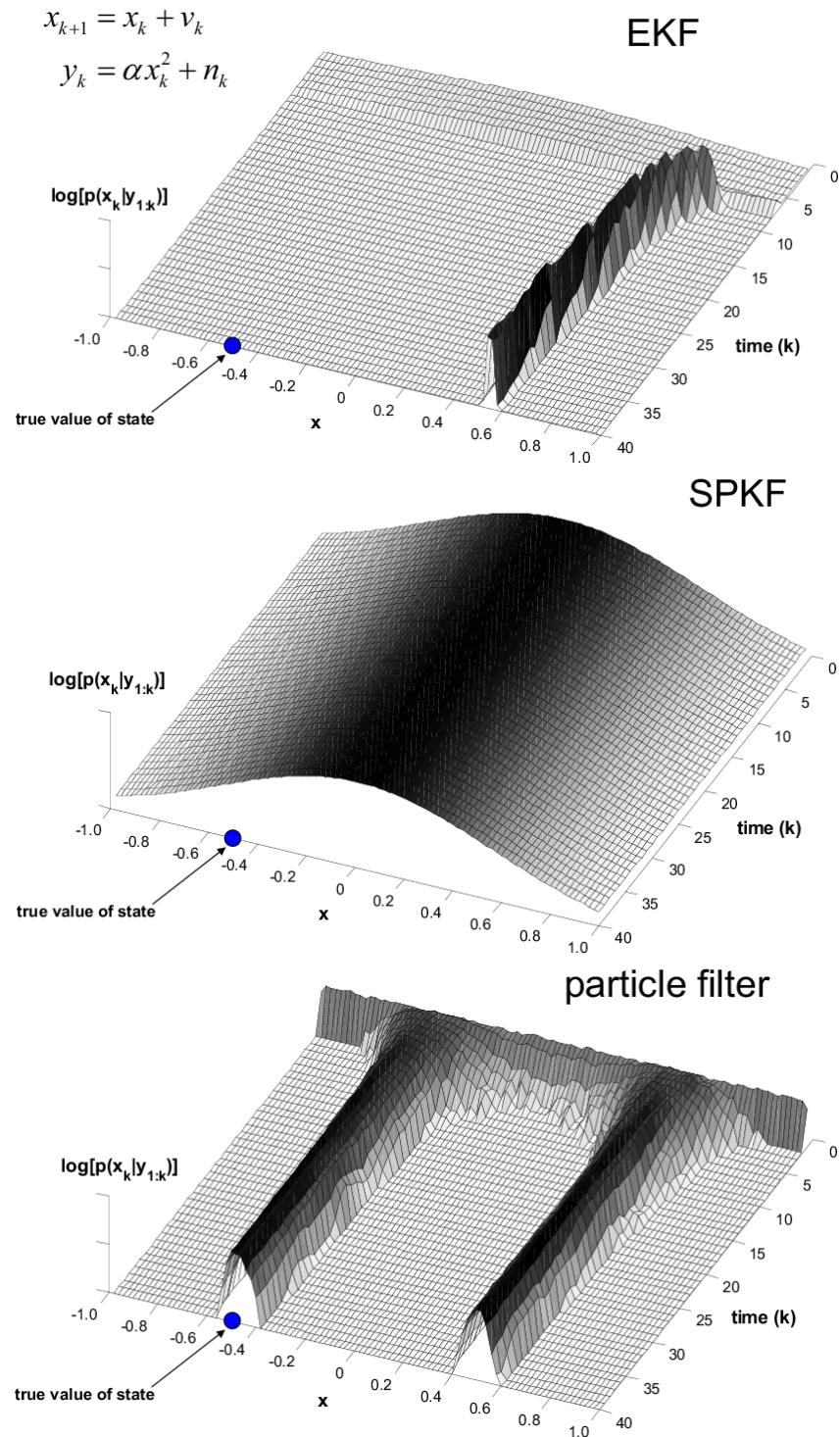


Figure 6.4: Non-Gaussian (bimodal) state estimation demonstration

quickly converges to one of the two modes of the true posterior. Unfortunately it picks the incorrect mode and rapidly becomes highly peaked (overly confident), which will result in a large estimation error as well as a divergent filter, i.e., if the true state value starts to move the EKF will be tracking the wrong mode.

The SPKF on the other hand seems to do a bit better. It correctly represents the large underlying uncertainty implied by the dual modes in the posterior, by generating a single Gaussian posterior that closely (up to third order) matches the true variance of the bimodal distribution. Although the dual hypothesis nature of the true posterior cannot be represented, it compensates for this shortcoming in a robust fashion by not allowing over confidence in any one of the two possible solutions. Here we see an excellent demonstration of the SPKF's superior capability of generating *consistent* estimates: Even though the SPKF has to model the bimodal posterior distribution with a single Gaussian, the variance of this Gaussian does seem to match the variance of the underlying true bimodal distribution. In other words, the single Gaussian includes the support of both the main modes in the true posterior. This property of the SPKF will play an important role in a new proposed method to improve the standard particle filter which will be presented in the next section.

6.3 Improving Particle Filters: Designing Better Proposal Distributions

The success of the particle filter algorithm depends on the validity of the following underlying assumptions:

Monte Carlo (MC) assumption: The Dirac point-mass approximation provides an adequate representation of the posterior distribution.

Importance sampling (IS) assumption: It is possible to obtain samples from the posterior by sampling from a suitable proposal distribution and applying importance sampling corrections.

If any of these conditions are not met, the PF algorithm can perform poorly. The discreteness of the approximation poses a resolution problem. In the resampling stage, any particular sample with a high importance weight will be duplicated many times. As a result, the cloud of samples may eventually collapse to a single sample. This degeneracy will limit the ability of the algorithm to search for lower minima in other regions of the error surface. In other words, the number of samples used to describe the posterior density function will become too small and inadequate. One strategy to address this problem is to implement a Markov chain Monte Carlo (MCMC) step after the selection step as discussed earlier in Section 6.2.4 and covered in more detail in [189, 39]. As already pointed out, this method is only successful if the point-mass posterior approximation is already a good (close) approximation of the true posterior.

Another brute force strategy to overcome this problem is to increase the number of particles. Unfortunately, this might result in a very large (and unnecessary) computational burden. Typically, the number of particles needed during the initial (highly uncertain) stages of any particle filter based inference solution is significantly more than the number needed later on during the estimation process. Initially the posterior is usually quite wide with multiple modes, whereas later on in the estimation process after the observation of a large amount of data, the posterior becomes peaked up around a small number (possibly even a single) of likely hypothesis. The number of needed particles (and hence computational cost) is thus typically much higher earlier on than during the final stages of estimation. Fox [54, 53] exploited this phenomenon to develop an adaptive particle filter where the number of particles used to represent the posterior is adapted on-line. This method, based on calculating the KL divergence between the point-mass posterior distribution and the true posterior and adapting the number of used particles accordingly, resulted in a significant reduction in computational cost for the same level of performance.

As already pointed out in Section 6.2.4, one of the main causes of sample depletion is the failure to move particles to areas of high observational likelihood. This failure stems directly from the most common choice of importance distribution, the transition prior. Even though this proposal loosely satisfies the **importance sampling assumption** stated above, it does this at the cost of not incorporating the latest observation, which

in turn quickly leads to sample depletion which will eventually violate the **Monte Carlo assumption**. An experimental demonstration of this problem associated with the generic particle filter is presented in Section 6.4.3. The crux of improving the performance of particle filters thus lie in designing *better* proposal distributions that not only allow for easy sampling and evaluation of the importance weights (importance sampling assumption), but also address the sample depletion problem (Monte Carlo assumption).

Such an improvement in the choice of proposal distribution over the simple transition prior can be accomplished by moving the particles toward regions of increased likelihood as dictated by the most recent observation \mathbf{y}_k (See Figure 6.1). This can be done by choosing a proposal distribution that is *conditioned* on \mathbf{y}_k . An effective approach to accomplish this, is to use a Kalman filter generated Gaussian approximation of the optimal proposal, i.e,

$$\pi(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) \stackrel{\circ}{=} p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k) \quad (6.44)$$

$$\approx q_{\mathcal{N}}(\mathbf{x}_k | \mathbf{y}_{1:k}) \quad (6.45)$$

where $q_{\mathcal{N}}(\cdot)$ denotes a Gaussian proposal distribution. Note that this approximation consists of two steps: First, the prior state conditioning term of optimal proposal distribution (\mathbf{x}_{k-1}) is integrated out with respect to the posterior state density at time $k - 1$, i.e.,

$$\int p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k) p(\mathbf{x}_{k-1} | \mathbf{y}_{1:k-1}) d\mathbf{x}_{k-1} = p(\mathbf{x}_k | \mathbf{y}_k, \mathbf{y}_{1:k-1}) \quad (6.46)$$

$$= p(\mathbf{x}_k | \mathbf{y}_{1:k}). \quad (6.47)$$

The implicit integration operation of Equation 6.46 effectively averages the optimal proposal distribution with respect to the previous posterior density of the state. This is done since we don't have an exact tractable form for $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k)$ or know what the true value of \mathbf{x}_{k-1} is. All that we know about \mathbf{x}_{k-1} is contained (summarized) in the posterior density $p(\mathbf{x}_{k-1} | \mathbf{y}_{1:k-1})$. This averaging operation can be thought of as an "expected" conditioning with respect to the posterior distribution of \mathbf{x}_{k-1} . Further insight can be found

by expanding the optimal proposal using Bayes rule:

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k) = \frac{p(\mathbf{y}_k, \mathbf{x}_{k-1} | \mathbf{x}_k)p(\mathbf{x}_k)}{p(\mathbf{y}_k, \mathbf{x}_{k-1})} \quad (6.48)$$

$$= \frac{p(\mathbf{y}_k | \mathbf{x}_{k-1}, \mathbf{x}_k)p(\mathbf{x}_{k-1} | \mathbf{x}_k)p(\mathbf{x}_k)}{p(\mathbf{y}_k | \mathbf{x}_{k-1})p(\mathbf{x}_{k-1})} \quad (6.49)$$

$$= \frac{p(\mathbf{y}_k | \mathbf{x}_k)p(\mathbf{x}_k | \mathbf{x}_{k-1})}{p(\mathbf{y}_k | \mathbf{x}_{k-1})}, \quad (6.50)$$

where we made use of the conditional independence of the observations, i.e., $p(\mathbf{y}_k | \mathbf{x}_k, \mathbf{x}_{k-1}) = p(\mathbf{y}_k | \mathbf{x}_k)$, in the last step. If we substitute Equation 6.50 back into the importance weight expression (Equation 6.21), we get the following expression for the optimal importance weights:

$$\begin{aligned} w_k &= w_{k-1} \frac{p(\mathbf{y}_k | \mathbf{x}_k)p(\mathbf{x}_k | \mathbf{x}_{k-1})}{p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k)} \\ &= w_{k-1} \frac{p(\mathbf{y}_k | \mathbf{x}_k)p(\mathbf{x}_k | \mathbf{x}_{k-1})p(\mathbf{y}_k | \mathbf{x}_{k-1})}{p(\mathbf{y}_k | \mathbf{x}_k)p(\mathbf{x}_k | \mathbf{x}_{k-1})} \\ &= w_{k-1} p(\mathbf{y}_k | \mathbf{x}_{k-1}) \\ &= w_{k-1} \int p(\mathbf{y}_k | \mathbf{x}_k)p(\mathbf{x}_k | \mathbf{x}_{k-1}) d\mathbf{x}_k. \end{aligned} \quad (6.51)$$

The multi-dimensional integral in the final line of the expression above is in general intractable for most nonlinear systems, implying that the importance weights for the optimal proposal density cannot be calculated in general. This reason, combined with the difficulty of actually sampling from the optimal proposal distribution, rules out its use in general.

If we compare the expansion of the optimal proposal distribution in Equation 6.50 with the recursive Bayesian expansion⁴ of the proposed approximate proposal

$$p(\mathbf{x}_k | \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k | \mathbf{x}_k)p(\mathbf{x}_k | \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k | \mathbf{y}_{1:k-1})}, \quad (6.52)$$

we see that the relevant densities in the optimal proposal (Equation 6.50) is conditioned on \mathbf{x}_{k-1} , whereas those in the proposed approximate proposal is conditioned on $\mathbf{y}_{1:k-1}$. This makes intuitive sense if we realize that the true state of the system at time \mathbf{x}_{k-1} already includes (summarizes) all of the information emitted in the noisy observation stream $\mathbf{y}_{1:k-1}$.

⁴See Equation 1.11 in Chapter 1 for a derivation of this expansion.

Conditioning on the true posterior value of \mathbf{x}_{k-1} thus decouples the densities from $\mathbf{y}_{1:k-1}$. So, in lieu of knowing the true value of \mathbf{x}_{k-1} , we have to condition on all of the observations received for $k = 1, \dots, k-1$. This information is summarized in the posterior density of the state at time $k-1$, i.e., $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$.

Notice that at this point the proposed approximate proposal density (Equation 6.47) is the true posterior filtering density of the system state conditioned on all the observations. Since accurately approximating this density is the whole point of the sequential Monte Carlo approach, it is not yet available at this point to use as proposal distribution. For this reason we further approximate this density by a tractable single Gaussian distribution as generated by an Gaussian approximate recursive Bayesian estimation framework such as the Kalman filter, i.e.,

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx q_{\mathcal{N}}(\mathbf{x}_k|\mathbf{y}_{1:k}). \quad (6.53)$$

A valid question at this point is: *Why is this double approximation of the optimal proposal density $q_{\mathcal{N}}(\mathbf{x}_k|\mathbf{y}_{1:k})$ a better choice than the simple transition prior $p(\mathbf{x}_k|\mathbf{x}_{k-1})$?*

We already discussed above the fact that the optimal proposal distribution is in general not available in a tractable form which excludes it for practically implementable systems. Furthermore, we also already showed how failing to condition the proposal density on the latest observation can keep particles from moving to areas of high likelihood, possibly causing severe particle depletion. This is always a big concern if the transition prior and the observation likelihood densities are both peaked and have very little support overlap. In such a case the transition prior is clearly a bad choice. Also note that for a large number of systems the effect of process noise is modeled as an additive Gaussian noise term. This in turn implies that the transition prior proposal distribution will also be Gaussian. Given this, the fact that the proposed proposal density $q_{\mathcal{N}}(\mathbf{x}_k|\mathbf{y}_{1:k})$ is *Gaussian* does not by default make it a worse choice than the transition prior. As long as this Gaussian approximate proposal has significant support overlap with the optimal proposal *and* incorporates the latest observation by conditioning on it, it should satisfy the two requirements specified at the beginning of this section and result in an improved particle filter implementation.

A tractable way of generating Gaussian approximate proposal distributions within the particle filter framework, is to use a separate EKF to generate and propagate a Gaussian proposal distribution for each particle, i.e.,

$$q_N(\mathbf{x}_k | \mathbf{y}_{1:k}) = \mathcal{N} \left(\mathbf{x}_k; \boldsymbol{\mu}_k^{(i)}, \mathbf{P}_{\mathbf{x}_k}^{(i)} \right) \quad i = 1, \dots, N \quad (6.54)$$

That is, at time k one uses the EKF equations (Algorithm 1), with the new observed data, to compute the mean and covariance of the importance distribution for each particle from the previous time step $k - 1$. Next, we redraw the i -th particle (at time k) from this new updated distribution. This hybrid algorithm is called the *extended Kalman particle filter* (EKPF) and can be thought of as an adaptive bank of parallel running EKFs, each contributing its own estimate as a component in a very large adaptive mixture approximation of the posterior distribution [40]. The mixing proportion of each component is given by the respective importance weights. The EKPF has been shown to have improved performance on a number of applications [39, 44, 46], but since it makes use of the EKF as a component part, it does in general suffer from the same associated EKF problems, such as filter divergence, etc., as discussed in Chapter 2. For this reason we propose to replace the EKF in this algorithm with a SPKF. In the section we will introduce this new algorithm called the *sigma-point particle filter* (SPKF) and motivate why it is expected to generate better proposal distributions than the EKF.

6.3.1 Sigma-Point Particle Filter

By replacing the EKF with a SPKF⁵, we can more accurately propagate the mean and covariance of the Gaussian approximate proposal distribution for each particle. All of the benefits the SPKF has over the EKF (as presented in prior chapters) is thus leveraged for proposal distribution generation.

Distributions generated by the SPKF will tend to have greater *support overlap* with the true posterior distribution than the overlap achieved by the EKF estimates. Although this statement is made without rigorous proof, the underlying intuition for it is based on the

⁵Specifically we make use of either a *square-root unscented Kalman filter* (SR-UKF) or a *square-root central difference Kalman filter* (SR-CDKF).

fact that the SPKF generated estimates tend to be more *consistent* than those generated by the EKF. By *consistency* we again imply the normal definition, i.e., $\text{var}[\hat{\mathbf{x}}] \geq \text{var}[\mathbf{x}]$, where $\text{var}[\cdot]$ implies the variance (expected second order central moment) of the Gaussian approximate posterior estimate, $\hat{\mathbf{x}}$ is the SPKF generated estimate and \mathbf{x} is the true underlying state. This property was nicely demonstrated by the nonlinear non-Gaussian inference problem presented in Section 6.2.6. As mentioned before, the SPKF also tend to (in general) generate *efficient* estimates, i.e., although the estimated variance is larger than the true variance (to meet the consistency requirement) it is not *excessively large*. In [189, 190] Doucet presents a proof showing that if the proposal distribution used for importance sampling has heavier tails than the underlying true distribution, then the importance weights will be upper bounded resulting in a guarantee of filter convergence. For this reason, using a filter such as the SPKF for proposal generation, that not only incorporates the latest observation, but also generate proposals which tend to have more consistent support overlap with the true posterior, is theoretically better motivated and one could expect better performance. In addition, scaling parameters used for sigma point selection can be optimized to capture certain characteristic of the prior distribution if known, i.e., the algorithm can possibly be modified to work with distributions that have heavier tails than Gaussian distributions such as Cauchy or Student-t distributions⁶.

Algorithm 17 : The Sigma-Point Particle Filter (SPPF)

The new filter that results from using a SPKF for proposal distribution generation within a particle filter framework is called the *sigma-point particle filter* (SPPF):

- *Initialization:* $k=0$
1. For $i = 1, \dots, N$, draw (sample) particle $\mathbf{x}_0^{(i)}$ from the prior $p(\mathbf{x}_0)$.
- For $k = 1, 2, \dots$
1. *Importance sampling step*

⁶Modifying the SPKF to work with non-Gaussian (but still unimodal and symmetric) distributions is still an open and ongoing research question. The form of the Kalman measurement update will have to be modified depending on the nature of the distributions in use.

– For $i = 1, \dots, N$:

(a) Update the Gaussian prior distribution for each particle with the SPKF

:

* Calculate sigma-points for particle, $\mathbf{x}_{k-1}^{a,(i)} = [\mathbf{x}_{k-1}^{(i)} \ \bar{\mathbf{v}}_{k-1} \ \bar{\mathbf{n}}_{k-1}]^T$:

$$\mathcal{X}_{k-1,(0\dots 2L)}^{a,(i)} = \begin{bmatrix} \mathbf{x}_{k-1}^{a,(i)} & \mathbf{x}_{k-1}^{a,(i)} + \gamma\sqrt{\mathbf{P}_{k-1}^{a,(i)}} & \mathbf{x}_{k-1}^{a,(i)} - \gamma\sqrt{\mathbf{P}_{k-1}^{a,(i)}} \end{bmatrix}$$

* Propagate sigma-points into future (time update):

$$\begin{aligned} \mathcal{X}_{k|k-1,(0\dots 2L)}^{x,(i)} &= \mathbf{f}\left(\mathcal{X}_{k-1,(0\dots 2L)}^{x,(i)}, \mathcal{X}_{k-1,(0\dots 2L)}^{v,(i)}, \mathbf{u}_k\right) \\ \bar{\mathbf{x}}_{k|k-1}^{(i)} &= \sum_{j=0}^{2L} w_j^{(m)} \mathcal{X}_{k|k-1,j}^{x,(i)} \\ \mathbf{P}_{k|k-1}^{(i)} &= \sum_{j=0}^{2L} w_j^{(c)} (\mathcal{X}_{k|k-1,j}^{x,(i)} - \bar{\mathbf{x}}_{k|k-1}^{(i)}) (\mathcal{X}_{k|k-1,j}^{x,(i)} - \bar{\mathbf{x}}_{k|k-1}^{(i)})^T \\ \mathcal{Y}_{k|k-1,(0\dots 2L)}^{(i)} &= \mathbf{h}\left(\mathcal{X}_{k|k-1,(0\dots 2L)}^{x,(i)}, \mathcal{X}_{k-1,(0\dots 2L)}^{n,(i)}\right) \\ \bar{\mathbf{y}}_{k|k-1}^{(i)} &= \sum_{j=0}^{2L} W_j^{(m)} \mathcal{Y}_{j,k|k-1}^{(i)} \end{aligned}$$

* Incorporate new observation (measurement update):

$$\begin{aligned} \mathbf{P}_{\mathbf{y}_k \mathbf{y}_k} &= \sum_{j=0}^{2L} w_j^{(c)} [\mathcal{Y}_{k|k-1,j}^{(i)} - \bar{\mathbf{y}}_{k|k-1}^{(i)}] [\mathcal{Y}_{k|k-1,j}^{(i)} - \bar{\mathbf{y}}_{k|k-1}^{(i)}]^T \\ \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} &= \sum_{j=0}^{2L} w_j^{(c)} [\mathcal{X}_{k|k-1,j}^{(i)} - \bar{\mathbf{x}}_{k|k-1}^{(i)}] [\mathcal{Y}_{k|k-1,j}^{(i)} - \bar{\mathbf{y}}_{k|k-1}^{(i)}]^T \\ \mathbf{K}_k &= \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k} \mathbf{P}_{\mathbf{x}_k \mathbf{y}_k}^{-1} \\ \bar{\mathbf{x}}_k^{(i)} &= \bar{\mathbf{x}}_{k|k-1}^{(i)} + \mathbf{K}_k (\mathbf{y}_k - \bar{\mathbf{y}}_{k|k-1}^{(i)}) \\ \mathbf{P}_k^{(i)} &= \mathbf{P}_{k|k-1}^{(i)} - \mathbf{K}_k \mathbf{P}_{\mathbf{y}_k \mathbf{y}_k} \mathbf{K}_k^T \end{aligned}$$

(b) Sample $\mathbf{x}_k^{(i)} \sim q_{\mathcal{N}}(\mathbf{x}_k | \mathbf{y}_{1:k}) = \mathcal{N}\left(\mathbf{x}_k; \bar{\mathbf{x}}_k^{(i)}, \mathbf{P}_k^{(i)}\right)$

– For $i = 1, \dots, N$, evaluate the importance weights up to a normalizing constant:

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p(\mathbf{y}_k | \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)})}{q_{\mathcal{N}}(\mathbf{x}_k^{(i)} | \mathbf{y}_{1:k})} \quad (6.55)$$

– For $i = 1, \dots, N$, normalize the importance weights: $\tilde{w}_k^{(i)} = w_k^{(i)} / \sum_{j=1}^N w_k^{(j)}$.

2. *Selection step (resampling)*

- Multiply/suppress samples $\mathbf{x}_k^{(i)}$ with high/low importance weights $\tilde{w}_k^{(i)}$, respectively, to obtain N random samples approximately distributed according to $p(\mathbf{x}_k | \mathbf{y}_{1:k})$.
- For $i = 1, \dots, N$, set $w_k^{(i)} = \tilde{w}_k^{(i)} = N^{-1}$.
- (optional) Do a single MCMC (Markov chain Monte Carlo) move step to add further 'variety' to the particle set without changing their distribution.

3. *Output:* The output of the algorithm is a set of samples that can be used to approximate the posterior distribution as follows: $\hat{p}(\mathbf{x}_k | \mathbf{y}_{1:k}) = \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)})$. From these samples, any estimate of the system state can be calculated, such as the MMSE estimate,

$$\hat{\mathbf{x}}_k = E[\mathbf{x}_k | \mathbf{y}_{1:k}] \approx \frac{1}{N} \sum_{i=1}^N \mathbf{x}_k^{(i)}.$$

Similar expectations of the function $\mathbf{g}(\mathbf{x}_k)$ (such as MAP estimate, covariance, skewness, etc.) can be calculated as a sample average.

- *General note:* In the resampling stage, not only the particles but also their respective SPKF propagated means and covariances are discarded or duplicated, i.e., we're resampling the whole parallel ensemble of SPKFs. The SPPF presented above makes use of a UKF for proposal generation. Our preferred form however, is a SPPF based around the square-root CDKF (SR-CDKF) which has the numerical efficiency and stability benefits discussed in Section 3.4. The UKF was used in the pseudo-code above in order to simplify the presentation of the algorithm, a similar derivation can be done for the SR-UKF or SR-CDKF [188, 193]. For experimental verification of the superior estimation performance of the SPPF over a standard particle filter, see Section 6.4.3.

The sigma-point particle filter (SPPF) algorithm as presented above and published first in [189, 190, 188], has received quite a bit of interest from a number of different researchers, leading to various direct implementations for a variety of probabilistic inference and machine learning applications. Yong Rui at Microsoft Research implemented a real-time human face tracking system based on a direct implementation of our SPPF algorithm [169]. This serves as an external verification of our algorithm in a real-world inference

system. More detail on this application is presented (with permission from Rui) in Section 6.4.4. Subsequently, our SPPF algorithm has received further wide ranging interest in the literature [196, 117, 36] and has been applied successfully to numerous other real-world inference problems [86, 117].

6.3.2 Gaussian Mixture Sigma-Point Particle Filters

As pointed out at the start of this section, generic particle filters need to use a large number of particles in an attempt to mitigate the sample depletion/degeneracy problem allowing for accurate and robust operation. This can result in very high computational costs for complex problems, since the computational cost of a particle filter scales directly in relation to the number of particles used. In the SPPF section above, we showed how the sample depletion/degeneracy problem can be addressed by moving particles to areas of high likelihood through the use of a SPKF generated proposal distribution. Although the SPPF has large estimation performance benefits over the standard PF (this is demonstrated in Section 6.4), it still incurs a heavy computational burden since it has to run an $\mathcal{O}(L_x^3)$ SPKF for each particle in the posterior state distribution. Here L_x is the dimension of the state. Although the total number of particles needed in the SPPF is typically much less than the requirement for the generic PF, the complexity per particle is higher. There is thus a trade-off between these two factors, which, for some problems⁷ can make the computational cost of the SPPF prohibitively high.

In this section we present a further refinement of the SPPF called the *Gaussian mixture sigma-point particle filter* (GMSPPF) [193, 188]. This filter has equal or better estimation performance when compared to standard particle filters and the SPPF, at a largely reduced computational cost. The GMSPPF combines an *importance sampling* (IS) based measurement update step with a *SPKF based Gaussian sum filter* (GSF) for the time-update and proposal density generation. The GMSPPF uses a finite Gaussian mixture model (GMM) representation of the posterior filtering density, which is recovered from the weighted posterior particle set of the IS based measurement update stage, by means

⁷This is a big consideration for certain real-time applications.

of a *Expectation-Maximization (EM)* algorithm. The EM step either follows directly after the resampling stage of the particle filter, or it can completely replace that stage if a *weighted EM* algorithm is used. The EM or WEM recovered GMM posterior further mitigates the “sample depletion” problem through its inherent “kernel smoothing” nature. The three main algorithmic components used in the GMSPPF are briefly discussed below to provide some background on their use. Then we show how these three components are combined to form the GMSPPF algorithm. Figure 6.5 gives a schematic summary of the main components and data flow of the GMSPPF algorithm.

SPKF based Gaussian mixture approximation

It can be shown [4] than any probability density $p(\mathbf{x})$ can be approximated as closely as desired by a Gaussian mixture model (GMM) of the following form,

$$p(\mathbf{x}) \approx p_{\mathcal{G}}(\mathbf{x}) = \sum_{g=1}^G \alpha^{(g)} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{(g)}, \mathbf{P}^{(g)}) , \quad (6.56)$$

where G is the number of mixing components, $\alpha^{(g)}$ are the mixing weights and $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{(g)}, \mathbf{P}^{(g)})$ are Gaussian density functions with mean vectors $\boldsymbol{\mu}^{(g)}$ and positive definite covariance matrices $\mathbf{P}^{(g)}$. Using the DSSM process and observation functions (Equations 1.1 and 1.2), and assuming that the prior density $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ and system noise densities $p(\mathbf{v}_{k-1})$ and $p(\mathbf{n}_k)$ are also modeled by GMMs, i.e.,

$$\begin{aligned} p_{\mathcal{G}}(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) &= \sum_{g=1}^G \alpha_{k-1}^{(g)} \mathcal{N}\left(\mathbf{x}_{k-1}; \boldsymbol{\mu}_{k-1}^{(g)}, \mathbf{P}_{k-1}^{(g)}\right) \\ p_{\mathcal{G}}(\mathbf{v}_{k-1}) &= \sum_{i=1}^I \beta_{k-1}^{(i)} \mathcal{N}\left(\mathbf{v}_{k-1}; \boldsymbol{\mu}_{v,k-1}^{(i)}, \mathbf{Q}_{k-1}^{(i)}\right) \\ p_{\mathcal{G}}(\mathbf{n}_k) &= \sum_{j=1}^J \gamma_k^{(j)} \mathcal{N}\left(\mathbf{n}_k; \boldsymbol{\mu}_{n,k}^{(j)}, \mathbf{R}_k^{(j)}\right) \end{aligned}$$

the following densities can also be approximated by GMMs: 1) the (*time updated*) *predictive prior density*,

$$p(\mathbf{x}_k | \mathbf{y}_{1:k-1}) \approx p_{\mathcal{G}}(\mathbf{x}_k | \mathbf{y}_{1:k-1}) = \sum_{g'=1}^{G'} \alpha_{k|k-1}^{(g')} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{k|k-1}^{(g')}, \mathbf{P}_{k|k-1}^{(g')}) , \quad (6.57)$$

and 2) the *measurement updated posterior density*,

$$p(\mathbf{x}_k | \mathbf{y}_{1:k}) \approx p_{\mathcal{G}}(\mathbf{x}_k | \mathbf{y}_{1:k}) = \sum_{g''=1}^{G''} \alpha_k^{(g'')} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k^{(g'')}, \mathbf{P}_k^{(g'')}) , \quad (6.58)$$

where $G' = GI$ and $G'' = G'J = GIJ$ (G , I and J are the number of components in the state, process noise, and observation noise GMMs respectively). The predicted and updated Gaussian component means and covariances of $p_{\mathcal{G}}(\mathbf{x}_k | \mathbf{y}_{1:k-1})$ and $p_{\mathcal{G}}(\mathbf{x}_k | \mathbf{y}_{1:k})$, i.e.,

$$\text{predicted: } \left\{ \boldsymbol{\mu}_{k|k-1}^{(g')}, \mathbf{P}_{k|k-1}^{(g')} \right\} , \quad (6.59)$$

and

$$\text{updated: } \left\{ \boldsymbol{\mu}_{k|k}^{(g'')}, \mathbf{P}_{k|k}^{(g'')} \right\} , \quad (6.60)$$

are calculated using the SPKF filter equations implemented as a parallel bank of filters. The mixing weights $\alpha_{k|k-1}^{(g')}$ and $\alpha_k^{(g'')}$ are updated using the same method proposed by Alspach and Sorenson in their seminal 1972 paper on the *Gaussian sum filter* [3], i.e., the predictive weights are given by

$$\alpha_{k|k-1}^{(g')} = \frac{\alpha_{k-1}^{(g)} \beta_{k-1}^{(i)}}{\sum_{g=1}^G \sum_{i=1}^I \alpha_{k-1}^{(g)} \beta_{k-1}^{(i)}} , \quad (6.61)$$

and the measurement updated weights are given by

$$\alpha_k^{(g'')} = \frac{\alpha_{k|k-1}^{(g')} \gamma_k^{(j)} z_k^{(j)}}{\sum_{g'=1}^{G'} \sum_{j=1}^J \alpha_k^{(g')} \gamma_k^{(j)} z_k^{(j)}} , \quad (6.62)$$

where $z_k^{(j)} = p_j(\mathbf{y}_k | \mathbf{x}_k)$ evaluated at $\mathbf{x}_k = \boldsymbol{\mu}_k^{(g')}$ and the current observation \mathbf{y}_k . It is clear that the number of mixing components in the GMM representation grows from G to G' in the predictive (time update) step and from G' to G'' in the subsequent measurement

update step. Over time, this will lead to an exponential increase in the total number of mixing components and must be addressed by a mixing-component reduction scheme. This will be addressed in a later section (see below).

Importance sampling (IS) based measurement update

In the GMSPPF we use the GMM approximate $p_{\mathcal{G}}(\mathbf{x}_k|\mathbf{y}_{1:k})$ generated by the Gaussian sum SPKFs (as described above) as the proposal distribution $\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})$, i.e.,

$$\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) \doteq p_{\mathcal{G}}(\mathbf{x}_k|\mathbf{y}_{1:k}) . \quad (6.63)$$

In Section 6.3 we showed that sampling from such a proposal (which incorporates the latest observation), moves particles to areas of high likelihood which in turn reduces the “sample depletion” problem. Furthermore we use the predictive prior distribution $p_{\mathcal{G}}(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ as a *smoothed* (over prior distribution of \mathbf{x}_{k-1}) evaluation of the $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ term in the importance weight equation (Equation 6.21). This is needed since the GMSPPF represents the posterior (which becomes the prior at the next time step) with a GMM, which effectively smoothes the posterior particle set by a set of Gaussian kernels. One can thus not evaluate the transition prior $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ for a particle at time k conditioned on a particle at time $k-1$, i.e., $t = p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})$, but rather have to calculate this value “averaged” over the prior distribution of all of the $\mathbf{x}_{k-1}^{(i)}$ particles, i.e.

$$\begin{aligned} \tilde{t} &= E \left[p \left(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1} \right) \right] \\ &= \int p \left(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1} \right) p \left(\mathbf{x}_{k-1} | \mathbf{y}_{1:k-1} \right) d\mathbf{x}_{k-1} \\ &= p \left(\mathbf{x}_k^{(i)} | \mathbf{y}_{1:k-1} \right) \\ &\approx p_{\mathcal{G}}(\mathbf{x}_k|\mathbf{y}_{1:k-1}) . \end{aligned} \quad (6.64)$$

Particles are thus randomly sampled from $p_{\mathcal{G}}(\mathbf{x}_k|\mathbf{y}_{1:k})$ after which their respective importance weights are calculated using Equation 6.21 with the terms as defined above.

EM/WEM for GMM recovery

The output of the IS-based measurement update stage is a set of N weighted particles, which in the standard particle filter is *resampled* in order to discard particles with insignificant weights and multiply particles with large weights. The GMSPPF represents the posterior by a GMM which is recovered from the resampled, equally weighted particle set using a standard *Expectation-Maximization* (EM) algorithm, or directly from the *weighted* particles using a *weighted-EM* (WEM) [132] step. Whether a *resample-then-EM* or a *direct-WEM* GMM recovery step is used depends on the particular nature of the inference problem at hand. As discussed in Section 6.2.4, resampling is needed to keep the variance of the particle set from growing too rapidly. Unfortunately, resampling can also contribute to the “particle depletion” problem in cases where the measurement likelihood is very peaked, causing the particle set to collapse to multiple copies of the same particle. In such a case, the *direct-WEM* approach might be preferred. On the other hand, we have found that for certain problems where the disparity (as measured by the KL-divergence) between the true posterior and the GMM-proposal is large, the *resample-then-EM* approach performs better. This issue is still being investigated further.

This GMM recovery step implicitly smoothes over the posterior set of samples, avoiding the “particle depletion” problem, and at the same time the number of mixing components in the posterior is reduced to G , avoiding the exponential growth problem alluded to above. Alternatively, one can use a more powerful “clustering” approach that automatically tries to optimize the model order, i.e., number of Gaussian component densities in the posterior GMM, through the use of some probabilistic cost function such as AIC or BIC [137, 153]. This adaptive approach allows for the complexity of the posterior to change over time to better model the true nature of the underlying process. This is somewhat akin to Fox’s KD-sampling particle filter approach to adaptively change the number of particles needed to accurately model the posterior. In the GMSPPF case however, we are varying the complexity of a parametric model and not the raw number of samples in an empirical (non-parametric) representation. As we will show in Section 6.4.5, we do in fact make use of such an adaptive clustering approach for this step in a specific application of the GMSPPF algorithm to the problem of *mobile robot localization*. Note however that even

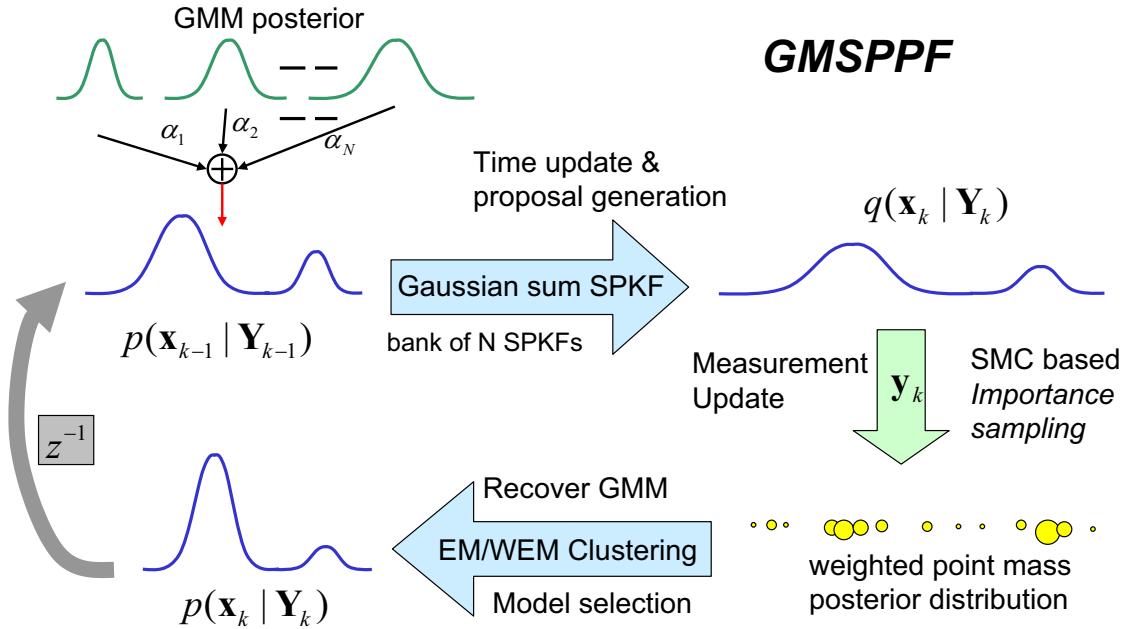


Figure 6.5: Schematic diagram of the Gaussian mixture sigma-point particle filter (GMSPPF).

though we use adaptive clustering in our algorithm, it was not a research issue itself that was addressed within the scope of this thesis. *Adaptive optimal clustering* is a large active research area within the larger machine learning field. We made use of some of the latest state-of-the-art clustering techniques developed by Andrew Moore's group at CMU [153].

Algorithm 18 : The Gaussian Mixture Sigma-Point Particle Filter (GMSPPF)

The full GMSPPF algorithm will now be presented based on the component parts discussed above. As a graphical aid to understand this algorithm, please refer to the schematic presented in Figure 6.5:

A) Time update and proposal distribution generation

At time $k - 1$, assume the posterior state density is approximated by the G -component

GMM

$$\tilde{p}_G(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) = \sum_{g=1}^G \tilde{\alpha}_{k-1}^{(g)} \mathcal{N}\left(\mathbf{x}_{k-1}; \tilde{\boldsymbol{\mu}}_{k-1}^{(g)}, \tilde{\mathbf{P}}_{k-1}^{(g)}\right), \quad (6.65)$$

and the process and observation noise densities are approximated by the following I and J component GMMs respectively

$$p_{\mathcal{G}}(\mathbf{v}_{k-1}) = \sum_{i=1}^I \beta_{k-1}^{(i)} \mathcal{N}\left(\mathbf{v}_{k-1}; \boldsymbol{\mu}_{v,k-1}^{(i)}, \mathbf{Q}_{k-1}^{(i)}\right) \quad (6.66)$$

$$p_{\mathcal{G}}(\mathbf{n}_k) = \sum_{j=1}^J \gamma_k^{(j)} \mathcal{N}\left(\mathbf{n}_k; \boldsymbol{\mu}_{n,k}^{(j)}, \mathbf{R}_k^{(j)}\right) \quad (6.67)$$

Following the GSF approach of [3], but replacing the EKF with a SPKF, the output of a bank of $G'' = GIJ$ parallel SPKFs are used to calculate GMM approximations of $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ and $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ according to the pseudo-code given below. For clarity of notation define

$$g' = g + (i - 1)G, \quad (6.68)$$

noting that references to g' implies references to the respective g and i , since they are uniquely mapped. Similarly define

$$g'' = g' + (j - 1)GI, \quad (6.69)$$

with the same implied unique index mapping from g'' to g' and j . The time-update now proceeds as follows:

1. For $j=1 \dots J$, set $\tilde{p}(\mathbf{n}_k)^{(j)} = \mathcal{N}(\mathbf{n}_k; \boldsymbol{\mu}_{n,k}^{(j)}, \mathbf{R}_k^{(j)})$.
 For $i=1 \dots I$, set $\tilde{p}(\mathbf{v}_{k-1})^{(i)} = \mathcal{N}(\mathbf{v}_{k-1}; \boldsymbol{\mu}_{v,k-1}^{(i)}, \mathbf{Q}_{k-1}^{(i)})$
 For $g=1 \dots G$, set $\tilde{p}(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})^{(g)} = \mathcal{N}(\mathbf{x}_{k-1}; \tilde{\boldsymbol{\mu}}_{k-1}^{(g)}, \tilde{\mathbf{P}}_{k-1}^{(g)})$.
2. For $g'=1 \dots G'$ use the time update step of a SPKF (employing the DSSM process equation $\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}, \mathbf{u}_{k-1})$ and densities $\tilde{p}(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})^{(g)}$ and $\tilde{p}(\mathbf{v}_{k-1})^{(i)}$ from above) to calculate a Gaussian approximate predictive prior density

$$\tilde{p}(\mathbf{x}_k|\mathbf{y}_{1:k-1})^{(g')} = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}^{(g')}, \mathbf{P}_{k|k-1}^{(g')}), \quad (6.70)$$

and update the mixing weights:

$$\alpha_{k|k-1}^{(g')} = \frac{\alpha_{k-1}^{(g)} \beta_{k-1}^{(i)}}{\sum_{g=1}^G \sum_{i=1}^I \alpha_{k-1}^{(g)} \beta_{k-1}^{(i)}}. \quad (6.71)$$

3. For $g'' = 1 \dots G''$, complete the measurement update step of each SPKF (employing the DSSM observation equation $\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k)$, the current observation \mathbf{y}_k , and densities $\tilde{p}(\mathbf{x}_k | \mathbf{y}_{1:k-1})^{(g')}$ and $\tilde{p}(\mathbf{n}_k)^{(j)}$ from above) to calculate a Gaussian approximate posterior density

$$\tilde{p}(\mathbf{x}_k | \mathbf{y}_{1:k})^{(g'')} = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_k^{(g'')}, \mathbf{P}_k^{(g'')}), \quad (6.72)$$

and update the GMM mixing weights:

$$\alpha_k^{(g'')} = \frac{\alpha_{k|k-1}^{(g')} \gamma_k^{(j)} z_k^{(j)}}{\sum_{g'=1}^{G'} \sum_{j=1}^J \alpha_k^{(g')} \gamma_k^{(j)} z_k^{(j)}}, \quad (6.73)$$

where $z_k^{(j)} = p_j(\mathbf{y}_k | \mathbf{x}_k)$ is the observation likelihood evaluated at $\mathbf{x}_k = \boldsymbol{\mu}_k^{(g')}$ and the current observation, \mathbf{y}_k .

The predictive state density is now approximated by the following GMM:

$$p_{\mathcal{G}}(\mathbf{x}_k | \mathbf{y}_{1:k-1}) = \sum_{g'=1}^{G'} \alpha_{k|k-1}^{(g')} \mathcal{N}\left(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}^{(g')}, \mathbf{P}_{k|k-1}^{(g')}\right) \quad (6.74)$$

and the posterior state density (which will only be used as the proposal distribution in the IS-based measurement update step) is approximated by the following GMM:

$$p_{\mathcal{G}}(\mathbf{x}_k | \mathbf{y}_{1:k}) = \sum_{g''=1}^{G''} \alpha_k^{(g'')} \mathcal{N}\left(\mathbf{x}_k; \boldsymbol{\mu}_k^{(g'')}, \mathbf{P}_k^{(g'')}\right). \quad (6.75)$$

B) Measurement update

1. Draw N samples $\{\mathbf{x}_k^{(i)}; i = 1 \dots N\}$ from the GMM proposal distribution $p_{\mathcal{G}}(\mathbf{x}_k | \mathbf{y}_{1:k})$ (Equation 6.75) and calculate their corresponding importance weights:

$$\tilde{w}_k^{(i)} = \frac{p(\mathbf{y}_k | \mathbf{x}_k^{(i)}) p_{\mathcal{G}}(\mathbf{x}_k^{(i)} | \mathbf{y}_{1:k-1})}{p_{\mathcal{G}}(\mathbf{x}_k^{(i)} | \mathbf{y}_{1:k})}. \quad (6.76)$$

2. Normalize the weights:

$$w_k^{(i)} = \frac{\tilde{w}_k^{(i)}}{\sum_{i=1}^N \tilde{w}_k^{(i)}} . \quad (6.77)$$

3. Use one of the following approaches to fit a G -component GMM to the set of weighted particles $\{w_k^{(i)}, \mathbf{x}_k^{(i)}; i = 1 \dots N\}$, representing the updated GMM approximate state posterior distribution at time k , i.e.

$$\tilde{p}_G(\mathbf{x}_k | \mathbf{y}_{1:k}) = \sum_{g=1}^G \tilde{\alpha}_k^{(g)} \mathcal{N}\left(\mathbf{x}_k; \tilde{\boldsymbol{\mu}}_k^{(g)}, \tilde{\mathbf{P}}_k^{(g)}\right) . \quad (6.78)$$

(Option A) First *resample* the set of weighted particles into a new set of N equally weighted particles using any of the efficient resampling techniques⁸ presented in Section 6.2.4, and then apply an *expectation-maximization* (EM) algorithm to this new cloud of samples approximating the posterior density.

(Option B) Directly apply a *weighted expectation-maximization* (WEM) algorithm [132] to the weighted set of particles to recover the GMM posterior.

For both cases, the EM/WEM algorithm is “seeded” by the G means, covariances and mixing weights of the prior state GMM, $p_G(\mathbf{x}_{k-1} | \mathbf{y}_{1:k-1})$, and iterated until a certain convergence criteria (such as relative dataset likelihood increase) is met. Convergence usually occur within 4-6 iterations. Alternatively, as discussed earlier, an adaptive *model-order-selection* EM/WEM approach can be used to adaptively determine the optimal number of Gaussian component densities needed in the posterior GMM to accurately model the posterior cloud of samples.

C) Inference

Once the full posterior state density has been calculated in the previous step, any “optimal” estimate of the underlying system state can be calculated. These include estimates such as the condition mean (MMSE), maximum a-posteriori (MAP), mode, median, etc. to name but a few. As an example, we present the MMSE estimate:

⁸Our preferred method is *residual resampling* [189, 46].

The conditional mean state estimate

$$\hat{\mathbf{x}}_k = E[\mathbf{x}_k | \mathbf{y}_{1:k}] , \quad (6.79)$$

and the corresponding error covariance

$$\hat{\mathbf{P}}_k = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] , \quad (6.80)$$

can be calculated in one of two ways:

(Option A) The estimates can be calculated before the EM/WEM smoothing stage by a direct weighted sum of the particle set,

$$\hat{\mathbf{x}}_k = \sum_{i=1}^N w_k^{(i)} \mathbf{x}_k^{(i)} \quad (6.81)$$

$$\hat{\mathbf{P}}_k = \sum_{i=1}^N w_k^{(i)} (\mathbf{x}_k^{(i)} - \hat{\mathbf{x}}_k)(\mathbf{x}_k^{(i)} - \hat{\mathbf{x}}_k)^T , \quad (6.82)$$

(Option B) or after the posterior GMM has been fitted,

$$\hat{\mathbf{x}}_k = \sum_{g=1}^G \tilde{\alpha}_k^{(g)} \tilde{\boldsymbol{\mu}}_k^{(g)} \quad (6.83)$$

$$\hat{\mathbf{P}}_k = \sum_{g=1}^G \tilde{\alpha}_k^{(g)} \left[\mathbf{P}_k^{(g)} + \left(\tilde{\boldsymbol{\mu}}_k^{(g)} - \hat{\mathbf{x}}_k \right) \left(\tilde{\boldsymbol{\mu}}_k^{(g)} - \hat{\mathbf{x}}_k \right)^T \right] . \quad (6.84)$$

Since $N \gg G$, the first approach (Option A) is computationally more expensive than the second, but possibly generates better (lower variance) estimates, since it calculates the estimates before the implicit resampling of the WEM step. The choice of which method to use will depend on the specifics of the inference problem and is an ongoing research question.

6.3.3 Discussion about differences between SPPF and GMSPPF

One of the major differences between the SPPF and GMSPPF algorithms as presented above, is the difference in computational cost for a specific level of performance. The GMSPPF typically achieves the same level of performance as the SPPF at a significant decrease in computational cost. We will demonstrate this experimentally in the next section.

This difference in computational cost stems directly from the manner in which these two SPKF/SMC hybrid filters approximates the optimal proposal distribution. The SPPF uses a separate SPKF generated Gaussian proposal for each particle, whereas the GM-SPPF uses a single multi-component Gaussian mixture to sample *all* of the particles from. The number of component densities in this GMM is typically much smaller than the number of individual Gaussian proposals needed in the SPPF. Since an individual SPKF is used to propagate and update each of these Gaussians, the GMSPPF has a much lower computational cost.

That being said however, there is still a subtle non-obvious similarity between these two approaches. As mentioned earlier, the SPPF can be interpreted as a stochastic mixture of N parallel running Kalman filters, where N is the number of particles used by the filter. In effect, each individual SPKF generated Gaussian proposal is a component density of a very large implicit Gaussian mixture approximation of the posterior. The mixing weights are given by the respective particle importance weights. When the new set of particles are sampled though during each time-update, each particle in the set is drawn from its own SPKF proposal. In this context, the SPPF's resampling stage can be partially interpreted as a stochastic refitting of the large number of Gaussian component densities in the implicit stochastic GMM approximation of the posterior. This is not a full maximum likelihood fitting of a GMM to the posterior sample set though, since the location of the Gaussian densities and their covariances are not adapted. On the other hand, the GMSPPF approach explicitly models the posterior by a single lower complexity GMM that is fitted to the posterior cloud of particles using a full EM approach that not only adapts the location of the Gaussian components (centroids), but also their covariances. In this manner, a compact (hopefully non-redundant) GMM posterior approximation is found that requires

many fewer parameters to model than the large implied stochastic GMM approximation of the SPPF. This single GMM posterior is then used as a single proposal distribution for all of the new particles.

Except for the computational benefit of the GMSPPF already mentioned, it arguably also provides a closer match to the true optimal proposal distribution through the more powerful general modeling power of a GMM vs. a single Gaussian distribution as used by the SPPF. In general however, the exact difference in performance between these two approaches will depend on the inference problem at hand.

6.4 Experimental Results

We will now present a number of experiments demonstrating the performance benefits of the SPPF and GMSPPF algorithms in comparison with other probabilistic inference filters. Specifically, we will be comparing and contrasting our hybrid algorithms to the generic particle filter which will serve as the baseline reference. Emphasis will be put upon how these new algorithms directly address issues such as particle depletion and/or computational complexity.

The first experiment is a synthetic, scalar, *nonlinear, non-Gaussian time-series estimation* problem that is designed to cause serious sample depletion in the standard sampling-importance-resampling (SIR) particle filter. We will show how the SPPF achieves almost an order of magnitude improvement in estimation accuracy. In the same experiment we also compare the performance of the EKF and SPKF to a variety of sequential Monte Carlo methods, including the generic particle filter and our new proposed SPPF algorithm. Note that the GMSPPF algorithm is not included, due to “historical reasons”⁹, in the set of algorithms to be compared in this experiment. The same inference problem is, however, revisited in Experiment 3 for a direct comparison between the SPPF and GMSPPF algorithms.

⁹The results of this experiment was originally published in [189, 190], more than a year before the GMSPPF algorithm was proposed in [193]. For this reason the GMSPPF was not included in the original experiment set. We do however, repeat the same inference problem in Experiment 3, for a thorough computational cost and estimation performance comparison between the PF, SPPF and GMSPPF. For the same reason, the GMSPPF is not included in the results of Experiment 2.

Experiment 2 applies the SPPF algorithm to an *econometrics problem*, namely *financial options pricing*. We show how using the SPPF approach has certain benefits over the industry standard Black-Scholes [19] pricing strategy to this problem.

We *revisit* the *nonlinear, non-Gaussian time-series estimation* problem of Experiment 1 in Experiment 3, this time focusing however on how the GMSPPF algorithm achieves the same (or better) level of estimation performance as the SPPF, but at a much reduced computational cost. Again we contrast this performance to that of the generic particle filter.

In Experiment 4 we present (with permission) the application of our SPPF algorithm to the real-world vision problem of *human face tracking*. This work was done by Yong Rui at Microsoft Research and published in [169]. The SPPF is compared to Isard and Blake's well known CONDENSATION algorithm [87] which is considered one of the best and most robust solutions for this problem. We will show how the SPPF in fact does even better. This application serves as an external verification of the SPPF algorithm as a viable approach for solving real-world probabilistic inference problems.

Lastly, in Experiment 5, we show how the GMSPPF algorithm can be used to solve the *mobile robot localization* problem. The problem of globally determining the pose (position and orientation relative to a map of the environment) of an autonomous mobile robot has recently started receiving a large amount of interest from the particle filtering community. Most particle filter solutions for this problem require extremely large number of particles ($N \geq 20,000$) to accurately represent the complex posterior density and combat the sample depletion problem. Because of this, the SPPF algorithm is not a viable (practical) solution due to its high computational cost for a large particle set requirement. We will show how the GMSPPF, which has much lower computational cost than the SPPF, is able not only accurately solve this problem, but also robustly outperform the standard SIR particle filter solution.

All of the experiments presented here, except Experiment 4, were performed using Matlab[186] and the *ReBEL Toolkit* [194]. *ReBEL* is an in-house developed toolkit for recursive Bayesian estimation in dynamic state space models, containing all of the algorithms presented in this thesis. See Appendix C for more detail.

6.4.1 Experiment 1 : Scalar Nonlinear, Non-Gaussian Time-series Estimation

For this experiment, a time-series was generated by the following process model:

$$x_{k+1} = 1 + \sin(\omega\pi k) + \phi_1 x_k + v_k \quad (6.85)$$

where v_k is a Gamma random variable¹⁰ modeling the process noise, and $\omega = 4e - 2$ and $\phi_1 = 0.5$ are scalar parameters. Due to the Gamma distributed process noise, the underlying state distribution will be heavy tailed and asymmetric. A non-stationary observation model,

$$y_k = \begin{cases} \phi_2 x_k^2 + n_k & k \leq 30 \\ \phi_3 x_k - 2 + n_k & k > 30 \end{cases} \quad (6.86)$$

is used, with $\phi_2 = 0.2$ and $\phi_3 = 0.5$. The observation noise, n_k , is drawn from a zero-mean Gaussian distribution, i.e., $n_k \sim \mathcal{N}(n; 0, 1e - 5)$. Due to the narrow observation noise distribution, the resulting observation likelihood function will also be highly peaked. This combined with the occasional state outlier produced by the heavy-tailed Gamma process noise distribution will put significant strain on a standard particle filter which uses the standard transition prior as proposal distribution. For this experiment it is of utmost importance to incorporate the latest observation into the proposal distribution in order to move the particle set to areas of high likelihood. For this reason we expect the SPPF algorithm to outperform the standard particle filter.

Given only the noisy observations, y_k , the different filters were used to estimate the underlying clean state sequence x_k for $k = 1 \dots 60$. The experiment was repeated 100 times with random re-initialization for each run. All of the particle filters used 200 particles and residual resampling. For this experiment we used an UKF inside the SPPF with scaling parameters set to $\alpha = 1$, $\beta = 0$ and $\kappa = 2$. These parameters are optimal for the scalar case. Take note however that any other SPKF form can be used inside the SPPF. Table 6.1 summarizes the performance of the different filters. The table shows the means and variances of the mean-square-error (MSE) of the state estimates. Figure 6.6 compares

¹⁰ $v_k \sim \mathcal{G}a(v; 3, 2)$. See [109] for detail about Gamma random variables.

Table 6.1: Scalar nonlinear non-Gaussian state estimation experiment results. This plot shows the mean and variance of the MSE calculated over 100 independent runs.

Algorithm	MSE	
	mean	var
EKF	0.374	0.015
SPKF	0.280	0.012
SIR-PF (generic particle filter)	0.424	0.053
SIR-PF-MCMC (generic particle filter with MCMC step)	0.417	0.055
EKPF (extended Kalman particle filter)	0.310	0.016
EKPF-MCMC (extended Kalman particle filter with MCMC step)	0.307	0.015
<i>SPPF (sigma-point particle filter)</i>	0.070	0.006
<i>SPPF-MCMC (sigma-point particle filter with MCMC step)</i>	0.074	0.008

the estimates generated from a single run of the different particle filters. The superior performance of the *sigma-point particle filter* (SPPF) is clearly evident. Also note how (at least for this experiment) the extra MCMC move step does not have any significant effect on the estimation accuracy of the different filters. Figure 6.7 shows the estimates of the state covariance generated by a stand-alone EKF and SPKF for this problem. Notice how the EKF's estimates are consistently smaller than those generated by the SPKF. This property makes the SPKF better suited than the EKF for proposal distribution generation within the particle filter framework.

This synthetic problem will be revisited in Experiment 3 when comparing the PF, SPPF and GMSPPF algorithms with regard to estimation accuracy and computational complexity.

6.4.2 Experiment 2 : Econometrics - Pricing Financial Options¹¹

Derivatives are financial instruments whose value depends on some basic underlying cash product, such as interest rates, equity indices, commodities, foreign exchange or bonds [82]. An option is a particular type of derivative that gives the holder the right to do something: For example, a *call option* allows the holder to buy a cash product, at a specified date in

¹¹This experiment was done in collaboration with Nando de Freitas who provided valuable insight into financial and econometric issues. These results were published in [189] and [190].

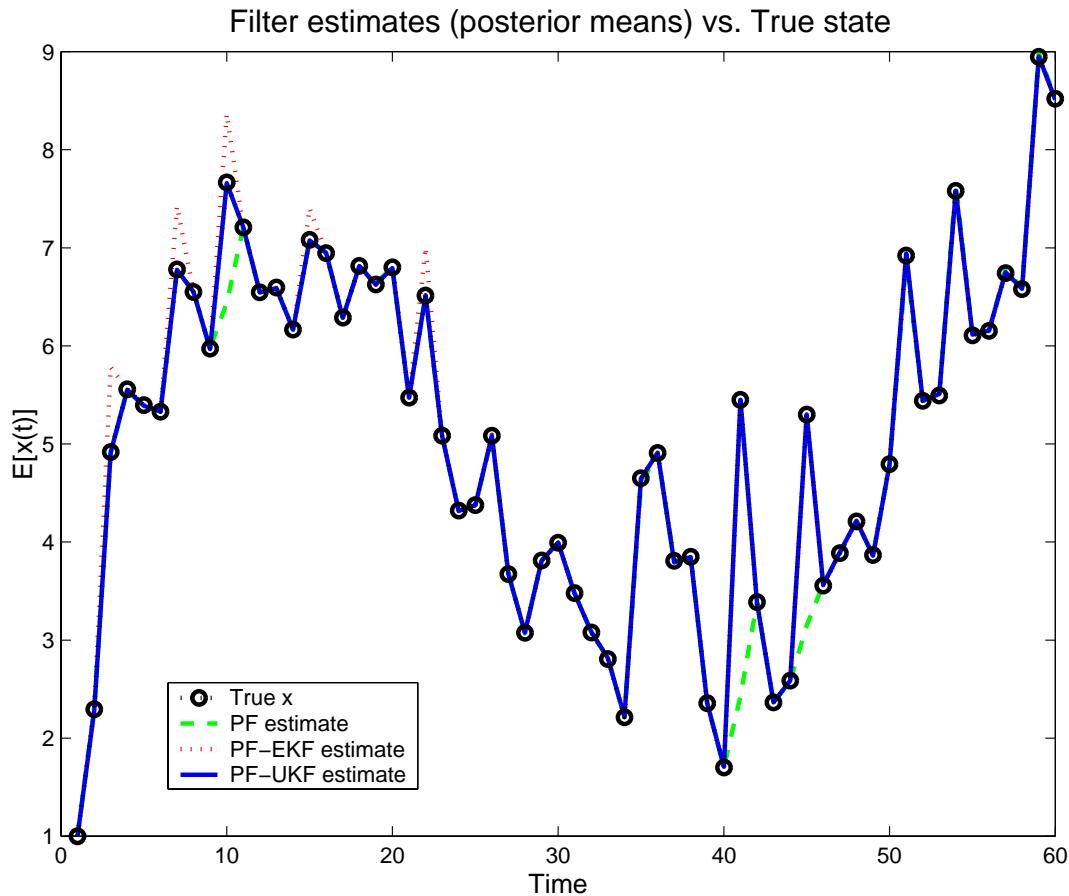


Figure 6.6: Scalar nonlinear non-Gaussian state estimation experiment results: state estimates (The PF-UKF label refers to a SPPF that uses a UKF internally).

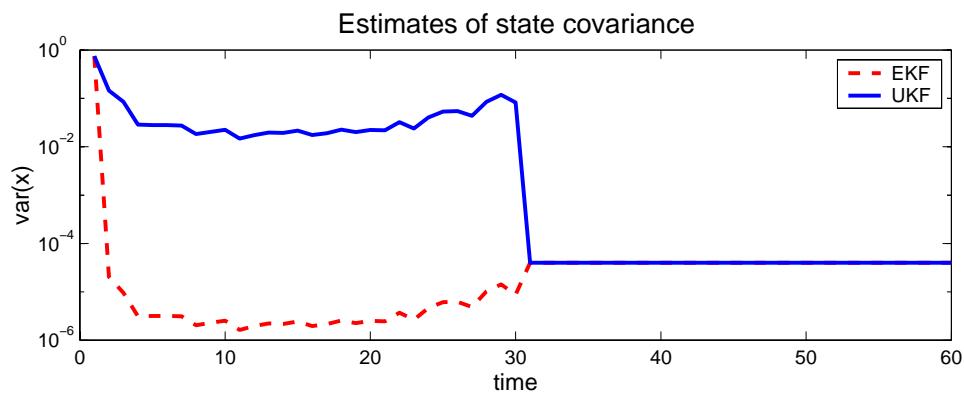


Figure 6.7: EKF and SPKF estimates of the state covariance for scalar nonlinear, non-Gaussian problem.

the future, for a price determined in advance. The price at which the option is exercised is known as the strike price, while the date at which the option lapses is often referred to as the maturity time. *Put options*, on the other hand, allow the holder to sell the underlying cash product at a predetermined price.

The Black Scholes partial differential equation is, essentially, the main industry standard for pricing options [82]. It relates the current value of an option (f) to the current value of the underlying cash product (S), the volatility of the cash product (σ) and the risk-free interest rate (r) as follows:

$$\frac{\partial f}{\partial t} + rS \frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf . \quad (6.87)$$

This basic equation is only valid under several conditions, such as: no risk-less arbitrage opportunities, an instantaneous risk-less portfolio, continuous trading, no dividends, constant volatility and risk-free interest rate. In addition, the cash product is assumed to be dictated by the following geometric Brownian motion model

$$\frac{dS}{S} = \mu dt + \sigma \epsilon dt , \quad (6.88)$$

where μ is the expected return and ϵ corresponds to a random sample from a standardized normal distribution (with mean zero and unit variance). In their seminal work [19], Black and Scholes derived the following solutions for pricing European call and put options:

$$C = S\mathcal{N}_c(d_1) - Xe^{-rt_m}\mathcal{N}_c(d_2) \quad (6.89)$$

$$P = -S\mathcal{N}_c(-d_1) + Xe^{-rt_m}\mathcal{N}_c(-d_2) \quad (6.90)$$

where C denotes the price of a call option, P the price of a put option, X the strike price, t_m the time to maturity, $\mathcal{N}_c(\cdot)$ is the cumulative normal distribution, and d_1 and d_2 are given by

$$d_1 = \frac{\ln(S/X) + (r + \sigma^2/2)t_m}{\sigma\sqrt{t_m}} \quad (6.91)$$

$$d_2 = d_1 - \sigma\sqrt{t_m} \quad (6.92)$$

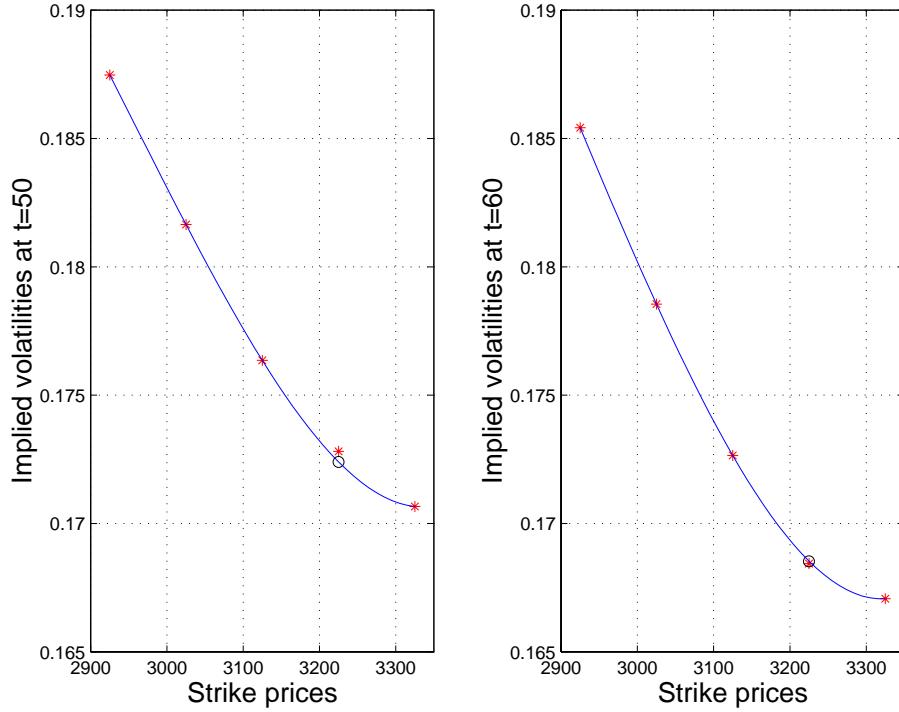


Figure 6.8: Volatility smile indicating that an option on the FTSE-100 index was over-priced. The option value 10 days later confirmed this hypothesis. Estimates obtained with a particle filter [*], 4-th order polynomial fit [—] and hypothesized volatility [o].

The volatility (σ) is usually estimated from a small moving window of data over the most recent 50 to 180 days [82]. The risk-free interest rate (r) is often estimated by monitoring interest rates in the bond markets. In our approach, which follows from [146], we use a DSSM representation to model the system given by Equations 6.89 and 6.90. We treat r and σ as the hidden states and C and P as the output observations. t_m and S are treated as known control signals (input observations). We believe that this approach is better since it constitutes a more natural way of dealing with the sequential behavior and non-stationarity of the data. In the end, we are able to compute daily complete probability distributions for r and σ and to decide whether the current value of an option in the market is being either over-priced or under-priced.

Typically, options on a particular equity and with the same exercise date are traded with several strike prices. For example, in our experiments, we used five pairs of call and put option contracts on the British FTSE-100 index (from February 1994 to December

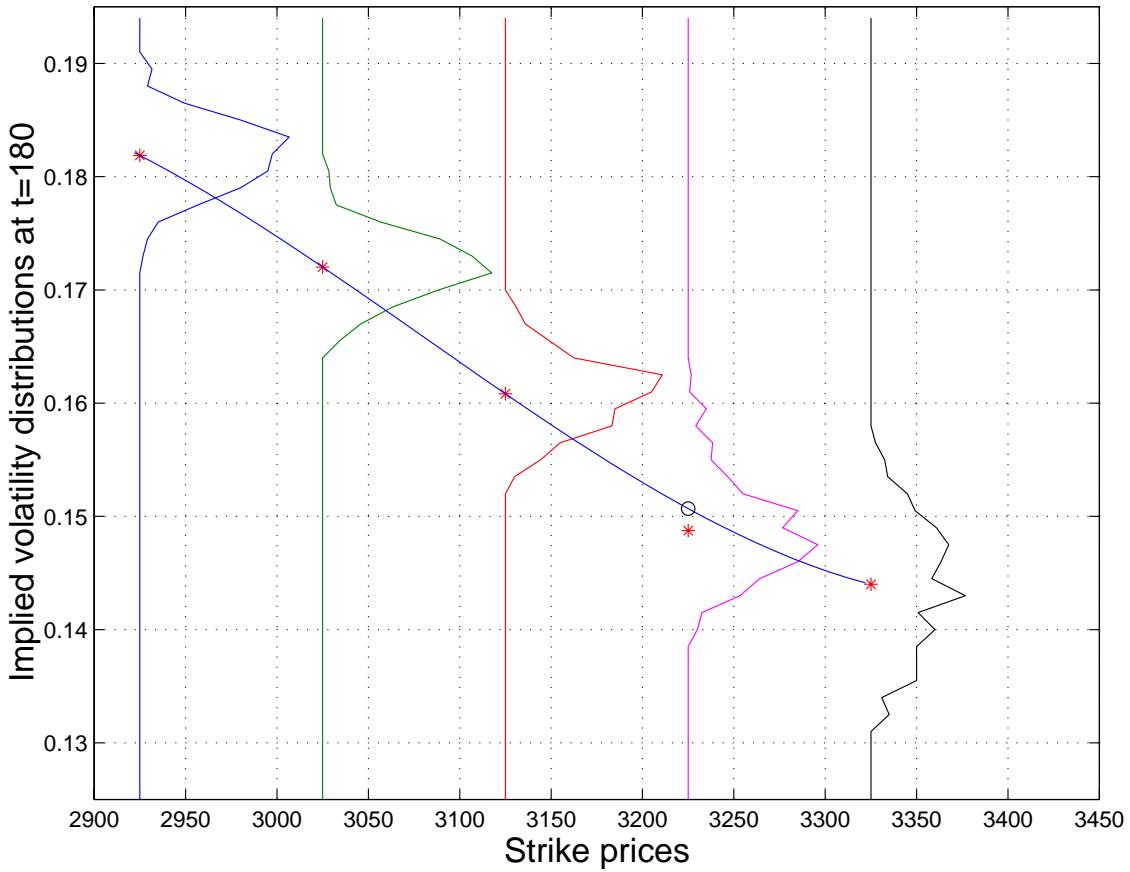


Figure 6.9: Probability smile for options on the FTSE-100 index (1994). Although the volatility smile indicates that the option with strike price equal to 3225 is under-priced, the shape of the probability gives us a warning against the hypothesis that the option is under-priced. Posterior mean estimates obtained with Black-Scholes model and particle filter [*], 4-th order polynomial fit [—] and hypothesized volatility [o].

1994) to evaluate the pricing algorithms. For each option on this set one can estimate a different volatility. By plotting the Black-Scholes estimates of the volatilities against their respective strike prices, we obtain a curve which is known as the *volatility smile* [82]. A well known pricing strategy is to leave one of the options out and then determine the volatility smile given by the other options. If the option that was left out is below the curve, it could mean that it is under-priced by the market.

Figure 6.8 shows an example of this phenomenon obtained by tracking 5 pairs of call and put option contracts on the FTSE-100 index (1994) with a particle filter. On the 50th day, option 4 seems to be over-priced. The state of this option 10 days later confirms

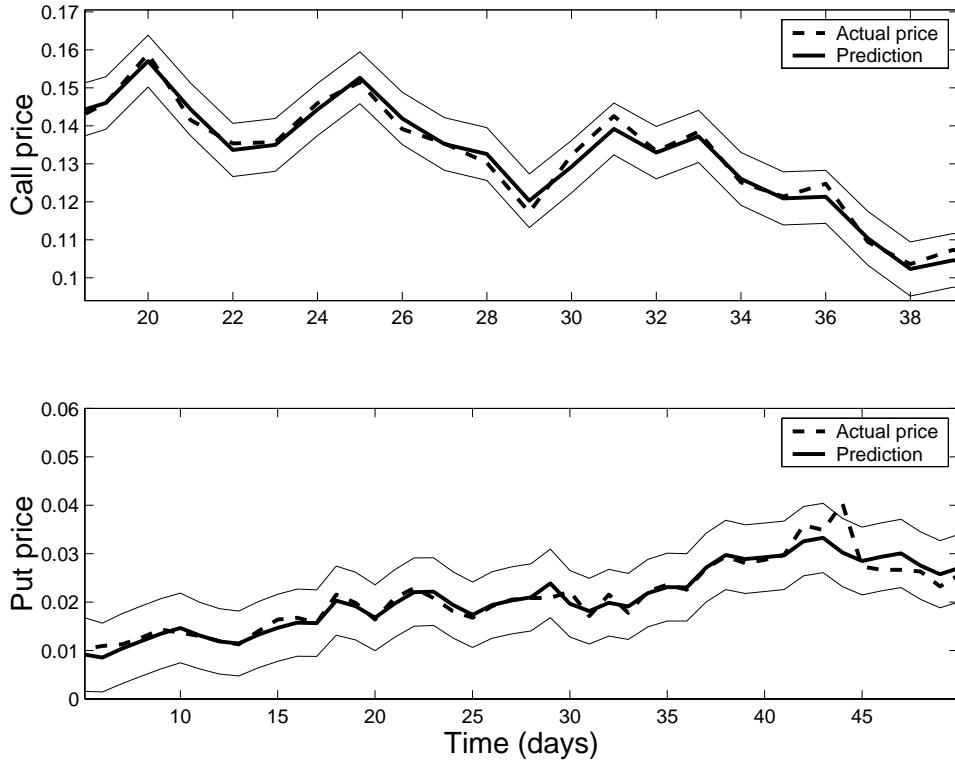


Figure 6.10: SPPF one-step-ahead predictions on the call and put option's prices with confidence intervals (2 standard deviations).

this hypothesis. However, depending on the state of the particular equity, some options might remain under-priced or over-priced during their entire life-time. For example, if an option on a company product seems to be over-priced according to its volatility smile, but investors know that the company is being bought by a larger company with better management, the option price will remain higher than the smile prediction [73].

In the sequential Monte Carlo framework, we can improve this trading strategy. Instead of plotting a volatility smile, we plot a *probability smile*. That is, we can plot the probability density function of each implied volatility against their respective strike prices, as shown in Figure 6.9. This plot, clearly, conveys more information than a simple plot of the posterior mean estimates.

The type of predictions obtained with the *sigma-point particle filter* were very close to the measured data as evidenced by Figure 6.10. Figure 6.11 shows the estimated volatility and interest rate for a contract with a strike price of 3225. Plots of the evolution of the

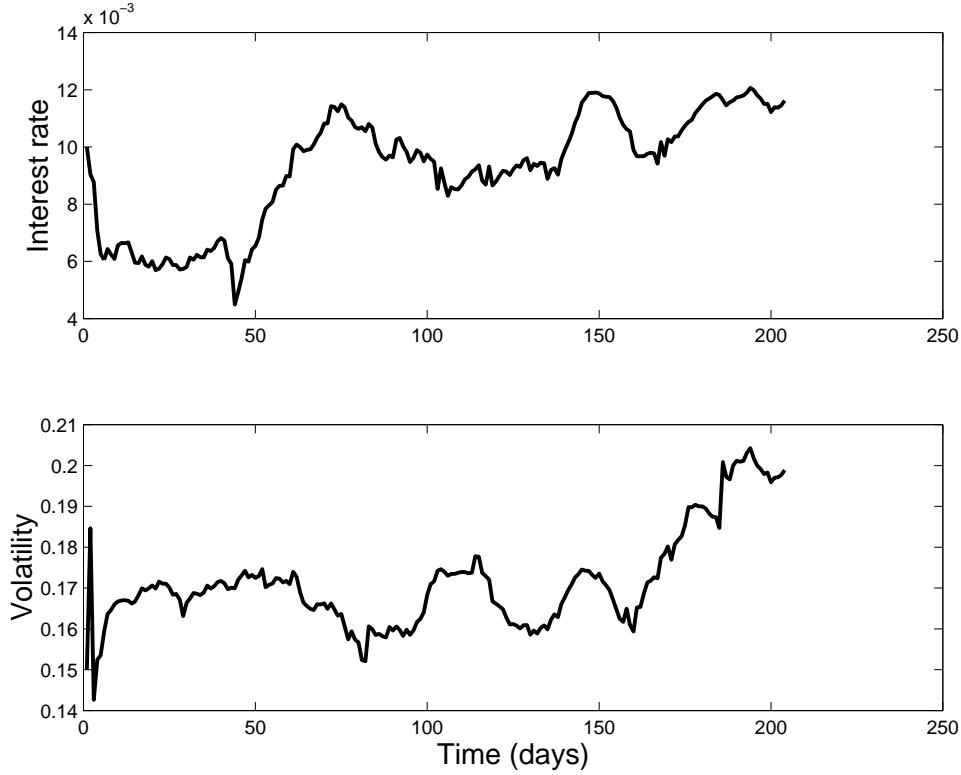


Figure 6.11: Estimated interest rate and volatility.

probability distributions of the interest rate and volatility are depicted in Figures 6.12 and 6.13.

In Table 6.2, we compare the one-step-ahead normalized square errors obtained with each method on a pair of options with strike price 2925. The normalized square errors are defined as follows:

$$NSE_C = \sqrt{\sum_t (C_t - \hat{C}_t)^2} \quad (6.93)$$

$$NSE_P = \sqrt{\sum_t (P_t - \hat{P}_t)^2}, \quad (6.94)$$

where \hat{C}_t and \hat{P}_t denotes the one-step-ahead predictions of the call and put prices. The square errors were only measured over the last 100 days of trading, so as to allow the algorithms to converge. The experiment was repeated 100 times and we used 100 particles in each particle filter.

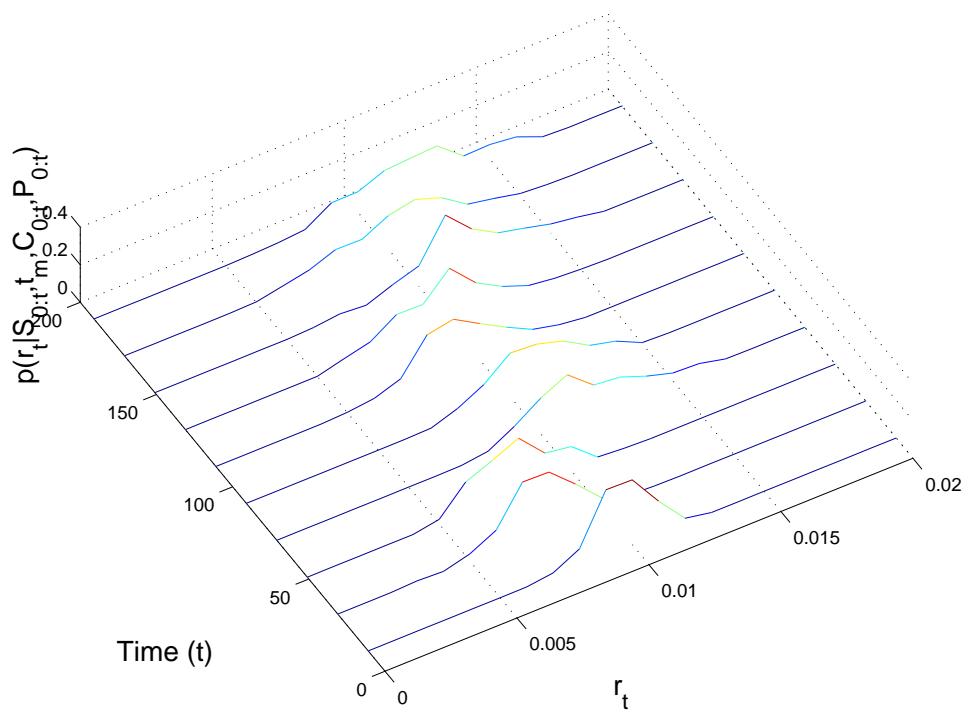


Figure 6.12: Probability distribution of the implied interest rate.

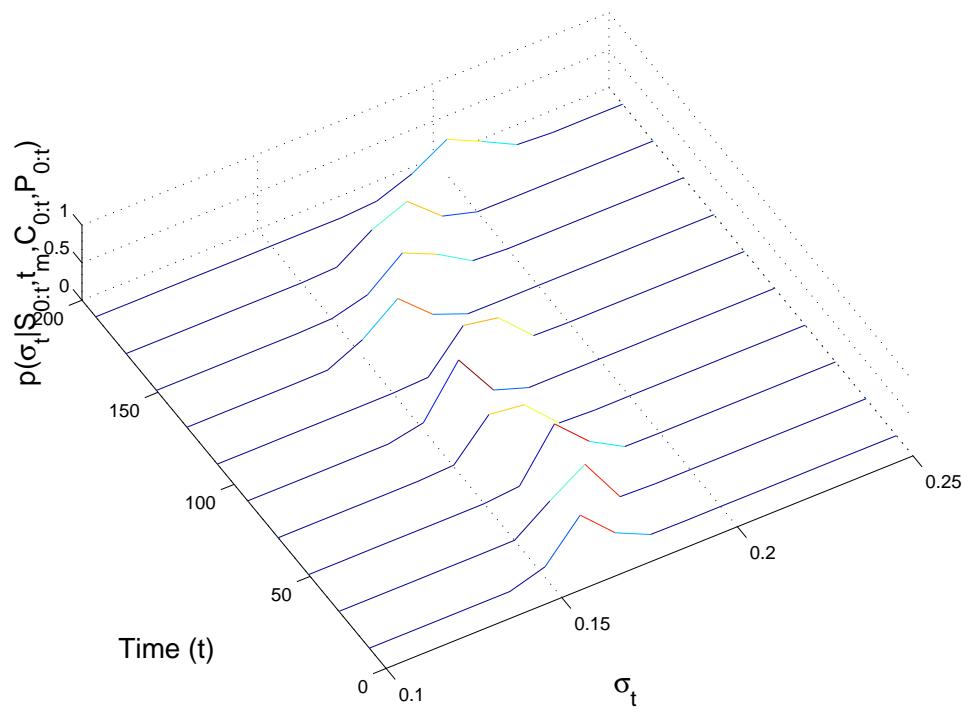


Figure 6.13: Probability distribution of the implied volatility.

Table 6.2: One-step-ahead normalized square errors over 100 runs. The trivial prediction is obtained by assuming that the price on the following day corresponds to the current price.

Option Type	Algorithm	Mean NSE
Call	trivial	0.078
	EKF	0.037
	SPKF	0.037
	SIR-PF (generic)	0.037
	EKPF (PF with EKF proposal)	0.009
	<i>SPPF</i>	0.009
Put	trivial	0.035
	EKF	0.023
	SPKF	0.023
	SIR-PF (generic)	0.023
	EKPF (PF with EKF proposal)	0.008
	<i>SPPF</i>	0.008

In this example, both the EKF and SPKF approaches to improving the proposal distribution, i.e., EKPF and SPPF algorithms, lead to a significant improvement over the standard particle filter. The main advantage of the SPKF based SPPF over the EKF based EKPF, is the ease of implementation, which avoids the need to analytically differentiate the Black-Scholes equations.

For reasons already laid out in the introduction to this section, the GMSPPF was not available for comparison when this experiment was performed and the results published in [189, 190, 77]. Since this specific inference problem is inherently off-line by nature and has a relatively low computational cost (low state and observation dimension as well as low number of particles needed), the extra implementational complexity of the GMSPPF is not necessarily justified. For this reason the experiment was not repeated with the GMSPPF. In the next experiment we do however directly compare the SPPF and GMSPPF.

6.4.3 Experiment 3 : Comparison of PF, SPPF & GMSPPF on nonlinear, non-Gaussian state estimation problem

For this experiment we revisit the nonlinear, non-Gaussian, scalar time series estimation problem as presented in Experiment 1. The specific aim this time however is to directly compare and contrast the performance of the SPPF and GMSPPF algorithms, with specific focus on computational complexity. As a baseline reference we will also show the relative performance of the generic SIR-PF algorithm.

Just to recap, a scalar time series was generated by the following process model:

$$x_k = \phi_1 x_{k-1} + 1 + \sin(\omega\pi(k-1)) + v_k , \quad (6.95)$$

where v_k is a Gamma $\mathcal{G}a(3, 2)$ random variable modeling the process noise, and $\omega = 0.04$ and $\phi_1 = 0.5$ are scalar parameters. A nonstationary observation model,

$$y_k = \begin{cases} \phi_2 x_k^2 + n_k & k \leq 30 \\ \phi_3 x_k - 2 + n_k & k > 30 \end{cases} \quad (6.96)$$

is used, with $\phi_2 = 0.2$ and $\phi_3 = 0.5$. The observation noise, n_k , is drawn from a Gaussian distribution $\mathcal{N}(n_k; 0, 10^{-5})$. Figure 6.15 shows a plot of the hidden state and noisy observations of the time series. Given only the noisy observations y_k , the different filters were used to estimate the underlying clean state sequence x_k for $k = 1 \dots 60$. The experiment was repeated 100 times with random re-initialization for each run in order to calculate Monte-Carlo performance estimates for each filter. All the particle filters used 500 particles. This is in contrast to Experiment 1 where we used 200 particles per filter. The increased number of particles will magnify the difference in computational cost between the algorithms. With the increased number of particles we also expect a lower overall estimation error for all the algorithms. Where applicable (SIR-PF and SPPF only) residual resampling were used. For the GMSPPF we made use of the direct *weighted EM* option to recover the GMM posterior after the IS based measurement update step. We found this approach to have slightly better performance than the *resample-then-EM* option. For the IS based measurement update step we used the same number of particles (200) as for the

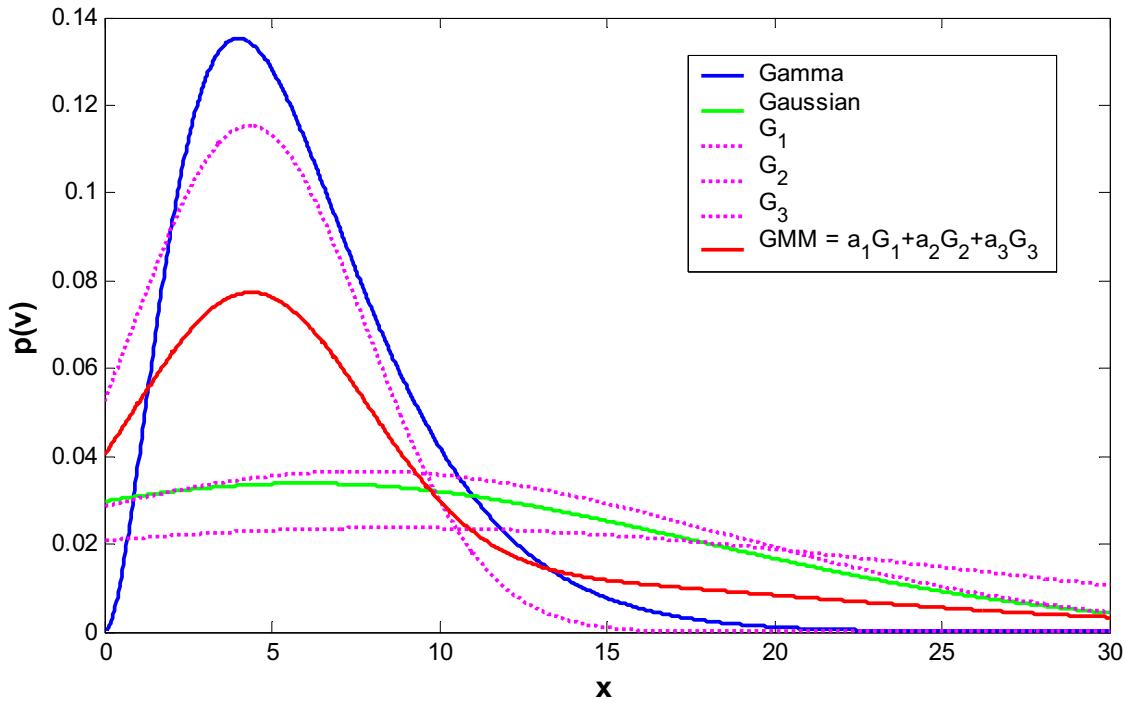


Figure 6.14: GMM approximation of heavy tailed, asymmetric Gamma distributed noise distribution. The blue curve is the Gamma distribution and the green curve is a single Gaussian approximation (same mean and covariance) of the Gamma distribution. The dotted magenta curves show the shape and position of the three Gaussian components of a GMM approximation of the Gamma distribution. Finally, the red curve shows the resulting GMM distribution built up by a linear weighted sum of the green component densities. The GMM was fitted to the Gamma distribution using an expectation-maximization (EM) approach.

generic particle filter.

Two different GMSPPF filters were compared: The first, GMSPPF (5-1-1), uses a 5-component GMM for the state posterior, and 1-component GMMs for both the process and observation noise densities. The second, GMSPPF (5-3-1), uses a 5-component GMM for the state posterior, a 3-component GMM to approximate the “heavy tailed” Gamma distributed process noise and a 1-component GMM for the observation noise density. The process noise GMM was fitted to simulated Gamma noise samples with an EM algorithm. Figure 6.14 shows how an asymmetrical heavy tailed Gamma distribution can be approximated by a 3-component GMM. Both GMSPPFs use Inference Method 1 (Equations 6.81 and 6.82) to calculate the state estimates.

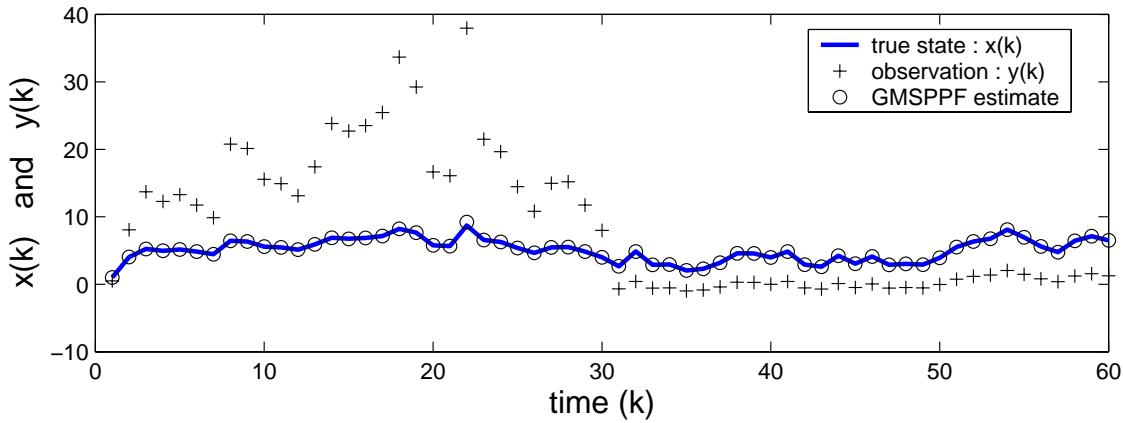


Figure 6.15: Non-stationary, nonlinear, non-Gaussian time series estimation experiment: Plot of true state, noisy observations and filter estimates.

Table 6.3 summarizes the performance of the different filters averaged over 100 randomly initialized Monte Carlo runs. The table shows the means and variances of the mean-square-error of the state estimates as well as the average processing time in seconds of each filter. The reason why the standard PF performs so badly on this problem is due to the highly peaked likelihood function of the observations (arising from the small observation noise variance) combined with the spurious jumps in the state due to the heavy tailed process noise. This causes severe “sample depletion” in the standard PF that uses the transition prior $p(x_k|x_{k-1})$ as proposal distribution. As discussed in Sections 6.3 and 6.3.1, the SPPF and GMSPPF addresses this problem by moving the particles to areas of high likelihood through the use of a SPKF derived proposal distributions, resulting in a drastic improvement in performance. Unfortunately, for the SPPF, this comes at a highly increased computational cost. Since the processing time for each algorithm (on the same CPU) is directly related to its computational complexity, Table 6.3 clearly indicates that the GMSPPF algorithms has the same order of computational complexity as the SIR-PF, but much better estimation performance. Conversely, for the same level of estimation performance as the SPPF, the GMSPPF realizes this at a much lower computational cost. The best performance is achieved by the 5-3-1 GMSPPF that better models the non-Gaussian nature of the process noise distribution.

Table 6.3: Non-stationary, nonlinear, non-Gaussian time series estimation experiment: Estimation results averaged over 100 Monte Carlo runs.

Algorithm	MSE (mean)	MSE (var)	Time (s)
SIR-PF (generic)	0.2517	4.9e-2	1.70
SPPF	0.0049	8.0e-5	35.6
<i>GMSPPF (5-1-1)</i>	0.0036	5.0e-5	1.04
<i>GMSPPF (5-3-1)</i>	0.0004	3.0e-6	2.03

6.4.4 Experiment 4 : Human Face Tracking

This section reports on the use of the SPPF for a computer vision based human face tracking system. The system was developed by Yong Rui and his group at Microsoft Research [169] and directly utilizes our SPPF algorithm in an efficient C++ implementation with specific application to a automated camera management system for a lecture room environment [121]. This application of our SPPF algorithm serves as an external verification of its viability as an approach for solving real-world probabilistic inference problems and is presented here with permission from Yong Rui. In the description of the human face tracking experiment that follows, we rely heavily on the exposition given by Rui and Chen in [169].

Robust human head tracking in a cluttered environment has many real-world applications such as automated camera pointing and speaker tracking in a lecture hall environment, immersive human-computer interaction for virtual-reality (VR) systems and adaptive microphone beam-forming for speech enhancement applications to name but a few [121]. The general shape of a human head can be modeled by a 1:1.2 aspect ratio ellipse centered at the visual center of mass of the image of head (See Figure 6.16 for a schematic). This simple parametric curve describing the outline of the head allows us to build a likelihood model $p(\mathbf{z}_k|\mathbf{x}_k)$ for a given observation \mathbf{z}_k (video frame) conditioned on the underlying 2D position (x-y) of the center of the persons head in the frame. Unfortunately this likelihood model (which is shown later) is highly nonlinear, increasing the difficulty of accurately tracking the head-ellipse from frame to frame. Even small changes in the parametric curve space could result in large changes in the observation likelihood.

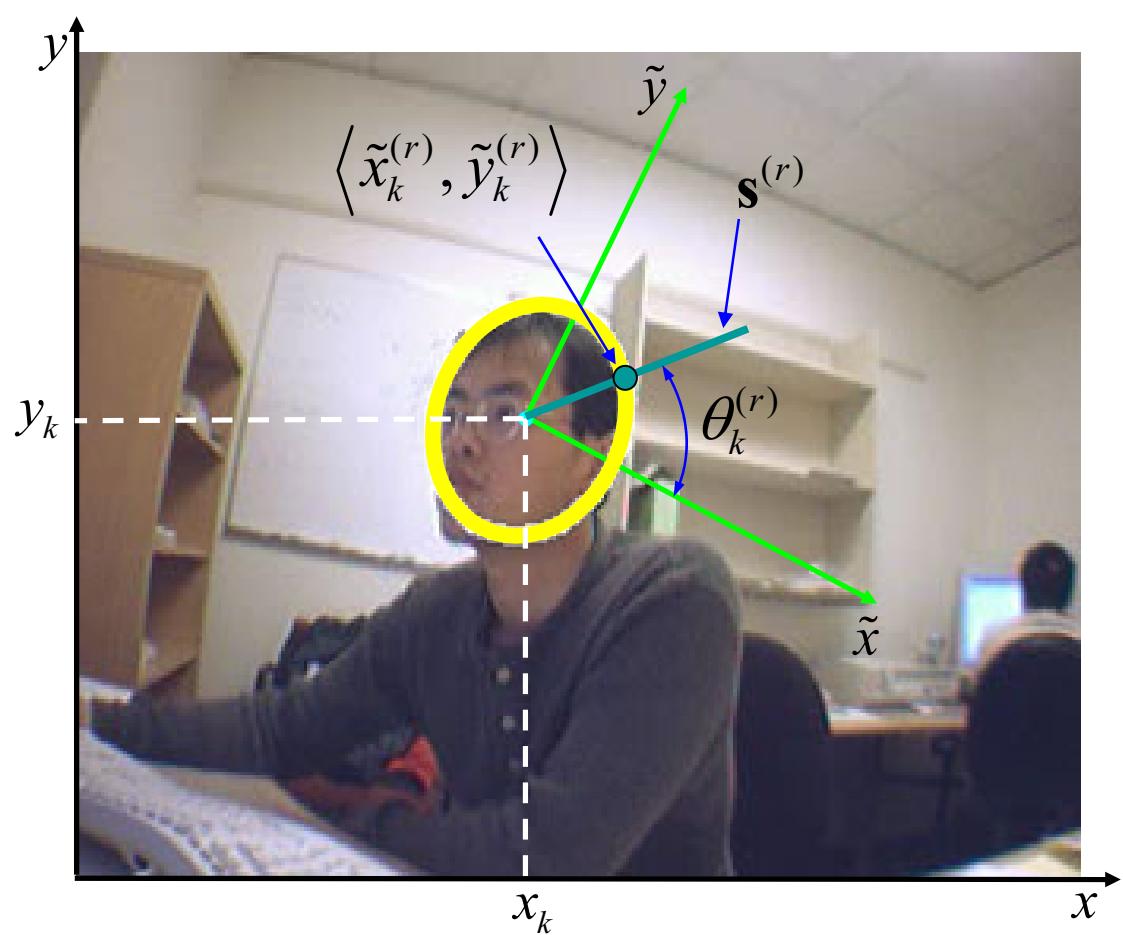


Figure 6.16: Human face tracking with the SPPF

For this reason, it is imperative for particle filter solutions to distribute the limited number of particles in an effective way to areas of increased likelihood as determined by the video stream. Clearly, this application stands to benefit significantly from the use of a SPPF that not only incorporates the latest observation into the proposal distribution, but is also capable of accurately dealing with the severe nonlinearities of the observation model through the implicit use of the SPKF.

In order to implement any form of particle filter (PF, SPPF, etc.) as a solution to this problem, we need to determine the specific forms of the process and observation models as well as the observation likelihood function (needed to evaluate importance weights). Note that in general, for a simple additive Gaussian observation noise source and single hypothesis noisy observations the observation model and observation likelihood function are uniquely linked through the observation noise density function. For this problem however, even though a specific realization of the system state will result in a unique (if noisy) observation model prediction. The likelihood function on the other hand will not be so simple. Due to visual clutter and related non-Gaussian corrupting disturbances, the likelihood function will typically be multimodal in order to accurately model the underlying multiple-hypothesis nature of the problem. For this reason, we need to specify the observation model and likelihood function separately. They are however related through the underlying hidden state value \mathbf{x}_k . The subtle difference between these two functions are however their causal relationship to the system state: The observation model is treated as a function of the hidden state, predicting the value of the expected observation \mathbf{y}_k . The likelihood function on the other hand treats the specific observation \mathbf{y}_k as fixed and then models the likelihood of the underlying (possible) values of \mathbf{x}_k . We will now show how these are derived for the human face tracking problem.

Process Model: $\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_{k-1})$

Let $(x, y)_k$ represent the image plane coordinates of the center of the head-contour ellipse as illustrated in Figure 6.16. The system states are the 2D position and velocity of the ellipse center, i.e.,

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^T . \quad (6.97)$$

To model the movement dynamics of a talking person, the Langevin process is used [169]. This 2nd order movement model has been motivated by numerous researchers in the field of human behavior tracking [87, 169, 29, 117]. This Langevin model is given by the following set of coupled continuous time differential equation:

$$\frac{d^2x_t}{dt^2} + \beta_x \frac{dx_t}{dt} = v \quad (6.98)$$

$$\frac{d^2y_t}{dt^2} + \beta_y \frac{dy_t}{dt} = v, \quad (6.99)$$

where β_x and β_y are the movement rate constants and v is a Gaussian thermal excitation process drawn from $\mathcal{N}(v; 0, \sigma_v^2)$. The discrete form of the Langevin process applied to the parametric model of the head-contour ellipse is given by [197]:

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & \tau & 0 \\ 0 & 1 & 0 & \tau \\ 0 & 0 & a_x & 0 \\ 0 & 0 & 0 & a_y \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_{k-1} + \begin{bmatrix} 0 \\ 0 \\ b_x \\ b_y \end{bmatrix} v_k, \quad (6.100)$$

where τ is the discretization time step, $a_x = \exp(-\beta_x \tau)$ and $a_y = \exp(-\beta_y \tau)$ are the discrete movement rate constants, $b_x = \bar{v}\sqrt{1 - a_x^2}$ and $b_y = \bar{v}\sqrt{1 - a_y^2}$ are the discrete excitation rate constants and \bar{v} is the steady-state root-mean square velocity. This linear model is used as the process model.

Observation Model: $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k)$

Referring to Figure 6.16 we see that the head-contour ellipse is centered at the current position subvector of the system state, i.e., $\mathbf{x}_k = [x_k \ y_k]^T$. In order to build our observation model we generate K rays, $\{\mathbf{s}^{(r)}; r = 1 \dots K\}$, from the ellipse center and let them intersect with the ellipse boundary. If we use the ellipse center as the origin of a local coordinate system that coincides with the ellipse's minor and major axes \tilde{x} and \tilde{y} , the K

intersections $(\tilde{x}_k^{(r)}, \tilde{y}_k^{(r)})$ at time k is given by

$$\tilde{x}_k^{(r)} = \sqrt{\frac{\left(\tan \theta_k^{(r)}\right)^2}{1.44 \left(\tan \theta_k^{(r)}\right)^2 + 1}} \quad (6.101)$$

$$\tilde{y}_k^{(r)} = \sqrt{\frac{1}{1.44 \left(\tan \theta_k^{(r)}\right)^2 + 1}}, \quad (6.102)$$

which results from jointly solving the ellipse equation and the ray equation for each of the K rays, i.e.,

$$(\tilde{x}_k^{(r)}, \tilde{y}_k^{(r)}) \Leftarrow \begin{cases} \frac{(\tilde{x}_k^{(r)})^2}{1} + \frac{(\tilde{y}_k^{(r)})^2}{(1.2)^2} = 1 \\ \tilde{x}_k^{(r)} = \tilde{y}_k^{(r)} \tan \theta_k^{(r)} \end{cases}, \quad (6.103)$$

where $\theta_k^{(r)}$ is the angle between the r 'th ray and the \tilde{x} axis. Transforming the ellipse local (\tilde{x}, \tilde{y}) coordinates back to the image coordinate frame, we obtain the following highly nonlinear observation model:

$$\begin{aligned} \mathbf{z}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k) \\ \begin{bmatrix} z_x^{(1)} \\ z_y^{(1)} \\ z_x^{(2)} \\ z_y^{(2)} \\ \vdots \\ z_x^{(K)} \\ z_y^{(K)} \end{bmatrix} &= \begin{bmatrix} \tilde{x}_k^{(1)} + x_k \\ \tilde{y}_k^{(1)} + y_k \\ \tilde{x}_k^{(2)} + x_k \\ \tilde{y}_k^{(2)} + y_k \\ \vdots \\ \tilde{x}_k^{(K)} + x_k \\ \tilde{y}_k^{(K)} + y_k \end{bmatrix} + \mathbf{n}_k, \end{aligned} \quad (6.104)$$

where \mathbf{n}_k is the observation noise term, drawn from a zero-mean Gaussian distribution, $\mathcal{N}(\mathbf{n}; 0, \mathbf{R})$.

Likelihood Model: $p(\mathbf{z}_k | \mathbf{x}_k)$

The edge intensity is used to model the likelihood function. Specifically, we use a Canny edge detector [27] to calculate the scalar edge intensity along each of the K ellipse centered

rays. This function is in general multi-peaked due to noise and clutter in the video image. Let J be the number of peaks in the edge-intensity function. Of these J peaks, at most one is from the true edge along the ray. Following a similar approach as used in [87] and [197], we can therefore define $J + 1$ hypotheses:

$$H_0 : \{p_j = F ; j = 1, \dots, J\} \quad (6.105)$$

$$H_j : \{p_j = T, p_k = F ; k = 1, \dots, J; k \neq j\} , \quad (6.106)$$

where $p_j = T$ means the j th peak is associated with the true edge and $p_j = F$ represents its complement, an association with a false peak. Hypothesis H_0 therefore implies that none of the observed peaks along a ray is associated with the true edge and Hypothesis H_j implies at least peak j is associated with the true edge. Using these hypotheses we can now define the combined edge likelihood along ray r as:

$$p^{(r)}(\mathbf{z}_k | \mathbf{x}_k) = \alpha_0^{(r)} p^{(r)}(\mathbf{z}_k | H_0) + \sum_{j=1}^J \alpha_j^{(r)} p^{(r)}(\mathbf{z}_k | H_j) , \quad (6.107)$$

where $\alpha_0^{(r)}$ and $\alpha_j^{(r)}$ are the prior probabilities of H_0 and H_j respectively (on ray r), such that $\alpha_0^{(r)} + \sum_{j=1}^J \alpha_j^{(r)} = 1$. The choice of probability distribution used to model Hypothesis H_0 generally dependent on certain assumptions of the background image clutter. For this experiment a uniform distribution was used, i.e.,

$$p^{(r)}(\mathbf{z}_k | H_0) = U . \quad (6.108)$$

Furthermore, a single Gaussian distribution centered at the location (along the ray) of peak j is used to model $p^{(r)}(\mathbf{z}_k | H_j)$. The variance of this Gaussian is typically set according to off-line data analysis done on large numbers of representative images. Finally, the overall likelihood considering all of the K rays is given by:

$$p(\mathbf{z}_k | \mathbf{x}_k) = \prod_{r=1}^K p^{(r)}(\mathbf{z}_k | \mathbf{x}_k) . \quad (6.109)$$

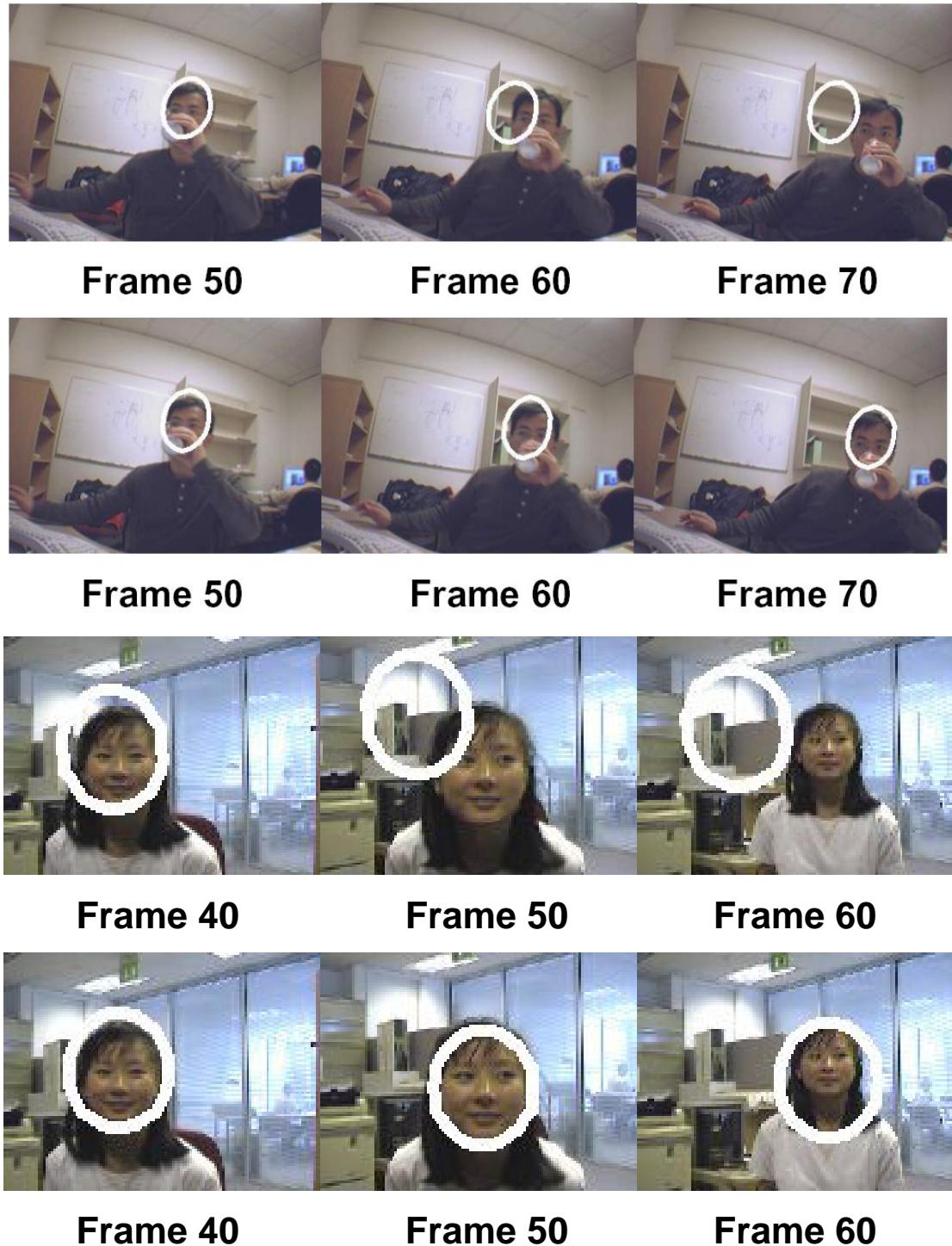


Figure 6.17: SPPF based human face tracking: The top plot of each sequence shows the tracking results of a standard (CONDENSATION [87]) particle filter and the bottom plot of each sequence shows the superior tracking performance of the SPPF.

Experimental Tracking Results

Using the models and functions as defined above and the SPPF algorithm as defined in Section 6.3.1, a real-time tracking system was implemented in C++ on a Windows 2000 platform [169]. Our SPPF algorithm was implemented as-is without attempting any further computational optimizations. The system was found to run comfortably at 30 frames/sec, utilizing 30 particles and 5 rays per particle. The video image resolution used is 320x240 pixels. The tracking experiment was conducted in a normal office with bookshelves, computer monitors and other people in the background. Two typical tracking sequences are reported here in Figure 6.17. The tracking results of the industry standard CONDENSATION algorithm [87] is compared against that of the proposed SPPF based system. The CONDENSATION algorithm is a standard SIR-PF using the above defined state, process and observation likelihood models. The top plot of each sequence (plot 1 and 3) shows the tracking result of the CONDENSATION based system and the bottom plots (plot 2 and 4) show the SPPF results. In both sequences, when the person moves to a location that is not readily predicted by the transition prior, the CONDENSATION algorithm is easily distracted by background clutter (e.g. the bookshelf in Sequence 1 and the oval shaped light-spot on the rear wall in Sequence 2). This is directly due to the fact that the transition prior does not take the most current observation into account when proposing new samples. On the other hand, because the SPPF's superior SPKF derived proposal distribution places the limited number of samples (particles) more effectively, it is able to more robustly track the true location of the persons head in both sequences.

In order to truly judge the difference in tracking performance between these two approaches, the reader is urged to download and view the following two real video-sequence captures (courtesy of Yong Rui). These sequences are encoded using the standard MPEG format and can be downloaded at the following URLs:

- http://wozar.com/phd/cond_tracking.mpg (CONDENSATION)
- http://wozar.com/phd/sppf_tracking.mpg (SPPF)

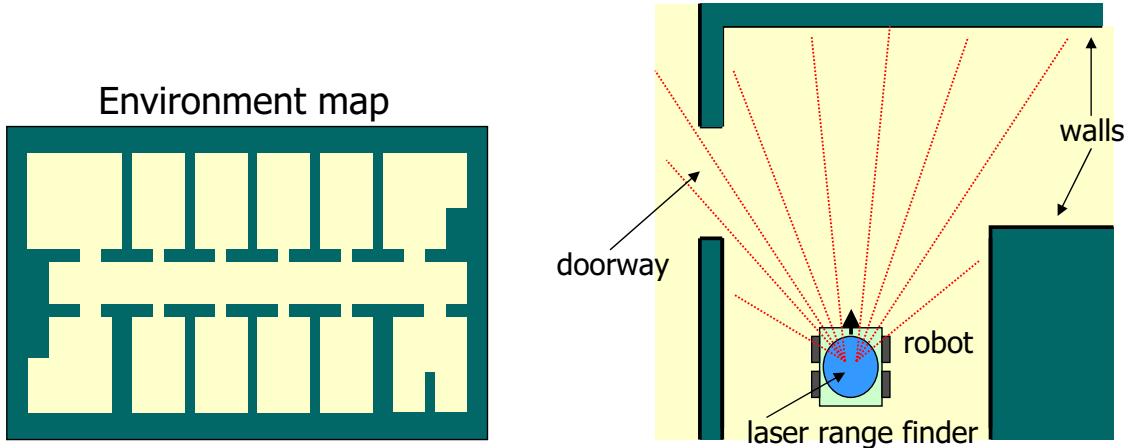


Figure 6.18: Mobile robot localization: (left) Global map of robot's environment. (right) Physical setup of robot within its environment.

6.4.5 Experiment 5 : Mobile Robot Localization¹²

Mobile robot localization (MRL) is the problem of estimating a robot's *pose* (2D position and heading) relative to a map of its environment. This problem has been recognized as one of the fundamental problems in mobile robotics [32]. The mobile robot localization problem comes in two flavors. The simpler of these is *position tracking* where the robot's *initial pose* is known, and localization seeks to correct small, incremental errors in the robots odometry. More challenging is the *global localization* problem, also known as the *hijacked robot problem*, where the robot does not know its initial pose, but instead has to determine it from scratch based only on noisy sensor measurements and a map of its environment. Note, the robot does not initially know *where* on the map it is.

In this inference problem there are often ambiguous or equally viable solutions active at any given moment (due to symmetries in the map and sensor data) which rules out simple single Gaussian representations of the posterior state distribution. Particle filter solutions

¹²This work was done in collaboration with Dieter Fox and Simon Julier who provided the environment maps as well as the observation likelihood model for the laser-based range finder sensors. The preliminary results of this collaboration is reported here, but will be presented more thoroughly in a future publication. See Dieter Fox's publications at <http://www.cs.washington.edu/homes/fox/> for a wealth of information on mobile robot localization.

are the algorithm of choice to tackle this problem [55] due to their ability to present the non-Gaussian multi-modal posterior state distribution related to the multiple *hypotheses* for the underlying robot pose.

One counter intuitive problem often arising in mobile robot localization using standard particle filters is that very accurate sensors often results in worse estimation (localization) performance than using inaccurate sensors. This is due to the fact that accurate sensors have very peaked likelihood functions, which in turn results in severe particle depletion during the SIR step of the sequential Monte Carlo measurement update. When the effective size of the particle set becomes too small due to particle depletion, the particle filter can no longer accurately represent the true shape of the posterior state distribution resulting in inaccurate estimates, or even filter divergence when the correct hypothesis are no longer maintained (the correct mode in the posterior disappears).

In order to address this problem one typically need to use a very large number ($\geq 20,000$) of particles, resulting in a high computational cost. The irony is that the number of particles needed later on in the localization process, when much of the uncertainty or ambiguity of the robot's pose has been reduced, is much less than what is initially needed. So, a large number of the particles eventually becomes superfluous but still incurs a computational burden. Recent work on *adaptive particle filters* using KLD sampling [53] attempts to address this by adapting the number of particles used in real time. We present here some preliminary results in using the GMSPPF to address the same problem. Take note, that due to the large number of particles needed to accurately represent the posterior (at least initially), the use of the SPPF, although highly accurate, would incur a prohibitively high computational cost.

The GMSPPF has two appealing features which make it an attractive solution to the MRL problem. As discussed in Sections 6.3, 6.3.1 and 6.3.2, the GMSPPF (and SPPF) moves particles to areas of high likelihood by using a proposal distribution that incorporates the latest measurement. This addresses the particle depletion problem. Furthermore,

by using an adaptive clustering EM algorithm such as *X-means*¹³ initialized *FastEM*¹⁴ [55, 137] in the GMM recovery step, the model order (number of Gaussian components) is automatically adapted over time to more efficiently model the true nature of the posterior. We typically find that the model order which is initially set to a large number (to model the almost uniform uncertainty over the state space) slowly decreases over time as the posterior becomes more peaked in only a few ambiguous areas. The advantage of this is that the computational cost also reduces over time in relation to the model order, since we use a fixed number of particles per Gaussian component in the posterior GMM.

Before we report the experimental results, we will first briefly describe the process (movement) model and observation models that were used for this problem:

Process Model: $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k)$

We make use of a rather simple, but highly nonlinear process model for the robot's movement. Firstly, the robot's state is defined as its 2D position (x, y) relative to the map of its environment, as well as its heading angle (ψ) :

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \psi_k \end{bmatrix}^T. \quad (6.110)$$

The robot's movement from one position on the map to another, within one discrete time step, is broken down into two sub-maneuvers: First the robot rotates towards the desired heading, and then it translates along that bearing at the required velocity needed to reach the desired end coordinates. In other words, the robot's exogenous command input (\mathbf{u}_k)

¹³ *X-means* is a highly efficient C-language implementation of the *k-means algorithm* for automatic clustering of multivariate data and was developed by Andrew Moore and Dan Pelleg at CMU [153]. *X-means* makes use of multi-resolution *kd-tree* data structure embedding of the data to reduce the large number of nearest-neighbor queries issued by the traditional k-means algorithm. The *kd-tree* is incrementally built and sufficient statistics are stored in the nodes of the tree. Then, an analysis of the geometry of the current cluster centers results in a great reduction of the work needed to update the centers. The *kd-tree* data structure further allows for efficient initialization of the k-means starting centroids.

¹⁴ *FastEM* [137] is a computationally efficient C-language implementation of the *expectation-maximization* (EM) [41] algorithm. Like the *X-means* algorithm, this computational efficiency is achieved through the use of *kd-tree* data structures for data embedding. The *FastEM* algorithm also allows for on-line adaptive model-selection, using either the AIC (Akaike Information Criteria) or BIC (Bayesian Information Criteria) goodness-of-fit cost functions.

comprises of the initial rotation angle and the desired translation distance, i.e.,

$$\mathbf{u}_k = \begin{bmatrix} \theta & d \end{bmatrix}^T . \quad (6.111)$$

Since we assume the desired distance can be covered within a single discrete time-step, the distance command is analogous to a constant velocity command over that time-step, and must be limited according to the physical constraints of the actuators and motors for the specific robot model in question.

The process noise affecting the accuracy of any specified desired movement has three components: First, due to stepper motor resolution limitations and quantization noise, there will always be a finite discrepancy between the commanded and actually realized rotation angle and translation distance. We model these as additive Gaussian noise terms. The second cause of movement inaccuracy stems from the small but finite differences between the response of the left hand and right hand driving wheels/tracks of any mobile robot (which are driven separately), as well as the surface over which it moves. These differences all contribute to the inability of the robot to follow a perfectly straight trajectory. The magnitude of this effect is in general proportional to the magnitudes of the commanded rotation angle and translation distance. In order to model this noise process, we make the magnitude of the first two additive noise components (mentioned above) proportional to the absolute magnitude of the commanded rotation angle and translation distance. Furthermore, we add a final post-translation random rotation error to the achieved heading angle. The magnitude of this Gaussian noise term, which forms the third component of the process noise vector, is proportional to the magnitude of the achieved translation. The process noise vector (\mathbf{v}_k) is thus given by,

$$\mathbf{v}_k = \begin{bmatrix} v_\theta & v_d & v_\psi \end{bmatrix}_k^T , \quad (6.112)$$

where $v_{\theta_k} \sim \mathcal{N}(v_\theta; 0, \sigma_\theta^2)$ is the pre-translation rotational error, $v_{d_k} \sim \mathcal{N}(v_d; 0, \sigma_d^2)$ is the translation error, and $v_{\psi_k} \sim \mathcal{N}(v_\psi; 0, \sigma_\psi^2)$ is the post-translation rotational error.

Finally, the full process model is given by:

$$\tilde{v}_{d_k} = \|d_k\| \cdot v_{d_k} \quad (6.113)$$

$$\tilde{v}_{\theta_k} = \|\theta_k\| \cdot v_{\theta_k} \quad (6.114)$$

$$\tilde{v}_{\psi_k} = \|d_k + \tilde{v}_{d_k}\| \cdot v_{\psi_k} \quad (6.115)$$

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k) \\ \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \psi_{k+1} \end{bmatrix} &= \begin{bmatrix} x_k + (d_k + \tilde{v}_{d_k}) \cos(\psi_k + \theta_k + \tilde{v}_{\theta_k}) \\ y_k + (d_k + \tilde{v}_{d_k}) \sin(\psi_k + \theta_k + \tilde{v}_{\theta_k}) \\ \psi_k + \theta_k + \tilde{v}_{\theta_k} + \tilde{v}_{\psi_k} \end{bmatrix}, \end{aligned} \quad (6.116)$$

where $\|\cdot\|$ is the standard norm operator. All of the angular addition operations in Equation 6.116 are performed *modulus*- 2π , such that all resulting angles fall within the $-\pi \leq \psi \leq \pi$ range.

Using the environment map (and computational ray tracing), the process model also determines if a specific movement trajectory will result in a collision with any of the walls. If such a collision occurs, the robot will stop and perform a stationary rotation such that it is pointing away from the wall (based on incidence angle) at the start of the next time step.

Observation Model: $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k)$

As schematically illustrated in Figure 6.18, the only observation sensor the robot uses to sense its environment is a high accuracy laser (or sonar) range finder. This sensor measures the line-of-sight distance to the closest walls along K rays in a 0.5π radian horizontal fan centered around the robots current heading. In order to model such a sensor we make use of a computational ray-tracing model based on the environment map (walls and free-space) defined on a finite resolution grid¹⁵. The observation vector is thus a K -dimensional vector indicating the distance to the closest obstacle along each of the K sensor beams. Each

¹⁵The computational cost of such a model grows quadratically with respect to the grid cell size. For this reason one has to reach a practical compromise between the computational overhead of the observation model and the grid resolution needed for accurate location and tracking performance. For our experiment we used environmental maps with a 10x10 cm grid cell size.

measurement is corrupted by an additive independent Gaussian random variable, i.e.,

$$\Delta\psi_k = \frac{0.5\pi}{K-1} \quad (6.117)$$

$$\begin{aligned} \mathbf{z}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k) \\ \begin{bmatrix} z_k^{(1)} \\ z_k^{(2)} \\ \vdots \\ z_k^{(K)} \end{bmatrix} &= \begin{bmatrix} r(\psi_k - 0.25\pi + 0 \cdot \Delta\psi_k, x_k, y_k) \\ r(\psi_k - 0.25\pi + 1 \cdot \Delta\psi_k, x_k, y_k) \\ \vdots \\ r(\psi_k - 0.25\pi + (K-1) \cdot \Delta\psi_k, x_k, y_k) \end{bmatrix} + \begin{bmatrix} n_k^{(1)} \\ n_k^{(2)} \\ \vdots \\ n_k^{(K)} \end{bmatrix}, \end{aligned} \quad (6.118)$$

where $r(\phi, x, y)$ is the ray-tracing distance function defined as:

$$r(\phi, x, y) \doteq \{\text{distance to obstacle along heading } \phi, \text{ from position } (x, y)\}, \quad (6.119)$$

and \mathbf{n}_k is the K -dimensional observation noise vector drawn from a zero-mean Gaussian distribution, $\mathbf{n}_k \sim \mathcal{N}(\mathbf{n}; \mathbf{0}, \mathbf{R})$. Due to the typically high accuracy of laser range-finder sensors, the measurement noise variance \mathbf{R} is set to a small value. This will result in very peaked likelihood functions, which in turn will result in severe sample depletion for standard particle filters. In order to mitigate this problem (as discussed before), we either have to use huge amounts of particles¹⁶ and/or use better proposal distributions that move the particles to areas of increased likelihood.

Experimental Results: SIR-PF vs. GMSPPF

Using the process and observation models defined above, we implemented a SIR-PF and GMSPPF based mobile robot localization system using the *ReBEL Toolkit* (See Appendix C for detail about the toolkit) and tested it (in simulation) on a benchmark MRL problem provided by Dieter Fox.

Figure 6.19 gives a graphical comparison of the localization results using a SIR particle filter (SIR-PF) on the left and the GMSPPF on the right. The same plots are reproduced

¹⁶As mentioned earlier, the number of particles can possibly be adapted over time using the KLD version of the particle filter [53]. This will not reduce the initially large computational cost though while there is still large amount of uncertainty in the posterior. The use of the KLD approach to possibly extend the SPPF (and possibly the GMSPPF) algorithms is an area of ongoing research.

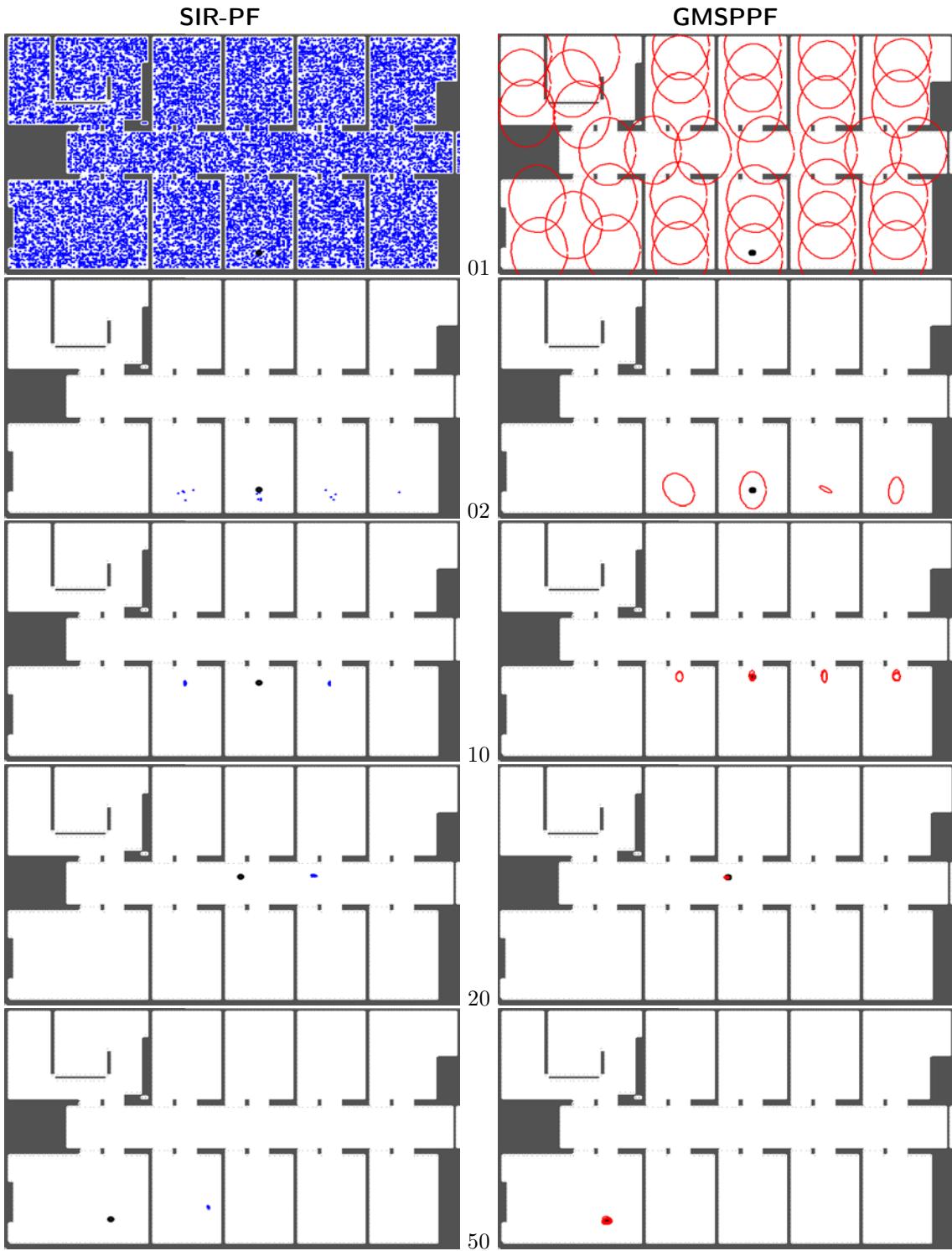


Figure 6.19: Mobile robot localization: Results using a SIR particle filter (left) and GMSPPF (right). The true robot position (black dot) and filter generated posterior is shown for $k = \{1, 2, 10, 20, 50\}$ (from top to bottom) as the robot moves from one room up into the hallway, turn left, move down the hallway and down again into the large room. SIR-PF particles are shown in blue (on the left) and the GMSPPF's GMM posterior's Gaussian component densities are shown in red on the right.

at a higher resolution in Figures 6.20-6.24.

The SIR-PF uses 20,000 particles (shown in blue) to represent the posterior and uses the standard transition prior (movement model of the robot) as proposal distribution. Residual resampling is used. The initial particle set was uniformly distributed in the free-space of the map (See Figure 6.20).

The GMSPPF uses 500 samples per Gaussian component, the number of which are determined automatically using an *x-means initialized EM clustering stage* [137, 153]. The initial number of Gaussian components (shown in red) in the posterior was set to 42, uniformly distributed across the free-space of the map (See Figure 6.20).

The SIR-PF quickly suffers from severe particle depletion even though 20,000 particles are used to represent the posterior. At $k = 10$ (third plot from the top on the left of Figure 6.19, or Figure 6.22 for a close-up) the SIR-PF has already lost track of the true position of the robot. The GMSPPF in contrast accurately represent all possible robot location hypotheses through multiple Gaussian components in its GMM posterior. As the ambiguity and uncertainty of the robots location is reduced, the number of modes in the posterior decreases, as well as the number of Gaussian component densities needed to model it. The GMSPPF accurately tracks the true position of the robot for the whole movement trajectory. The superior performance of the GMSPPF over that of the SIR-PF is clearly evident.

As was the case for Experiment 4, we strongly suggest that the reader download and view two short video sequences of the localization performance of these two filters. These sequences are encoded using the standard MPEG format and can be downloaded at the following URLs:

- http://wozar.com/phd/sirpf_mrl.mpg (SIR-PF)
- http://wozar.com/phd/gmsppf_mrl.mpg (GMSPPF)

We determined through extensive experimentation that the GMSPPF algorithm's adaptive clustering GMM recovery step is very important for robust filter operation. If the reclustering is done naively or if it is initialized badly, the GMSPPF might fail to recover a good GMM representation of the posterior. These include situations such as using too many

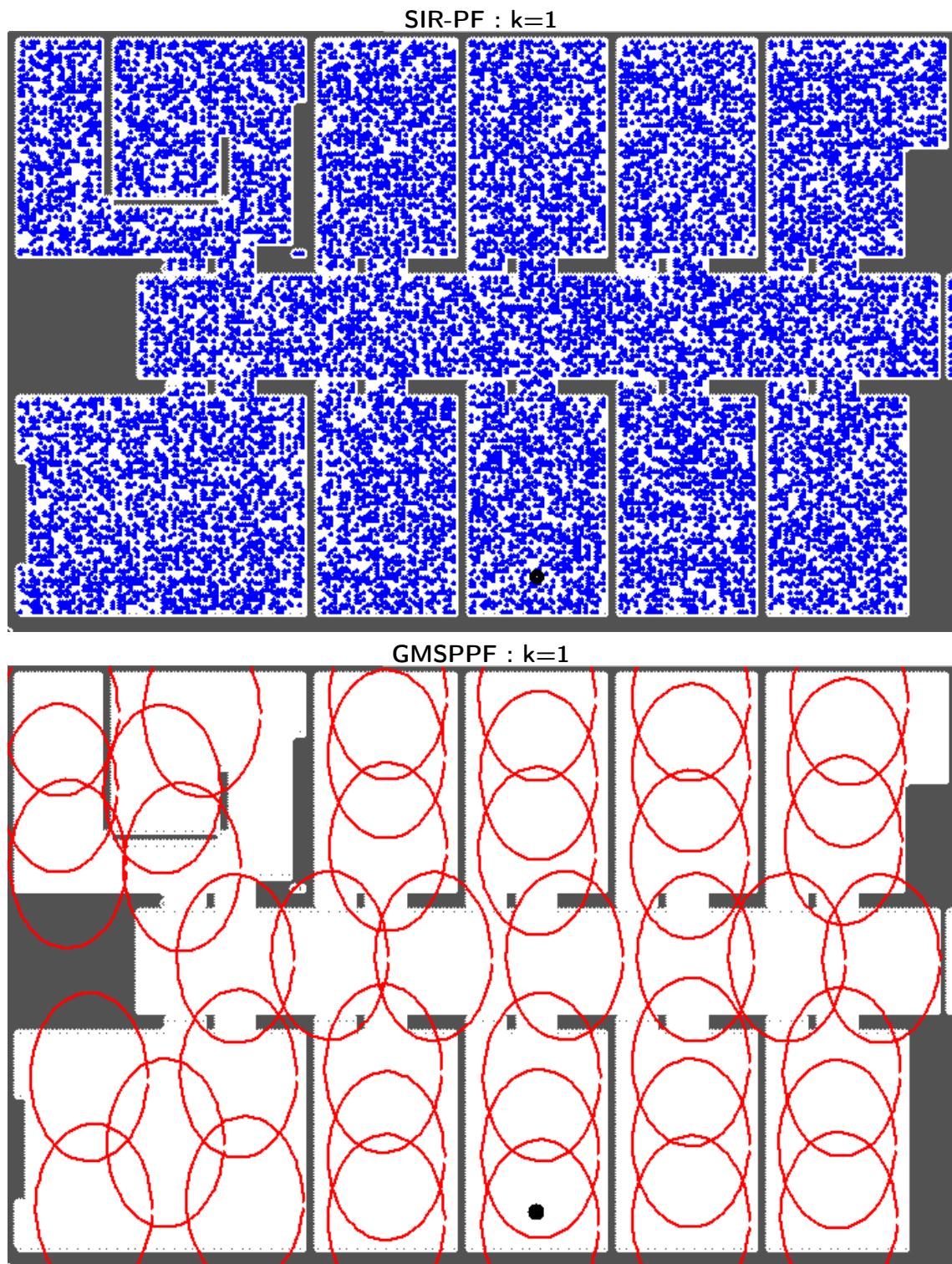


Figure 6.20: Mobile robot localization: This plot shows the initial distribution of particles and GMM component densities at time $k=1$. Notice the almost uniform coverage of the map's free space. The robot's true location is indicated by the large black dot.

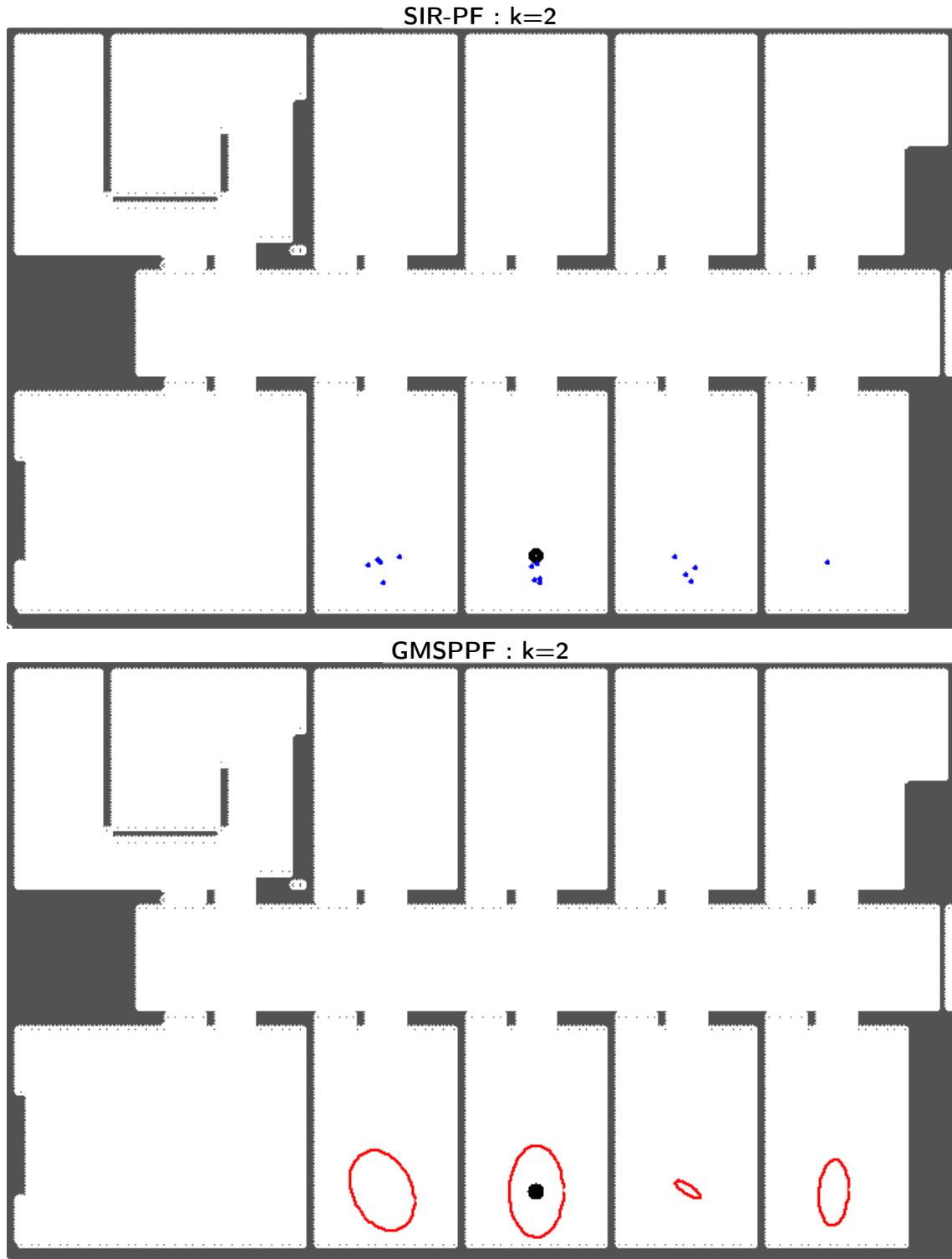


Figure 6.21: Mobile robot localization: Almost immediately ($k=2$) severe sample depletion is observed in the SIR-PF, due to the highly peaked observation likelihood and failure to incorporate the latest observations into the proposal distribution. The GMSPPF on the other hand accurately approximates the multiple-hypothesis posterior with a GMM component in each of the four possible room locations. The SIR-PF only has a few samples in each room which results in a highly degraded posterior approximation.

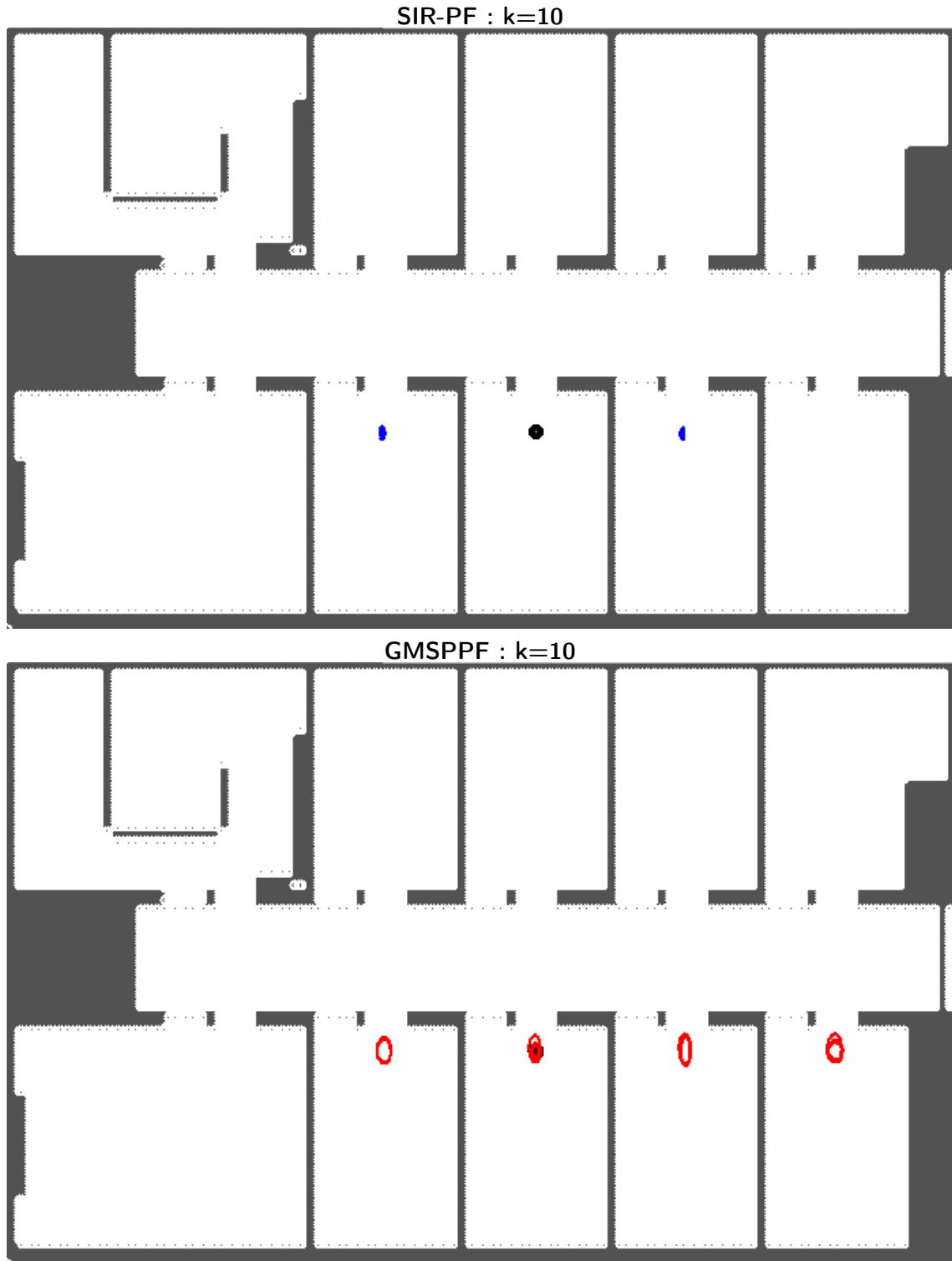


Figure 6.22: Mobile robot localization: The SIR-PF's posterior has collapsed to two highly peaked modes, neither of which coincides with the true location of the robot. At this point, the SIR-PF is no longer capable of accurately locating or tracking the robot. The GMSPPF, on the other hand, is still accurately tracking the four possible locations of the robot.

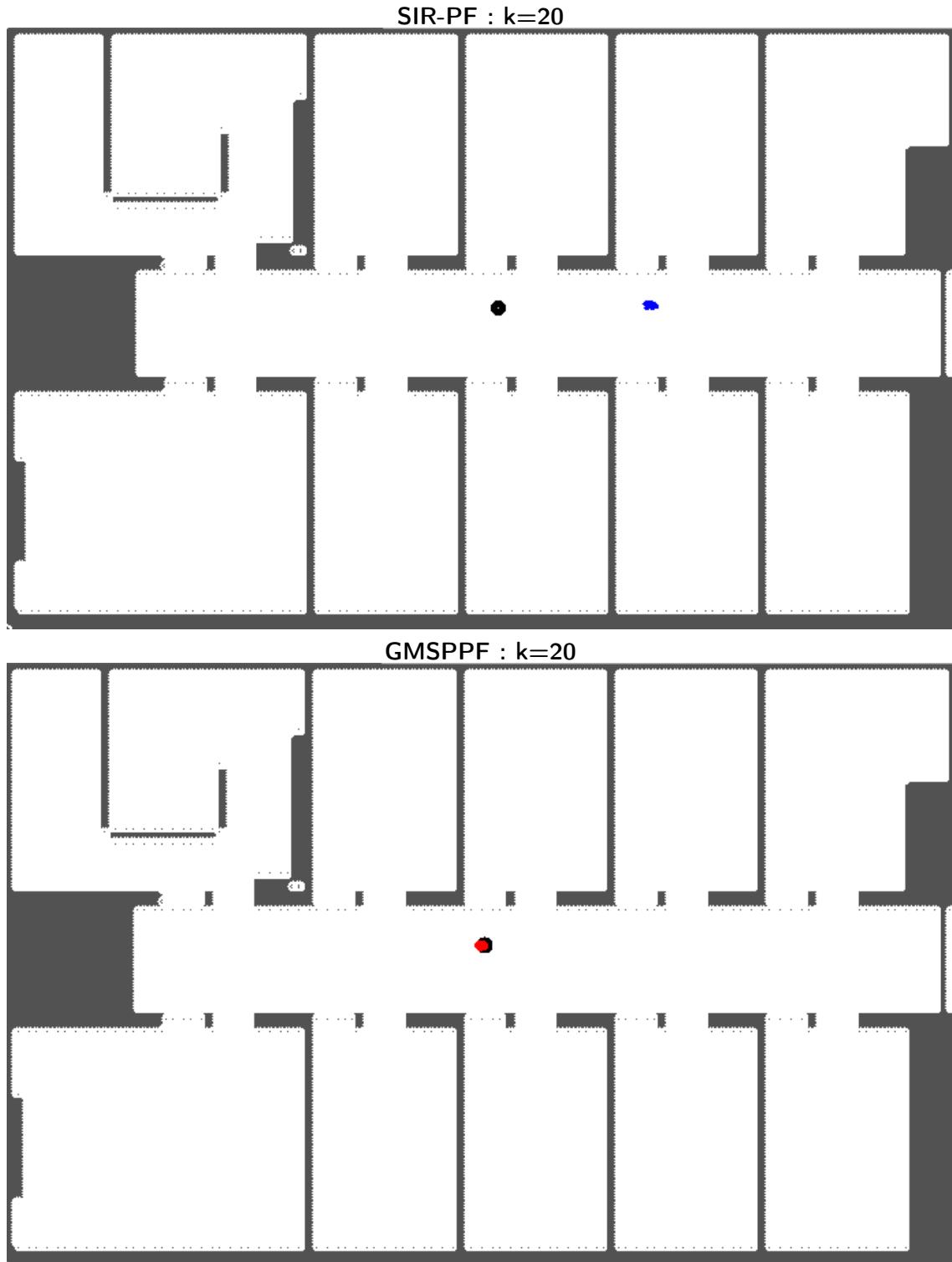


Figure 6.23: Mobile robot localization: As soon as the robot entered the hallway and turned left, the GMSPPF was able to completely disambiguate the the posterior and lock-in on the true location of the robot. The GMM posterior has just reduced to a single mode (modeled by a reduced number of Gaussian components). The SIR-PF's posterior has also reduced to a single mode, but unfortunately, not the correct one.

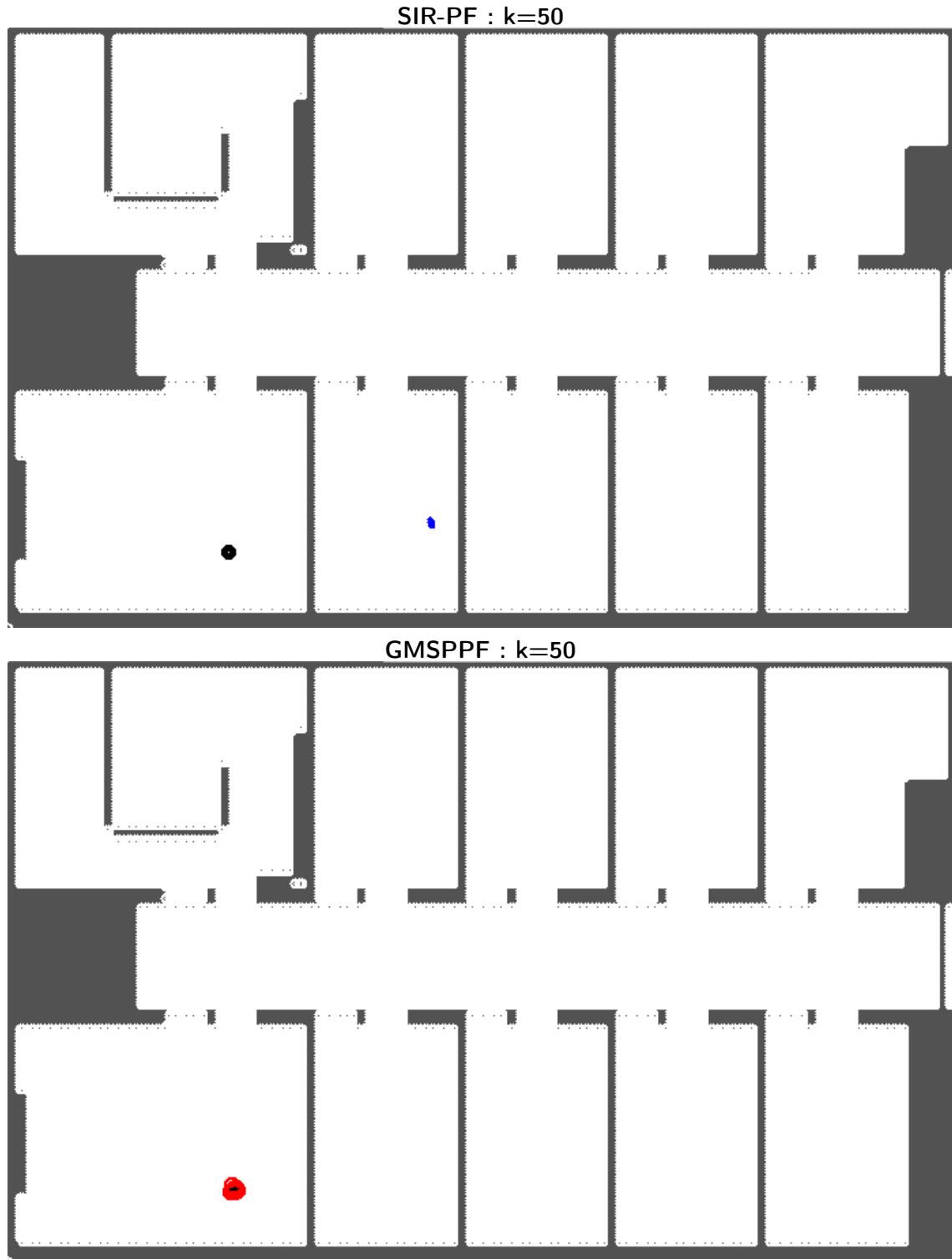


Figure 6.24: Mobile robot localization: The GMSPPF accurately tracks the robot to the end of its current trajectory (the large room at the bottom left). When the robot entered this room, the SIR-PF's posterior completely collapsed (no particles left), due to the paradox/inconsistency between what its posterior predicted and what the sensors reported.

Gaussian components in the GMM where fewer might suffice, resulting in wasted computational resources, or incorrectly grouping closely located by physically separate clusters of particles together in a single GMM component density. For the MRL problem, the latter situation can occur when particles situated on different sides of a wall or a hallway corner are clustered into a single Gaussian component. In order to robustly address this problem one can make use of more advanced *constrained* clustering algorithms [198, 103], which, for example, makes use of convexity constraints and other meta-information derived from the environment map, to constrain the clustering process. This is an ongoing research question which is currently under investigation by the author. Nonetheless, the preliminary experimental results reported here on the use of the GMSPPF algorithm for mobile robot localization, is quite encouraging and does serve as a successful proof-of-concept.

6.5 Chapter Summary

In the earlier chapters of this thesis we introduced the SPKF as an improved Gaussian approximate solution to the recursive Bayesian estimation problem. Many real-world inference problems can be adequately solved using Gaussian approximations and for those the SPKF is clearly the method of choice as repeatedly demonstrated in Chapters 3 and 5. This fact notwithstanding, there remain a large number of systems for which simple Gaussian approximations simply will not suffice. For these nonlinear, non-Gaussians problems, we must make use of more complex inference algorithms that allow for the complete and salient description of the true shape of the posterior density function.

In this chapter we introduced the class of approximate recursive Bayesian solutions, based on sequential importance sampling, called *sequential Monte Carlo methods* or *particle filters*. These algorithms have, over the last couple of years, become the method of choice to deal with real-world nonlinear, non-Gaussian inference problems. The derivation of the particle filter was presented in Section 6.2. Along with this exposition we also presented (in Section 6.2.4) the major performance decreasing ailment of general particle filters, called the sample degeneracy/depletion problem. Even though the resampling stage of the particle filter addresses this problem, it does not completely mitigate it. In

Section 6.3 we showed that one of the main causes for this problem is the fact that the proposal distribution used by the standard particle filter does not incorporate the latest observation. This prevents the particles from moving to areas of high likelihood during the proposal/sampling stage. We proposed to address this problem by designing better proposal distributions that do in fact incorporate the latest observation and hence better approximate the theoretical optimal proposal distribution. This was done through the hybrid combination of the SPKF with the sequential Monte Carlo inference framework of a particle filter. The results of this was two new hybrid algorithms called the *sigma-point particle filter (SPPF)* and the *Gaussian mixture sigma-point particle filter (GMSPPF)*. In Sections 6.3.1 and 6.3.2 respectively we showed how these filters are motivated, derived and implemented.

Finally in Section 6.4 we demonstrated the utility and improved performance capabilities of the proposed new algorithms on a variety of nonlinear, non-Gaussian inference problems.

Chapter 7

Conclusions and Future Work

7.1 Introduction

This chapter provides a concise, retrospective synopsis of the work presented in this dissertation and emphasizes the notable results. From these results conclusions are drawn and directions for possible future research are proposed. The work presented in this thesis and published in part in [199, 200, 204, 77, 189, 190, 191, 192, 193, 188], have already impacted a number of external related research efforts, resulting in further applications, refinements and derived works. These works are based in part on or derived from the research presented in this thesis. In this chapter, we will also briefly summarize some of the more important and significant of these derived works.

7.2 Concluding Summary

To design learning machines that infer information from, reason about, and act on the real world, we need to represent uncertainty. Probability theory provides a language for representing these uncertain beliefs and a calculus for manipulating these beliefs in a consistent manner. Utilizing probability theory and the general descriptive power of dynamic state space models, recursive Bayesian estimation provides a theoretically well founded and mathematically robust framework to facilitate sequential probabilistic inference in systems where reasoning under uncertainty is essential. However, as pointed out in Chapter 1, the optimal recursive Bayesian solution to the probabilistic inference problem is in general intractable for most real world problems. This necessitates the need for practically implementable approximate solutions. This thesis has focused on two of the main groups of

such approximate solutions: *Gaussian approximate solutions* and *sequential Monte Carlo methods*.

Within the family of Gaussian approximate solutions, the extended Kalman filter (EKF) has become the *de facto* algorithm of choice for probabilistic inference, used in numerous diverse nonlinear estimation algorithms and related applications. One of the reasons for its wide spread use has been the clear understanding of the underlying theory of the EKF and how that relates to different aspects of the inference problem. This insight has been enhanced by unifying studies towards the relationship between the EKF and other related algorithms [170], allowing for the improvement of numerous existing algorithms and the development of new ones.

Recently, the unscented Kalman filter (UKF) and the central difference Kalman filter (CDKF) have been introduced as viable and more accurate alternatives to the EKF within the framework of state estimation. Like most new algorithms, these methods were not widely known or understood and their application has been limited. We have attempted to unify these differently motivated and derived algorithms under a common family called sigma-point Kalman filters, and extended their use to other areas of probabilistic inference, such as parameter and dual estimation as well as sequential Monte Carlo methods. In doing so we have also extended the theoretical understanding of SPKF based techniques and developed new novel algorithmic structures based on the SPKF. These algorithms were successfully applied to a large group of representative inference and machine learning problems from a variety of related fields, including a major real-world application, namely UAV Autonomy.

The SPKF and its derivatives are hopefully set to become an invaluable standard tool within the “probabilistic inference & machine learning toolkit”. To this end we have released a Matlab toolkit, called *ReBEL*, that not only contains all of the algorithms presented in this dissertation, but also provides a powerful, easily extendable, general algorithmic framework for sequential probabilistic inference and machine learning in dynamic state-space models.

7.3 General Overview

The general probabilistic inference problem within the context of dynamic state-space models was introduced in Chapter 1. We presented the recursive Bayesian estimation algorithm that maintains the posterior density of system state as noisy observations become available, as the optimal solution to the probabilistic inference problem. The optimal Bayesian solution provides a mathematically rigorous and conceptually intuitive framework to derive inference and learning systems that can “reason under uncertainty”. We continue to show how the true optimal Bayesian solution are generally intractable for most real world systems, necessitating the use of approximate solutions. The main subgroups of approximate solutions were introduced with specific focus on the two areas that this thesis aimed to contribute to: *Gaussian approximate solutions* and *sequential Monte Carlo methods*.

Chapter 2 provided an in-depth discussion of optimal Gaussian approximate recursive Bayesian estimation, introduced the extended Kalman filter and showed why the EKF is in fact a highly suboptimal (flawed) approximate solution. This chapter covered much of the introductory motivation of why a better solution than the EKF is needed for Gaussian approximate nonlinear estimation.

Chapter 3 covered the algorithmic development of the *sigma-point Kalman filter* as an improved, theoretically better motivated alternative to the EKF. The two main SPKF variants, the UKF and CDKF, were introduced and shown to be different implementational variations of a common derivativeless Kalman filter framework, which in turn is based on a deterministic sampling technique called the sigma-point approach. We derived numerically efficient and stable square-root versions of the SPKF as well as inference type specific forms. After the different SPKF algorithms were introduced, we covered in detail (using numerous experimental examples) how they are applied to the three domains of probabilistic inference, specifically *state-*, *parameter-* and *dual-estimation*. These experiments also verified the superiority of the SPKF over the EKF for all classes of estimation problems.

In Chapter 4 a number of theoretical aspects of the sigma-point Kalman filter were derived and analyzed in order to gain further insight and relate it to other algorithms.

Specific attention was given to the *weighted statistical linear regression* interpretation of the sigma-point approach as employed by all SPKF forms. This interpretation allowed us to later (in the same chapter) relate the parameter estimation form of the SPKF with other 2nd order optimization methods, specifically online modified Newton methods. We also investigated the theoretical accuracy of the sigma-point approach in comparison to the accuracy of the optimal solution. We showed how all SPKF forms achieve *at least* 2nd order accuracy in the calculation of both the posterior mean and covariance of the Gaussian approximate optimal Kalman update. As already mentioned, the chapter concluded with an in-depth analysis of how SPKF based parameter estimation relates to other 2nd order nonlinear optimization methods. At the end of this chapter the most salient (and important) characteristics of the SPKF were summarized and contrasted with those of the EKF.

In Chapter 5 we focused specifically on the application of the SPKF to the *UAV Autonomy* problem. This large-scale application was covered in depth, giving detail about the vehicle itself (dynamic models, etc.), the experimental setup that was used (high-fidelity simulator) and the numerous experiments that was done. We showed how a novel SPKF centric GPS latency compensation technique can be implemented within the SPKF state estimation framework. This allowed for the implementation of a highly accurate, robust, stand-alone, vehicle-agnostic integrated navigation system which provides navigational state estimates to the UAV’s control system. Experimental results were presented for a number of state estimation (open and closed loop) experiments as well as parameter and dual estimation experiments. These experiments were performed using a high-fidelity nonlinear vehicle simulator that modeled most of the highly nonlinear dynamics of the complete UAV flight system. We compared these results with those of a state-of-the-art, hand-tuned EKF solution and found our SPKF based system to be superior in all aspects of the navigation and state estimation problem. Although all of the experimental results clearly showed the benefit of using our proposed SPKF system over the industry standard EKF based solutions, we still need to verify these results on real (as opposed to simulated) flight-data. This issue is addressed in more detail in Section 7.4.

Chapter 6 focused on the second group of approximate recursive Bayesian estimation

solutions which this thesis addresses, namely *sequential Monte Carlo (SMC) methods*. Unlike Gaussian approximate solutions such as the Kalman filter framework, SMC methods make no assumptions about the form of the posterior state density. They are thus ideally suited to most general, nonlinear, non-Gaussian estimation problems where even the SPKF fails to generate acceptable results. We first derived the general SMC framework, called particle filters, as a recursive implementation of importance sampling combined with a resampling stage. We showed how generic particle filters often suffer from a debilitating ailment called *particle depletion* which is primarily caused by the use of badly designed proposal distributions during the importance sampling stage. Based on this intuition, we showed how the inherent accuracy and robustness of the SPKF can be leveraged to design better proposal distributions that approximate the optimal proposal by incorporating all of the available observation data. This allowed us to derive two new hybrid SMC/SPKF algorithms: the *sigma-point particle filter* and the *Gaussian-mixture sigma-point particle filter*. As with the chapter on the SPKF, we verified these new algorithms on numerous representative inference problems including nonlinear non-Gaussian time-series prediction, financial options pricing, human face tracking and global robot localization.

For a summary of the original contributions made by the work presented in this dissertation, please see Section 1.7.

7.4 Possible Extensions & Future Work

In pursuing the numerous research objectives posed at the beginning of the dissertation, a number of related issues were identified. Although not central to this thesis, these open research questions provide fertile ground for future research aimed at building on or extending the work presented here. A brief summary of some of the more interesting of these suggestions will now be presented:

- **Batch forms of SPKF algorithms:** In their current form, the SPKF based algorithms are online or sequential algorithms. These methods can possibly be extended through DSSM reformulation to batch forms that operate on a whole sequence of data at once. This extension is needed for certain types of applications such as batch

form parameter estimation and perceptual metric based speech enhancement. The development of batch form algorithms are also related to the next issue: arbitrary cost function optimization.

- **Arbitrary metric / cost function optimization:** As reported in Section 3.5.2, the SPKF can be adapted to minimize a non-MSE cost functions as long as the cost function is analytical and expressible as a sum of instantaneous costs. This is not directly applicable to “black box” cost functions such as psychoacoustic perceptual speech metrics used for certain speech coding and speech enhancement applications [162]. These cost functions typically operate in batch mode on sequences of processed data. Furthermore, since they are based on proprietary algorithms, they can only be accessed through a well-defined API, i.e., their inner workings are not made available and must thus be treated as a “black box scoring” system. Since the SPKF only requires functional evaluations and not analytical derivatives, it can easily be used for “black box” DSSMs. If this ease of implementation can be expanded to cope with “black box” cost functions is not yet obvious and requires further research.
- **Online probabilistic noise estimation:** One aspect of the inference problem not directly addressed by this thesis is that of estimating the process and observation noise distributions, $p(\mathbf{v}_k)$ and $p(\mathbf{n}_k)$. We typically assumed that these can be derived from prior knowledge (process noise) or that they are specified by sensor accuracies, etc. In Chapter 3 (Section 3.5.2) we introduced a number of on-line methods which can be used to adapt the covariance of the “synthetic” process noise terms used for SPKF based parameter estimation. Even though these methods performed satisfactory at a negligible computational cost, other techniques are available to incorporate the “noise estimation” problem into the larger probabilistic inference framework at a more fundamental level. This will require the use of a dual estimation framework where not only the system states and parameters, but also the salient parameters of the relevant noise distributions, are estimated. We have already done some preliminary work¹ on this issue with specific focus on model estimation for colored noise

¹The results of these efforts are included in the ReBEL toolkit.

sources. As a possible future direction for this research, the work done by Nelson [143] on noise estimation should be consulted.

- **Nonlinear guaranteed/robust estimation:** Recent work by Scholte and Campbell [173, 174] on guaranteed nonlinear estimation resulted in the publication of an algorithm called the extended set-membership filter (ESMF). This approach makes use of interval mathematics [139, 72] and an assumption that the noise sources are bounded to obtain hard bounds on state and parameter estimates. This allows for the implementation of robust estimation systems that can guarantee some level of maximum error. There seems to be some similarities between the ESMF and the interval Kalman filter developed by Chen et al [28, 179]. Extensions of this body of work on guaranteed/robust estimation to the SPKF framework should be highly beneficial to the larger nonlinear control and estimation field.
- **SPKF with non-Gaussian unimodal posteriors:** Extending the SPKF framework to model the state posterior distribution with heavy tailed unimodal density functions such as the Cauchy or Student-t distributions, has certain attractive benefits for proposal distribution generation inside the SPPF. As mentioned in Chapter 6², the SPPF is theoretically guaranteed to converge if we can upper bound the importance weights. This can be achieved by ensuring that the chosen proposal distribution has heavier tails than the true posterior (up to proportionality). If the SPKF can thus be modified to optimally propagate heavy tailed posteriors for each particle which are then used as proposal distributions inside a particle filter framework, filter convergence will be guaranteed. However, making this modification to the SPKF will not be trivial. The SPKF was derived based on the Gaussian approximate optimal Kalman framework. In other words, only the mean and covariance of the true posterior is propagated and updated using an optimal linear update rule. In order to modify the SPKF to propagate heavy tailed distributions, one will likely have to maintain not only the mean and covariance, but also the fourth-order moments (kurtosis). This will require an increase in the size of the sigma-point set resulting in

²See [189] for a proof.

increased computational complexity. The linear Kalman update itself will possibly have to be adapted as well.

- **Scaling parameters:** One of the confusing aspects of implementing certain types of SPKFs, such as the UKF and its derived forms, is the setting of the scaling parameters (α , β and κ). As discussed in Section 3.2.2, there are certain guidelines how the values of these parameters should be chosen, but no definitive criteria by which to judge optimality of choice. Even though we found the estimation performance of these filters to be rather insensitive to the exact values of these parameters, investigating this matter further has certain merits. The values of these scaling parameters could conceivably be made data dependent and hence be adapted on-line depending on factors such as the severity of nonlinearities, variances of noise sources, etc.
- **Padé approximant based SPKF:** There might be merit in investigating other numerical interpolation techniques such as Padé approximants [9, 156], to determine if they can form the basis of further SPKF implementational variations. Padé approximants consists of rational functions of polynomials, i.e.,

$$\begin{aligned} y &= g(x) \approx g_p(x) \\ &= \frac{b(x)}{a(x)}, \end{aligned}$$

where $b(x)$ and $a(x)$ are both interpolating polynomials in x . Due to this “ratio of polynomials” form, Padé approximants have increased fidelity for modeling certain nonlinearities for a given polynomial order. They are usually superior to Taylor expansions when functions contain poles, because the use of rational functions allows them to be well-represented. Systems with poles within the range of operation could possibly benefit from such approximations. Following a similar approach as for the derivation of the CDKF (based on Sterling interpolation), it is conceivable that a Padé approximants based SPKF can be derived and implemented.

- **Recurrent derivatives:** When using the EKF for inference in recursive structures such as the *training of recurrent neural networks* [158], or *dual-EKF time-series*

estimation [143], care must be taken to correctly account for the effect of recursive derivatives when explicitly (analytically) calculating the Jacobians of the system models with respect to either the model states and/or parameters. Since the SPKF's sigma-point approach does not explicitly calculate any analytical derivatives, but rather implicitly performs a statistical linearization of nonlinearities in question, it is not yet obvious what the SPKF-analogy of the EKF's recursive derivatives notion is. This is an open research question, which, although not directly addressed in this thesis, is relevant when applying the SPKF to recurrent (loopy) inference structures.

- **Iterated SPKF:** The *iterated EKF* (IEKF) algorithm [90, 62] has been proposed in the literature as an ad-hoc improvement of the standard EKF algorithm. The idea behind the IEKF is that the measurement update step of the normal EKF algorithm is *iterated* a number of times, relinearizing the nonlinear observation equation around the new state estimate, i.e.,

$$\hat{\mathbf{x}}_{k,i+1} = \hat{\mathbf{x}}_k^- + \mathbf{K}_{k,i} [\mathbf{y}_k - \mathbf{h}(\hat{\mathbf{x}}_{k,i}) - \mathbf{H}_{k,i} (\hat{\mathbf{x}}_k^- - \hat{\mathbf{x}}_{k,i})] ,$$

where $\hat{\mathbf{x}}_{k,i}$ is the i th iterate of the measurement updated state estimate, and $\mathbf{H}_{k,i} = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_{k,i}}$. The assumption is that the measurement updated state estimate might provide a “better” (closer to true value) linearization point around which to expand the nonlinear observation function, resulting in a smaller linearization error and hence better estimation accuracy. Some researchers have reported increased accuracy and robustness, as well as a smaller filter divergence likelihood when comparing the IEKF to the EKF algorithm. The downside to this is that the computational cost scales with the number of iterations, i.e., if N iterations are performed the cost goes up from $\mathcal{O}(L^3)$ to $\mathcal{O}(NL^3)$ in general. We suggest investigating the utility of extending this *iterated* measurement update approach to the SPKF framework. In order to do this, the sigma-point generation scheme must be adapted to calculate new sigma-points at each iteration from the latest (iteration) estimate of the posterior state distribution. Care must be taken though to ensure that the terms which are combined in the Kalman update, i.e., $\hat{\mathbf{x}}_k^-$ and $(\mathbf{y}_k - \hat{\mathbf{y}}_{k,i}^-)$, stay uncorrelated during the iteration

process.

- **Estimator integrated Bayesian optimal control:** The most straightforward way that a estimation system and control system are usually integrated, is through the use of a *separation principle*; both systems are designed independently and then one simply substitutes the estimates of the system state $\hat{\mathbf{x}}_k$ for the actual state in the feedback control law, i.e., $\mathbf{u}_k = \mathbf{K}(\hat{\mathbf{x}}_k)$, where \mathbf{K} is the control function. Within the optimal control framework, the control signal \mathbf{u}_k is calculated by minimizing a cost function $J_k(\mathbf{x}_k, \mathbf{u}_k)$. This is generally done by simply substituting the current estimate of the state $\hat{\mathbf{x}}_k$ into the cost function and then solving for the optimal \mathbf{u}_k . However, within the Bayesian estimation context the current estimate of the system state is in fact a *random variable* and should be treated as such when designing the control system. Given this, the control law should in fact be derived by minimizing the *expected cost*, $E[J_k(\mathbf{x}_k, \mathbf{u}_k)]$, where the expectation is taken with respect to the estimated posterior density, $\hat{p}(\mathbf{x}_k | \mathbf{y}_{1:k})$, which is provided by the Bayesian estimation framework. We call this approach *estimator integrated Bayesian optimal control*. Two possible approaches to deriving the control law in this framework include:

1. Instead of setting $\mathbf{u}_k = \mathbf{K}(\hat{\mathbf{x}}_k)$, we set $\mathbf{u}_k = E[\mathbf{K}(\mathbf{x}_k)]$, where the expectation is evaluated using either the SPKF framework (sigma-point approach) or using a more general SMC/SPKF hybrid approach such as the SPPF or GMSPPF. This improvement is analogous to the motivation of why the SPKF performs better than the EKF in calculating the optimal terms in the Kalman framework.
2. We can attempt to directly minimize $E[J_k(\mathbf{x}_k, \mathbf{u}_k)]$. If the control function $\mathbf{K}(\cdot)$ is given by a general nonlinear neural network³ for instance, the following approximate method can be used to minimize the expected cost function: We train the neural network using a set of sigma-points (SPKF) or particles (SMC/SPKF hybrid) calculated/drawn from the posterior state distribution $\hat{p}(\mathbf{x}_k | \mathbf{y}_{1:k})$ at every time step, as opposed to training using just $\hat{\mathbf{x}}_k$ as a single example. This corresponds to inferring a distribution over the input space to

³Such a control function is used in the *model predictive neural control* (MPNC) framework [22].

the control function, resulting in an optimization of the neural network with respect to this distribution.

While much of what is proposed here is speculative, we are currently in the process of investigating some of these ideas under a recently awarded NSF proposal⁴.

- **SPKF application to UAV autonomy:** The application of the SPKF framework to the problem of UAV autonomy as presented in Chapter 5 is an ongoing research project. Outstanding issues which are currently being addressed are the verification of the derived SPKF based state estimation system on real flight-data, as well as the migration of the system itself to the real flight hardware. This will require porting the Matlab code to a highly optimized C language formulation that can run within the asynchronous message passing environment of the QNX operating system used by the XCell platform’s flight computer. The final aim of the project is to make the SPKF based integrated navigation system completely self-contained with a well defined software and hardware interface, allowing for easy adaptation to any general aircraft equipped with GPS, IMU and possibly other avionics sensors. This will require further investigation into the robustness of the approach when used together with low cost, lower accuracy GPS and IMU sensors. This will include further experimental studies into the effect of IMU degradation and navigational anomalies such as GPS outages.
- **GMSPPF - Constrained clustering:** One of the crucial steps in the Gaussian-mixture sigma-point particle filter (GMSPPF) algorithm is the EM based re-clustering step used to reconstitute the posterior state GMM from the updated (and possibly resampled) posterior particle cloud. Naive clustering in this step can easily lead to suboptimal performance. This becomes especially important when using the GM-SPPF algorithm on applications that have strict physical constraints on the allowed shape of the posterior distribution. In the mobile robot localization application of Section 6.4.5 for example, due to the presence of physical barriers such as walls or

⁴NSF-ITR IIS-0313350 : “*Bayesian Integrated Vision, Estimation, and Control for Unmanned Aerial Vehicles*”

hallway corners, we might end up with groups of particles that are located close together (in free space), but lie on separate sides of a physical barrier. A naive (unconstrained) clustering approach will typically group these points together into a single cluster. In reality however, these points should be clustered into separate groups that take the physical constraints of the space into account. Using *constrained clustering* approaches with well designed data *priors* are recommended in these situations. Such techniques have recently received quite a bit of interest in the machine learning literature [198, 103] with specific application to data mining. Drawing from these techniques to improve the robustness of the GMSPPF algorithm is well worth investigating.

Although certainly not exhaustive, the above list contains ideas regarded by the author as the most promising in terms of their potential benefit to the research community. Some of these proposals are quite straight-forward; others represent a considerable amount of work. All of them would increase the impact of the SPKF paradigm on a variety of fields.

7.5 Derived & Related Work

The following research efforts have benefited in part, either directly or indirectly, from the work presented in this thesis, and can be interpreted as derived or related works. This list is not exhaustive, but rather lists some of the more prominent publications that has come to the authors attention.

- **Human face tracking:** As discussed in detail in Section 6.4.4, a research project by Rui at Microsoft Research [169, 29] has successfully applied our SPPF algorithm to the problem of human face tracking. Based on a direct C code implementation of the SPPF algorithm they developed a real-time system for the automated tracking (by camera) of a lecturer in an auditorium environment.
- **General visual contour tracking for human-computer interaction:** Subsequent to Rui's work on human face tracking using the SPPF, other researchers

have followed suite and opted to use the SPPF for user interface tracking purposes in their own human-computer interaction (HCI) systems [117]. See <http://choosh.bme.ogi.edu/demos> for more detail and demonstrations.

- **Nonlinear filtering for Command and Control applications:** Irwin et al. [86] successfully applied the SPPF algorithm to the problem of probabilistic inference in a military Command and Control (C2) scenario where an automated system is used to monitor and track objects in a battle-space. C2 involves a body of applications used by all branches of the armed forces. These include, but are not limited to, command applications, operations applications, intelligence applications, fire-support applications, logistics applications, and communication applications.
- **Ground target tracking:** Cui et al. [36] successfully applied the SPPF to the problem of tracking multiple ground vehicles. In a derived work, they extended the SPPF algorithm by leveraging Fox's KLD based particle set pruning method to adaptively determine the number of SPKF-particles needed to accurately model the posterior density. Although not improving on the estimation accuracy of the original SPPF, their new proposed approach has a slightly lower computational cost.
- **Robotics:** Mihaylova et al. used our work on the UKF, as well as the ReBEL toolkit implementation thereof, for application to the problem of active sensing of a nonholonomic wheeled mobile robot [134, 135].
- **Estimation architecture for future autonomous vehicles:** Brunke and Campbell [26] applied a square-root UKF formulation to the problem of estimating the nonlinear state and model parameters for an aircraft during failure, as well as generating aerodynamic models for potential on-line control reconfiguration.
- **ReBEL toolkit:** Numerous research groups have downloaded the ReBEL toolkit for use in a wide variety of probabilistic inference and machine learning problems. Please see Appendix C for a summary of some of these applications.
- **Tutorials:** A number of well received and widely cited tutorials in the sequential Monte Carlo field have been published that make specific mention of our work on

the SPKF and SPPF algorithms. The most noteworthy of these are [6] and [45].

Demonstrations of some of these applications discussed above can be viewed at the following URL: <http://choosh.bme.ogi.edu/demos>

Bibliography

- [1] AKASHI, H., AND KUMAMOTO, H. Random Sampling Approach to State Estimation in Switching Environments. *Automatica* 13 (1977), 429–434.
- [2] ALEXANDER, H. L. State Estimation for Distributed Systems with Sensing Delay. *SPIE : Data Structures and Target Classification* 1470 (1991), 103–111.
- [3] ALSPACH, D. L., AND SORENSEN, H. W. Nonlinear Bayesian Estimation using Gaussian Sum Approximation. *IEEE Transactions on Automatic Control* 17, 4 (1972), 439–448.
- [4] ANDERSON, B., AND MOORE, J. *Optimal Filtering*. Prentice-Hall, 1979.
- [5] ANDRIEU, C., DE FREITAS, N., DOUCET, A., AND JORDAN, M. An Introduction to MCMC for Machine Learning. *Machine Learning* 50 (Jan 2003), 5–43.
- [6] ARULAMPALAM, M. S., MASKELL, S., GORDON, N., AND CLAPP, T. A Tutorial on PArticle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing* 50, 2 (Feb 2002), 174–188.
- [7] AVITZOUR, D. A Stochastic Simulation Bayesian Approach to Multitarget Tracking. *IEE Proceedings on Radar, Sonar and Navigation* 142, 2 (1995), 41–44.
- [8] BAK, M., LARSEN, T. D., NORGAARD, M., ANDERSEN, N. A., POULSEN, N. K., AND RAVN, O. Location Estimation using Delayed Measurements. In *Proceedings of the 5th International Workshop on Advanced Motion Control (AMC98)* (Coimbra, Portugal, June 1998), pp. 180–185.
- [9] BAKER, G. A. J., AND GRAVES-MORRIS, P. *Padé Approximants*. Cambridge University Press, New York, 1996.
- [10] BEADLE, E. R., AND DJURIĆ, P. M. A Fast Weighted Bayesian Bootstrap Filter for Nonlinear Model State Estimation. *IEEE Transactions on Aerospace and Electronic Systems* 33, 1 (1997), 338–343.

- [11] BEAL, M., AND GHAHRAMANI, Z. Variational-Bayes.org : Online repository of papers, software, and links related to the use of variational methods for approximate Bayesian learning. <http://www.variational-bayes.org> [Viewed: October 15, 2003].
- [12] BELL, B. M., AND CATHEY, F. W. The Iterated Kalman Filter Update as a Gauss-Newton Method. *IEEE Transactions on Automatic Control* 38, 2 (February 1993), 294–297.
- [13] BELL, B. M., AND SCHUMITZKY, A. Asymptotic properties of extended least squares estimators, 1997. <http://www.seanet.com/~bradbell/elsq.htm> [Viewed: July 29, 2003].
- [14] BENGIO, Y. Markovian Models for Sequential Data. *Neural Computing Surveys* 2 (1999), 129–162.
- [15] BERTSEKAS, D. P. Incremental Least-Squares Methods and the Extended Kalman Filter. *SIAM Journal on Optimization* 6, 3 (August 1996), 807–822.
- [16] BERZUINI, C., BEST, N. G., GILKS, W. R., AND LARIZZA, C. Dynamic Conditional Independence Models and Markov Chain Monte Carlo Methods. *Journal of the American Statistical Association* 92, 440 (Dec. 1997), 1403–1412.
- [17] BIEZAD, D. J. *Integrated Navigation and Guidance Systems*. Education Series. AIAA, Reston, VA (USA), 1999.
- [18] BISHOP, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [19] BLACK, F., AND SCHOLES, M. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy* 81 (1973), 637–659.
- [20] BOGDANOV, A. A., AND WAN, E. A. SDRE Control with Nonlinear Feedforward Compensation for a Small Unmanned Helicopter. In *Proceedings of the 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations Conference* (San Diego, CA, September 2003). AIAA Paper Number: 2003-6512.
- [21] BOGDANOV, A. A., WAN, E. A., CARLSSON, M., KIEBURTZ, R., HARVEY, G., HUNT, J., AND VAN DER MERWE, R. State-Dependent Riccati Equation Control of a Small Unmanned Helicopter. In *Proceedings of the AIAA Guidance Navigation and Control Conference* (Austin, TX, August 2003). AIAA Paper Number: 2003-5672.

- [22] BOGDANOV, A. A., WAN, E. A., CARLSSON, M., ZHANG, Y., AND KIEBURTZ, R. Model Predictive Neural Control of a High Fidelity Helicopter Model. In *Proceedings of the AIAA Guidance Navigation and Control Conference* (Montreal, Canada, August 2001). AIAA Paper Number: 2001-4164.
- [23] BOGDANOV, A. A., WAN, E. A., CARLSSON, M., ZHANG, Y., KIEBURTZ, R., AND BAPTISTA, A. Model predictive neural control of a high fidelity helicopter model. In *Proceedings of the AIAA Guidance Navigation and Control Conference* (Montreal, Quebec, Canada, August 2001). AIAA Paper Number: 2001-4164.
- [24] BRAMWELL, A. R. S. *Bramwell's Helicopter Dynamics*. AIAA, Reston, VA, 2001.
- [25] BROWN, R. G., AND HWANG, P. Y. C. *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley and Sons, Inc., New York, 1992.
- [26] BRUNKE, S., AND CAMPBELL, M. E. Estimation Architecture for Future Autonomous Vehicles. In *Proceedings of the American Control Conference* (Anchorage, AK, 2002), AIAA, pp. 1108–1114.
- [27] CANNY, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8, 6 (November 1986), 679–698.
- [28] CHEN, G., WANG, J., AND SHIEH, L. S. Interval Kalman Filtering. *IEEE Transactions on Aerospace and Electronic Systems* 33, 1 (Jan 1997), 250–258.
- [29] CHEN, Y., HUANG, T., AND RUI, Y. Parametric Contour Tracking using the Unscented Kalman Filter. In *Proceedings of the International Conference on Image Processing (ICIP)* (2002), vol. 3, pp. 613–616.
- [30] CLOUTIER, J. R., D'SOUZA, C. N., AND MRACEK, C. P. Nonlinear regulation and nonlinear H-infinity control via the state-dependent Riccati equation technique: Part1, Theory. In *Proceedings of the International Conference on Nonlinear Problems in Aviation and Aerospace* (Daytona Beach, FL, May 1996), pp. 117–130.
- [31] CLOUTIER, J. R., D'SOUZA, C. N., AND MRACEK, C. P. Nonlinear regulation and nonlinear H-infinity control via the state-dependent Riccati equation technique: Part2, Examples. In *Proceedings of the International Conference on Nonlinear Problems in Aviation and Aerospace* (Daytona Beach, FL, May 1996), pp. 131–141.
- [32] COX, I. J., AND WILFONG, G. T., Eds. *Autonomous Robot Vehicles*. Springer Verlag, 1990.

- [33] COX, R. T. Probability, frequency, and reasonable expectation. *American Journal of Physics* 14, 1 (1946), 1–13.
- [34] CRISAN, D., AND DOUCET, A. Convergence of Sequential Monte Carlo Methods. Tech. Rep. CUED/F-INFENG/TR381, Dept. of Engineering, University of Cambridge, 2000.
- [35] CROSSBOW TECHNOLOGY, INC. Crossbow: Smarter Sensors in Silicon. <http://www.xbow.com> [Viewed: October 19, 2003].
- [36] CUI, N., HONG, L., AND LAYNE, J. R. A Comparison of Nonlinear Filtering Approaches with Application to Ground Target Tracking. *Signal Processing (EURASIP/Elsevier)* (2003). Submitted.
- [37] DAHLQUIST, G., AND BJÖRCK, A. *Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1974.
- [38] DARPA. Software Enabled Control (SEC) Program. <http://dtsn.darpa.mil/ixo/programdetail.asp?progid=39> [Viewed: July 30, 2003].
- [39] DE FREITAS, J. F. G. *Bayesian Methods for Neural Networks*. PhD thesis, Cambridge University Engineering Department, 1999.
- [40] DE FREITAS, J. F. G., NIRANJAN, M., GEE, A. H., AND DOUCET, A. Sequential Monte Carlo Methods for Optimisation of Neural Network Models. Tech. Rep. CUED/F-INFENG/TR 328, Department of Engineering, University of Cambridge, 1998.
- [41] DEMPSTER, A., LAIRD, N. M., AND RUBIN, D. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B* 39 (1977), 1–38.
- [42] DITTRICH, J. S. Design and Integration of an Unmanned Aerial Vehicle Navigation System. Master's thesis, School of Aerospace Engineering, Georgia Institute of Technology, May 2002.
- [43] DOUCET, A. *Monte Carlo Methods for Bayesian Estimation of Hidden Markov Models: Application to Radiation Signals*. PhD thesis, University Paris-Sud, Orsay, France, 1997.
- [44] DOUCET, A. On Sequential Simulation-Based Methods for Bayesian Filtering. Tech. Rep. CUED/F-INFENG/TR 310, Department of Engineering, University of Cambridge, 1998.

- [45] DOUCET, A., DE FREITAS, N., AND GORDON, N. *Sequential Monte-Carlo Methods in Practice*. Springer-Verlag, April 2001.
- [46] DOUCET, A., GODSILL, S. J., AND ANDRIEU, C. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing* 10, 3 (2000), 197–208.
- [47] DOUCET, A., GORDON, N. J., AND KRISHNAMURTHY, V. Particle Filters for State Estimaion of Jump Markov Linear Systems. Tech. Rep. CUED/F-INFENG/TR 359, Cambridge University Engineering Department, Cambridge, United Kingdom, 1999.
- [48] EFRON, B. *The Bootstrap, Jackknife and other Resampling Plans*. SIAM, Philadelphia, 1982.
- [49] ESA. Mars Express / Beagle-2 Mission. <http://sci.esa.int/science-e/www/area/index.cfm?fareaid=9> [Viewed: July 21, 2003].
- [50] FAHRMEIR, L., AND KAUFMANN, H. On Kalman Filtering, Posterior Mode Estimation and Fisher Scoring in Dynamic Exponential Family Regression. *Metrika* 38 (1991), 37–60.
- [51] FERGUSON, P. F., AND HOW, J. Decentralized Estimation Algorithms for Formation Flying Spacecraft. In *Proceedings of the AIAA Guidance Navigation and Control Conference* (Austin, TX, August 2003). AIAA Paper Number: 2003-5442.
- [52] FERON, E. Aerial robotics project : Laboratory for information and decision systems mit. <http://gewurtz.mit.edu/research/heli.htm> [Viewed: September 11, 2003].
- [53] FOX, D. KLD-Sampling: Adaptive Particle Filters. In *Advances in Neural Information Processing Systems* 14 (2001), pp. 713–720.
- [54] FOX, D. KLD-Sampling: Adaptive Particle Filters and Mobile Robot Localization. Tech. Rep. UW-CSE-01-08-02, University of Washington, Jan. 2002.
- [55] FOX, D., THRUN, S., DELLAERT, F., AND BURGARD, W. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2000, ch. Particle filters for mobile robot localization., pp. 401–428.
- [56] FRAZZOLI, E. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2001.
- [57] FRÖBERG, C. E. *Introduction to Numerical Analysis*. Addison-Wesley, Reading, MA, 1970.

- [58] GAVRILETS, V. *Autonomous Aerobatic Maneuvering of Miniature Helicopters: Modeling and Control*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003.
- [59] GAVRILETS, V. MIT X-Cell-60 UAV Platform EKF Estimator. Private Communications, February 2003.
- [60] GAVRILETS, V., MARTINOS, I., METTLER, B., AND FERON, E. Flight Test and Simulation Results for an Autonomous Aerobic Helicopter. In *Proceedings of 21st AIAA/IEEE Digital Avionics Systems Conference* (Indianapolis, October 2002), vol. 2, pp. 8C31–8C36.
- [61] GAVRILETS, V., METTLER, B., AND FERON, E. Nonlinear Model for a Small-size Acrobatic Helicopter. In *Proceedings of the AIAA Guidance Navigation and Control Conference* (Montreal, Canada, August 2001). AIAA Paper Number: 2001-4333.
- [62] GELB, A., Ed. *Applied Optimal Estimation*. MIT Press, 1974.
- [63] GEWEKE, J. Bayesian Inference in Econometric Models using Monte Carlo Integration. *Econometrica* 24 (1989), 1317–1399.
- [64] GHAHRAMANI, Z., AND BEAL, M. Variational Inference for Bayesian Mixture of Factor Analysers. In *Advances in Neural Information Processing Systems 12* (1999), pp. 449–455. <http://citeseer.nj.nec.com/ghahramani00variational.html> [Viewed: September 8th, 2003].
- [65] GHAHRAMANI, Z., AND ROWEIS, S. Probabilistic Models for Unsupervised Learning. Tutorial presented at the 1999 NIPS Conference. <http://www.gatsby.ucl.ac.uk/~zoubin/NIPSTutorial.html> [Viewed: October 15, 2003].
- [66] GHAHRAMANI, Z., AND ROWEIS, S. T. Learning nonlinear dynamical systems using an EM algorithm. In *Advances in Neural Information Processing Systems 11: Proceedings of the 1998 Conference* (1999), M. J. Kearns, S. A. Solla, and D. A. Cohn, Eds., MIT Press, pp. 431–437.
- [67] GOEL, A. *Operating System Support for Low-Latency Streaming*. PhD thesis, OGI School of Science & Engineering, OHSU, Portland, OR, USA, July 2003.
- [68] GORDON, N. J., SALMOND, D. J., AND SMITH, A. F. M. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F* 140, 2 (Apr. 1993), 107–113.

- [69] HAMMERSLEY, J. M., AND MORTON, K. W. Poor Man's Monte Carlo. *Journal of the Royal Statistical Society B* 16 (1954), 23–38.
- [70] HANDSCHIN, J. E. Monte Carlo Techniques for Prediction and Filtering of Non-Linear Stochastic Processes. *Automatica* 6 (1970), 555–563.
- [71] HANDSCHIN, J. E., AND MAYNE, D. Q. Monte Carlo Techniques to Estimate the Conditional Expectation in Multi-Stage Non-Linear Filtering. *International Journal of Control* 9, 5 (1969), 547–559.
- [72] HANSEN, E. *Global Optimization Using Interval Analysis*. Monographs and Textbooks in Pure and Applied Mathematics. Marcel Dekker, Inc., 1992.
- [73] HAUGEN, R. A. *Modern Investment Theory*. Prentice-Hall International, New Jersey, 1990.
- [74] HAYKIN, S. *Neural Networks : A Comprehensive Foundation*. Macmillan College Publishing Company, Inc, Englewood Cliffs, NJ, 1994.
- [75] HAYKIN, S. *Adaptive Filter Theory*, 3 ed. Prentice-Hall, Inc, 1996.
- [76] HAYKIN, S., Ed. *Kalman Filtering and Neural Networks*. Wiley, 2001.
- [77] HAYKIN, S., Ed. *Kalman Filtering and Neural Networks*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. Wiley, 2001, ch. 7 - The Unscented Kalman Filter, E. A. Wan and R. van der Merwe, pp. 221–280.
- [78] HIGUCHI, T. Monte Carlo Filter Using the Genetic Algorithm Operators. *Journal of Statistical Computation and Simulation* 59, 1 (1997), 1–23.
- [79] HILDEBRAND, F. B. *Introduction to Numerical Analysis*. McGraw-Hill, New-York, 1956.
- [80] HOAG, D. Apollo Guidance and Navigation: Considerations of Apollo IMU Gimbal Lock. Tech. Rep. E-1344, MIT Instrumentation Laboratory, April 1963. <http://www.hq.nasa.gov/office/pao/History/alsj/e-1344.htm> [Viewed: September 15, 2003].
- [81] HONEYWELL SOLID STATE ELECTRONICS. Honeywell Precision Barometer: HPA & HPB User Manual. <http://www.ssec.honeywell.com/pressure/datasheets/hpbmanual.pdf> [Viewed: September 19, 2003].
- [82] HULL, J. C. *Options, Futures, and Other Derivatives*, third ed. Prentice Hall, 1997.

- [83] IKEDA, K. Multiple-valued stationary state and its instability of light by a ring cavity system. *Optics Communications* 30, 2 (August 1979), 257–261.
- [84] INERTIAL SCIENCE, INC. ISIS-GPS: Integrated INS/GPS GNS System. <http://www.inertialscience.com> [Viewed: October 19, 2003].
- [85] INERTIAL SCIENCE, INC. ISIS-IMU. <http://www.inertialscience.com/isis-imu-new.htm> [Viewed: September 19, 2003].
- [86] IRWIN, M. E., CRESSIE, N., AND JOHANNESSEN, G. Spatial-Temporal Nonlinear Filtering Based on Hierarchical Statistical Models. *Sociedad de Estadistica e Investigacion Operativa : Test* 11, 2 (2002), 249–302.
- [87] ISARD, M., AND BLAKE, A. Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision* (Cambridge, UK, 1996), pp. 343–356.
- [88] ITO, K., AND XIONG, K. Gaussian Filters for Nonlinear Filtering Problems. *IEEE Transactions on Automatic Control* 45, 5 (May 2000), 910–927.
- [89] JAYNES, E. T. *Probability Theory : The Logic of Science*. Cambridge University Press, 2003.
- [90] JAZWINSKY, A. *Stochastic Processes and Filtering Theory*. Academic Press, New York., 1970.
- [91] JONES, E. M., AND FJELD, P. Gimbal Angles, Gimbal Lock, and a Fourth Gimbal for Christmas, Nov 2002. <http://www.hq.nasa.gov/office/pao/History/alsj/gimbals.html> [Viewed: September 15, 2003].
- [92] JORDAN, M. I., GHAHRAMANI, Z., JAAKKOLA, T. S., AND SAUL, L. K. *M. I. Jordan, (Ed.), Learning in Graphical Models*. Kluwer Academic Publishers, 1998, ch. An Introduction to Variational Methods for Graphical Models.
- [93] JORDAN, M. I., SEJNOWSKI, T. J., AND POGGIO, T., Eds. *Graphical Models: Foundations of Neural Computation*. MIT Press, 2001.
- [94] JULIER, S., UHLMANN, J., AND DURRANT-WHYTE, H. A new approach for filtering nonlinear systems. In *Proceedings of the American Control Conference* (1995), pp. 1628–1632.
- [95] JULIER, S. J. *Comprehensive Process Models for High-Speed Navigation*. PhD thesis, University of Oxford, England, 1997.

- [96] JULIER, S. J. A Skewed Approach to Filtering. In *SPIE Conference on Signal and Data Processing of Small Targets* (Orlando, Florida, April 1998), vol. 3373, SPIE, pp. 271–282.
- [97] JULIER, S. J. The Scaled Unscented Transformation. In *Proceedings of the American Control Conference* (May 2002), vol. 6, pp. 4555–4559.
- [98] JULIER, S. J. Ideas on time-delayed fusion. Private communications, April 2003.
- [99] JULIER, S. J., AND UHLMANN, J. K. A General Method for Approximating Nonlinear Transformations of Probability Distributions. Tech. rep., RRG, Dept. of Engineering Science, University of Oxford, Nov 1996. <http://citeseer.nj.nec.com/julier96general.html> [Viewed: September 8th, 2003].
- [100] JULIER, S. J., AND UHLMANN, J. K. A New Extension of the Kalman Filter to Nonlinear Systems. In *Proc. SPIE - Int. Soc. Opt. Eng. (USA)* (Orlando, FL, April 1997), vol. 3068, pp. 182–193.
- [101] JULIER, S. J., AND UHLMANN, J. K. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE* 92, 3 (March 2004), 401–422.
- [102] KALMAN, R. E. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering* (1960), 35–45.
- [103] KAMVAR, S. D. Constrained Clustering for Improved Pattern Discovery, July 2002. <http://www.stanford.edu/~sdkamvar/talks/constrained-clustering.ppt> [Viewed: Novermber 5, 2003].
- [104] KAY, S. M. *Estimation Theory*, vol. 1 of *Fundamentals of Statistical Signal Processing*. Prentice Hall PTR, 1993.
- [105] KIEBURTZ, R. Model-Relative Control of Autonomous Vehicles : A project of the DARPA Software Enabled Control Program. <http://www.cse.ogi.edu/PacSoft/projects/sec/> [Viewed: July 30, 2003].
- [106] KITAGAWA, G. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics* 5 (1996), 1–25.
- [107] KOLNICK, F. QNX 4 Real-Time Operating System. Basis Computer Systems, Canada, 1998.
- [108] KONG, A., LIU, J. S., AND WONG, W. H. Sequential Imputations and Bayesian Missing Data Problems. *Journal of the American Statistical Association* 89, 425 (1994), 278–288.

- [109] KREYSZIG, E. *Advanced Engineering Mathematics*, 6 ed. Wiley, 1988.
- [110] LAPEDES, A., AND FARBER, R. Nonlinear Signal Processing using Neural Networks: Prediction and System modelling. Tech. Rep. LAUR872662, Los Alamos National Laboratory, 1987.
- [111] LARSEN, T. D., ANDERSEN, N. A., AND RAVN, O. Incorporation of Time Delayed Measurements in a Discrete-time Kalman Filter. In *Proceedings of the 37th IEEE Conference on Decision & Control* (Tampa, Florida, USA, Dec 1998), pp. 3972–3977.
- [112] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (November 1998), 2278–2324.
- [113] LECUN, Y., MATAN, O., BOSER, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W., JACKEL, L. D., AND BAIRD, H. S. Handwritten zip code recognition with multilayer networks. In *Proc. of the International Conference on Pattern Recognition* (Atlantic City, 1990), IAPR, Ed., vol. II, IEEE, pp. 35–40.
- [114] LEFEBVRE, T., BRUYNINCKX, H., AND DE SCHUTTER, J. 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), 1406–1409.
- [115] LEONARD, J., AND DURRANT-WHYTE, H. F. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Press, Boston, MA, USA, 1991.
- [116] LEWIS, F. L. *Optimal Estimation*. John Wiley & Sons, Inc., New York, 1986.
- [117] LI, P., AND ZHANG, T. Visual Contour Tracking Based on Particle Filters. In *Proceedings of the First International Workshop on Generative-Model-Based Vision (GMBV)* (Copenhagen, June 2002), pp. 61–70. <http://www.diku.dk/publikationer/tekniske.rapporter/2002/02-01/> [Viewed: July 31, 2003].
- [118] LIU, J., AND CHEN, R. Sequential Monte Carlo Methods for Dynamic Systems. *Journal of the American Statistical Association* 93 (1998), 1032–1044.
- [119] LIU, J., CHEN, R., AND LOGVINENKO, T. A Theoretical Framework for Sequential Importance Sampling with Resampling. In *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. J. Gordon, Eds. Springer-Verlag, 2000, pp. 225–242.

- [120] LIU, J. S., AND CHEN, R. Blind Deconvolution via Sequential Imputations. *Journal of the American Statistical Association* 90, 430 (1995), 567–576.
- [121] LIU, Q., RUI, Y., GUPTA, A., AND CADIZ, J. J. Automating Camera Management in a Lecture Room Environments. In *Proceedings of ACM Conference on Human Factors in Computing Systems* (Seattle, WA, March 2001), pp. 442–449.
- [122] LJUNG, L. Convergence Analysis of Parametric Identification Methods. *IEEE Transaction on Automatic Control* 23 (1978), 770–783.
- [123] LJUNG, L. Asymptotic Behaviour of the Extended Kalman Filter as a Parameter Estimator for Linear Systems. *IEEE Transactions on Automatic Control AC-24*, 1 (1979), 36–50.
- [124] LJUNG, L., AND SÖDERSTRÖM, T. *Theory and Practice of Recursive Identification*. MIT Press, Cambridge, MA, 1983.
- [125] LJUNGQUIST, D., AND BALCHEN, J. G. Recursive Prediction Error Methods for Online Estimation in Nonlinear State-Space Models. *Modeling, Identification and Control* 15, 2 (April 1994), 109–121.
- [126] LUENBERGER, D. G. *Linear and Nonlinear Programming*, 2 ed. Addison-Wesley, 1984.
- [127] MACKAY, D. J. C. Robot-Arm Dataset. <http://wol.ra.phy.cam.ac.uk/mackay/SourceData.html> [Viewed: August 31, 2003].
- [128] MACKAY, D. J. C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation* 4 (1992), 448–472.
- [129] MACKEY, M. C., AND GLASS, L. Oscillation and chaos in a physiological control system. *Science* 197, 4300 (June 1977), 287–289.
- [130] MAGELLAN CORPORATION, SANTA CLARA, CA. Ashtech G12 GPS Sensor. <http://www.navtechgps.com/supply/g12sens.asp> [Viewed: September 19, 2003].
- [131] MATTHEWS, M. B. A state-space approach to adaptive nonlinear filtering using recurrent neural networks. In *Proceedings IASTED Internat. Symp. Artificial Intelligence Application and Neural Networks* (1990), pp. 197–200.
- [132] MCLACHLAN, G., AND KRISHNAN, T. *The EM Algorithm and Extensions*. Wiley, 1997.

- [133] MEILIJSEN, I. A fast improvement to the EM algorithm on its own terms. *Journal of the Royal Statistical Society B* 51 (1989), 127–138.
- [134] MIHAYLOVA, L., BRUYNINCKX, H., AND DE SCHUTTER, J. Active Sensing of a Nonholonomic Wheeled Mobile Robot. In *Proceedings of the IEEE Benelux Meeting* (Leuven, Belgium, March 2002), pp. 125–128.
- [135] MIHAYLOVA, L., DE SCHUTTER, J., AND BRUYNINCKX, H. A Multisine Approach for Trajectory Optimization Based on Information Gain. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems* (Lauzanne, Switzerland, September 2002), pp. 661–666.
- [136] MINIATURE AIRCRAFT USA (ORLANDO, FL). X-Cell Line of R/C Helicopters. <http://www.x-cellrc helicopters.com> [Viewed: September 11, 2003].
- [137] MOORE, A. Very Fast EM-based Mixture Model Clustering Using Multiresolution KD-trees. In *Advances in Neural Information Processing Systems* (340 Pine Street, 6th Fl., San Francisco, CA 94104, April 1999), M. Kearns and D. Cohn, Eds., Morgan Kaufman, pp. 543–549.
- [138] MOORE, G. E. Cramming more components onto integrated circuits. *Electronics* 38, 8 (April 1965). <http://www.intel.com/research/silicon/mooreslaw.htm> [Viewed: March 11, 2004].
- [139] MOORE, R. E. *Interval Analysis*. Prentice Hall, 1966.
- [140] MURPHY, K. P. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, July 2002.
- [141] NASA/JPL. Mars Exploration Rover Mission. <http://mars.jpl.nasa.gov/mer> [Viewed: July 21, 2003].
- [142] NAVSYS CORPORATION. NAVSYS Corporation: GPS/Inertial Products. <http://www.navsys.com> [Viewed: October 19, 2003].
- [143] NELSON, A. T. *Nonlinear Estimation and Modeling of Noisy Time-Series by Dual Kalman Filtering Methods*. PhD thesis, Oregon Graduate Institute, 2000.
- [144] NELSON, A. T., AND WAN, E. A. Neural Speech Enhancement Using Dual Extended Kalman Filtering. In *Proceedings of International Conference on Neural Networks (ICNN)* (Houston, TX, June 1997), vol. 4, pp. 2171–2175.

- [145] NELSON, A. T., AND WAN, E. A. A Two-Observation Kalman Framework for Maximum-Likelihood Modeling of Noisy Time Series. In *Proceedings of the IEEE International Joint Conference on Neural Networks* (Anchorage, AK, May 1998), pp. 2489–2494.
- [146] NIRANJAN, M. Sequential Tracking in Pricing Financial Options Using Model Based and Neural Network Approaches. In *Advances in Neural Information Processing Systems (NIPS)* (1996), M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., vol. 8, pp. 960–966.
- [147] NØRGAARD, M., POULSEN, N., AND RAVN, O. Advances in Derivative-Free State Estimation for Nonlinear Systems. Tech. Rep. IMM-REP-1998-15, Dept. of Mathematical Modelling, Technical University of Denmark, 28 Lyngby, Denmark, April 2000.
- [148] NORGAARD, M., POULSEN, N., AND RAVN, O. New Developments in State Estimation for Nonlinear Systems. *Automatica* 36, 11 (November 2000), 1627–1638.
- [149] ONR. Autonomous Operations Future Naval Capability (ONR-AO-FNC). http://www.onr.navy.mil/fncts/auto_ops/ [Viewed: July 30, 2003].
- [150] OSBORNE, M. Fisher's Method of Scoring. *International Statistical Review*, 60 (1992), 99–117.
- [151] PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [152] PEEBLES, JR., P. Z. *Probability, Random Variables, and Random Signal Principles*. McGraw-Hill, 1993.
- [153] PELLEG, D., AND MOORE, A. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning* (San Francisco, 2000), Morgan Kaufmann, pp. 727–734.
- [154] PITT, M. K., AND SHEPHARD, N. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association* 94, 446 (1999), 590–599.
- [155] POLE, A., WEST, M., AND HARRISON, P. J. Non-normal and Nonlinear Dynamic Bayesian Modelling. In *Bayesian Analysis of Time Series and Dynamic Models*, J. C. Spall, Ed. Marcel Dekker, New York, 1988.

- [156] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes in C : The Art of Scientific Computing*, 2 ed. Cambridge University Press, 1992.
- [157] PUSKORIUS, G., AND FELDKAMP, L. Decoupled Extended Kalman Filter Training of Feedforward Layered Networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (Seattle, WA, July 1991), vol. 1, pp. 771–777.
- [158] PUSKORIUS, G., AND FELDKAMP, L. Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks. *IEEE Transactions on Neural Networks* 5, 2 (1994), 279–297.
- [159] PUSKORIUS, G., AND FELDKAMP, L. Extensions and Enhancements of Decoupled Extended Kalman Filter Training. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN)* (Houston, TX, June 1997), vol. 3, pp. 1879–1883.
- [160] PUSKORIUS, G. V., FELDKAMP, L. A., AND DAVIS, L. I. Dynamic Neural Network Methods Applied to On-Vehicle Idle Speed Control. *Proceedings of the IEEE 84* (Oct 1996), 1407–1419.
- [161] RAO, S. S. *Engineering Optimization: Theory and Practice*. John Wiley & Sons, 1996.
- [162] RIX, A. W., BEERENDS, J. G., HOLLIER, M. P., AND HEKSTRA, A. P. Perceptual Evaluation of Speech Quality (PESQ) - A New Method for Speech Quality Assessment of Telephone Networks and Codecs. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Salt Lake City, UT, May 2001), pp. 749–752.
- [163] ROBBINS, H., AND MONRO, S. A Stochastic Approximation Method. *The Annals of Mathematical Statistics* 22 (1951), 400–407.
- [164] ROLFE, J. M., AND STAPLES, K. J. *Flight Simulation*. Cambridge University Press, United Kingdom, 1986.
- [165] ROSENBLUTH, M. N., AND ROSENBLUTH, A. W. Monte Carlo Calculation of the Average Extension of Molecular Chains. *Journal of Chemical Physics* 23 (1955), 356–359.
- [166] ROSENBROCK, H. H. An Automatic Method for Finding the Greatest or Least Value of a Function. *Computer Journal* 3 (1960), 175–184.

- [167] ROWEIS, S., AND GHAHRAMANI, Z. A Unifying Review of Linear Gaussian Models. *Neural Computation* 11, 2 (1999), 305–345.
- [168] RUBIN, D. B. Using the SIR Algorithm to Simulate Posterior Distributions. In *Bayesian Statistics 3* (Cambridge, MA, 1988), J. M. Bernardo, M. H. DeGroot, D. V. Lindley, and A. F. M. Smith, Eds., Oxford University Press, pp. 395–402.
- [169] RUI, Y., AND CHEN, Y. Better Proposal Distributions: Object Tracking Using Unscented Particle Filter. In *Proc. of IEEE CVPR* (Kauai, Hawaii, Dec 2001), vol. II, pp. 786–793.
- [170] SAYED, A. H., AND KAILATH, T. A State-Space Approach to Adaptive RLS Filtering. *IEEE Signal Processing Magazine* 11, 3 (July 1994), 18–60.
- [171] SCHEI, T. S. A Finite-difference Method for Linearizing in Nonlinear Estimation Algorithms. *Automatica* 33, 11 (1997), 2051–2058.
- [172] SCHMIDT, S. F. *C. T. Leondes, editor, Advances in Control Systems*, vol. 3. Academic Press, 1966, ch. Applications of State Space Methods to Navigation Problems, pp. 293–340.
- [173] SCHOLTE, E., AND CAMPBELL, M. E. On-line Nonlinear Guaranteed Estimation with Application to a High Performance Aircraft. In *Proceedings of the American Control Conference* (Anchorage, AK, May 2002), pp. 184–190.
- [174] SCHOLTE, E., AND CAMPBELL, M. E. A Nonlinear Set-Membership Filter for On-line Applications. *International Journal of Robust Nonlinear Control* 13, 15 (December 2003), 1337–1358. <http://www.mae.cornell.edu/sec/Papers/ESMF-stability.pdf> [Viewed: November 7, 2003].
- [175] SHOEMAKE, K. Animating Rotation with Quaternion Curves. *ACM SIGGRAPH* 19, 3 (1985), 245–254.
- [176] SHUMWAY, R. H., AND STOFFER, D. S. An approach to time series smoothing and forecasting using the EM algorithm. *J. Time Series Analysis* 3, 4 (1982), 253–264.
- [177] SHUSTER, M. D. A Survey of Attitude Representations. *The Journal of the Astronautical Sciences* 41, 4 (October 1993), 439–517.
- [178] SINGHAL, S., AND WU, L. Training multilayer perceptrons with the extended Kalman filter. In *Advances in Neural Information Processing Systems 1* (San Mateo, CA, 1989), Morgan Kauffman, pp. 133–140.

- [179] SIOURIS, G. M., CHEN, G., AND WANG, J. Tracking an Incomming Ballistic Missile Using an Extended Interval Kalman Filter. *IEEE Transactions on Aerospace and Electronic Systems* 33, 1 (Jan 1997), 232–240.
- [180] SMITH, A. F. M., AND GELFAND, A. E. Bayesian Statistics without Tears: a Sampling-Resampling Perspective. *American Statistician* 46, 2 (1992), 84–88.
- [181] STEVENS, B., AND LEWIS, F. *Aircraft Control and Simulation*. Wiley, New York, NY, 1992.
- [182] STIRLING, J. Methodus differentialis, sive tractatus de summation et interpolation serierum infinitarum, 1730.
- [183] SUM, J. P. F., LEUNG, C. S., AND CHAN, L. W. Extended Kalman Filter in Recurrent Neural Network Trainin and Pruning. Tech. Rep. CS-TR-96-05, Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong, May 1996.
- [184] TAWEL, R., ARANKI, N., PUSKORIUS, G. V., MARKO, K. A., FELDKAMP, L. A., JAMES, J. V., JESION, G., AND FELDKAMP, T. M. Custom VLSI ASIC for Automotive Applications with Recurrent Networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks* (Anchorage, AK, May 1998), pp. 598–602.
- [185] THALES NAVIGATION, INC. Thales Navigation & Magellan GPS Systems. <http://www.thalesnavigation.com> [Viewed: October 19, 2003].
- [186] THE MATHWORKS, INC. Matlab. <http://www.mathworks.com> [Viewed: October 11, 2003].
- [187] THE MATHWORKS, INC. Matlab Online Documentation. <http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml> [Viewed: October 1, 2003].
- [188] VAN DER MERWE, R. Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models. In *Workshop on Advances in Machine Learning* (Montreal, June 2003). <http://www.iro.umontreal.ca/~kegl/CRMWorkshop/program.html> [Viewed: July 21, 2003].
- [189] VAN DER MERWE, R., DE FREITAS, N., DOUCET, A., AND WAN, E. The unscented particle filter. Tech. Rep. CUED/F-INFENG/TR 380, Cambridge University Engineering Department, Aug. 2000.

- [190] VAN DER MERWE, R., DE FREITAS, N., DOUCET, A., AND WAN, E. The Unscented Particle Filter. In *Advances in Neural Information Processing Systems 13* (Nov 2001), pp. 584–590.
- [191] VAN DER MERWE, R., AND WAN, E. Efficient Derivative-Free Kalman Filters for Online Learning. In *Proceedings of the 9th European Symposium on Artificial Neural Networks (ESANN)* (Bruges, Belgium, Apr 2001), pp. 205–210.
- [192] VAN DER MERWE, R., AND WAN, E. The Square-Root Unscented Kalman Filter for State- and Parameter-Estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (Salt Lake City, UT, May 2001), vol. 6, pp. 3461–3464.
- [193] VAN DER MERWE, R., AND WAN, E. Gaussian Mixture Sigma-Point Particle Filters for Sequential Probabilistic Inference in Dynamic State-Space Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (Hong Kong, April 2003), vol. 6, pp. 701–704.
- [194] VAN DER MERWE, R., AND WAN, E. A. *ReBEL : Recursive Bayesian Estimation Library for Matlab*. <http://choosh.bme.ogi.edu/rebel/index.html> [Viewed: October 11, 2003].
- [195] VAN DER MERWE, R., WAN, E. A., AND JULIER, S. J. Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion: Applications to Integrated Navigation. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (Providence, RI, August 2004).
- [196] VERMA, V., GORDON, G., AND SIMMONS, R. Efficient Monitoring of Planetary Rovers. In *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)* (Nara, Japan, May 2003). http://www.ri.cmu.edu/pubs/pub_4424.html [Viewed: March 11, 2004].
- [197] VERMAAK, J., AND BLAKE, A. Nonlinear Filtering for Speaker Tracking in Noisy and Reverberant Environments. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (Salt Lake City, UT, May 2001), vol. 5, pp. 3021–3024.
- [198] WAGSTAFF, K., CARDIE, C., ROGERS, S., AND SCHROEDL, S. Constrained K-means Clustering with Background Knowledge. In *Proceedings of the International Conference on Machine Learning (ICML)* (Williams College, Massachusetts, Jun 2001), pp. 577–584.

- [199] WAN, E., AND VAN DER MERWE, R. Noise-Regularized Adaptive Filtering for Speech Enhancement. In *Proceedings of the 6th European Conference on Speech Communication and Technology (Eurospeech)* (Budapest, Hungary, September 1999), vol. 6, pp. 2643–2646.
- [200] WAN, E., VAN DER MERWE, R., AND NELSON, A. Dual Estimation and the Unscented Transformation. In *Neural Information Processing Systems 12* (Nov 2000), S. A. Solla, T. K. Leen, and K. R. Müller, Eds., MIT Press, pp. 666–672.
- [201] WAN, E. A., AND BOGDANOV, A. A. Model Predictive Neural Control with Applications to a 6DOF Helicopter Model. In *Proceedings of the American Control Conference* (Arlington, VA, June 2001), vol. 1, pp. 488–493.
- [202] WAN, E. A., AND NELSON, A. T. Neural Dual Extended Kalman Filtering: Applications in Speech Enhancement and Monaural Blind Signal Separation. In *Proceedings of the IEEE Signal Processing Society Neural Networks for Signal Processing Workshop* (Amelia Island, FL, September 1997), pp. 466–475.
- [203] WAN, E. A., AND VAN DER MERWE, R. Sigma-Point Kalman Filter Based Sensor Integration, Estimation and System Identification for Enhanced UAV Situational Awareness & Control : A project under the ONR Autonomous Operations Future Naval Capability (ONR-AO-FNC) - UAV Autonomy Program. <http://choosh.bme.ogi.edu/onr/> [Viewed: July 30, 2003].
- [204] WAN, E. A., AND VAN DER MERWE, R. The Unscented Kalman Filter for Nonlinear Estimation. In *Proceedings of IEEE Symposium on Adaptive Systems for Signal Processing Communications and Control (AS-SPCC)* (Lake Louise, Alberta, Canada, October 2000), pp. 153–158.
- [205] WEISSTEIN, E. W. mathworld.wolfram.com : Hermite-Gauss Quadrature. <http://mathworld.wolfram.com/Hermite-GaussQuadrature.html> [Viewed: September 9th, 2003].
- [206] WOODWARD, J. D. Biometrics: Facing Up to Terrorism, 2001. RAND/IP-218-A, <http://www.rand.org/publications/IP/IP218/> [Viewed: July 21, 2003].
- [207] WOODWARD, J. D. Super Bowl Surveillance: Facing Up to Biometrics, 2001. RAND/IP-209, <http://www.rand.org/publications/IP/IP209/> [Viewed: July 21, 2003].

- [208] ZARITSKII, V. S., SVETNIK, V. B., AND SHIMELEVICH, L. I. Monte-Carlo Techniques in Problems of Optimal Information Processing. *Automation and Remote Control* 36, 3 (1975), 2015–2022.

Appendix A

The Kalman Filter

A.1 Introduction

This appendix provides most of the background theoretical derivations related to Kalman filter theory.

A.2 MMSE Derivation of Kalman Filter

There is a common misconception that the Kalman filter can only be strictly applied to linear systems with Gaussian random variables (RV), i.e. a sequential probabilistic inference problems on linear DSSMs where the prior random variables and noise densities are Gaussian. However, Kalman's original derivation of the Kalman filter did not apply Bayes Rule and does not require the exploitation of any specific probability density function [102]. His only assumptions were that the system state RV could be consistently estimated by sequentially updating there first and second order moments (mean and covariance) and that the specific form of the estimator be linear, i.e. of the form

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \tilde{\mathbf{y}}_k , \quad (\text{A.1})$$

where $\hat{\mathbf{x}}_k^-$ is the prediction of the state at time k , \mathbf{K}_k is a linear gain term (the Kalman gain) and ν_k is the innovation signal at time k . The innovation signal is defined as the error between the observed noisy observation \mathbf{y}_k and its prediction $\hat{\mathbf{y}}_k^-$, i.e.,

$$\tilde{\mathbf{y}}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k^- . \quad (\text{A.2})$$

The predictions of the state, $\hat{\mathbf{x}}_k^-$, and the observation, $\hat{\mathbf{y}}_k$, are given by,

$$\hat{\mathbf{x}}_k^- = E[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k)] \quad (\text{A.3})$$

$$\hat{\mathbf{y}}_k^- = E[\mathbf{h}(\mathbf{x}_k^-, \mathbf{n}_k)] \quad (\text{A.4})$$

where the expectation are taken over the joint distributions of $\hat{\mathbf{x}}_{k-1}$ and \mathbf{v}_k , and $\hat{\mathbf{x}}_k^-$ and \mathbf{n}_k respectively.

Lets define the estimation error $\tilde{\mathbf{x}}_k$ as

$$\tilde{\mathbf{x}}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k^- . \quad (\text{A.5})$$

Substituting (A.1) into (A.5), we can formulate the estimation error as

$$\begin{aligned} \tilde{\mathbf{x}}_k &= \mathbf{x}_k - \hat{\mathbf{x}}_k^- - \mathbf{K}_k \tilde{\mathbf{y}}_k \\ &= \tilde{\mathbf{x}}_k^- - \mathbf{K}_k \tilde{\mathbf{y}}_k \end{aligned} \quad (\text{A.6})$$

using the fact that $E[\tilde{\mathbf{y}}_k] = 0$ under the assumption of an unbiased estimator. Taking outer products and expectations, the covariance of the update (Equation A.1) is given by

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k^-} - \mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_k} \mathbf{K}_k^T - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k \mathbf{x}_k} + \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} \mathbf{K}_k^T , \quad (\text{A.7})$$

where $\mathbf{P}_{\mathbf{x}_k}$ (the error covariance) is defined as

$$\mathbf{P}_{\mathbf{x}_k} \doteq E[\tilde{\mathbf{x}}_k \tilde{\mathbf{x}}_k^T] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] , \quad (\text{A.8})$$

and $\mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_k}$ (the cross covariance between the state and observation errors) as

$$\mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_k} \doteq E[\tilde{\mathbf{x}}_k^- \tilde{\mathbf{y}}_k^T] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{y}_k - \hat{\mathbf{y}}_k^-)^T] . \quad (\text{A.9})$$

In order to determine the MMSE optimal (Kalman) gain, we need to minimize the trace of the error covariance. This is done by setting the partial derivative of the trace of Equation A.7 with respect to the Kalman gain, \mathbf{K}_k , equal to zero. Using the following

linear algebra identities,

$$\frac{\partial}{\partial \mathbf{A}} (\text{trace}[\mathbf{ABA}^T]) = 2\mathbf{AB} ,$$

(where \mathbf{B} is symmetric) and

$$\frac{\partial}{\partial \mathbf{A}} (\text{trace}[\mathbf{AC}^T]) = \frac{\partial}{\partial \mathbf{A}} (\text{trace}[\mathbf{CA}^T]) = \mathbf{C} ,$$

the optimal Kalman gain can be written as

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_k} (\mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k})^{-1} . \quad (\text{A.10})$$

Substituting Equations A.10 and A.2 back into Equations A.1 and A.7 gives us the well known Kalman equations:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \quad (\text{A.11})$$

$$\mathbf{P}_{\mathbf{x}_k}^- = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} \mathbf{K}_k^T . \quad (\text{A.12})$$

Another commonly used form of the covariance update is given by substituting Equation A.10 into A.12, *i.e.*

$$\begin{aligned} \mathbf{P}_{\mathbf{x}_k}^- &= \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K} \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} (\mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k})^{-1} \mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_k}^T \\ &= \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_k}^T \end{aligned} \quad (\text{A.13})$$

Under Kalman's original assumptions (as per the derivation above) the Kalman filter will be the best *linear* estimator of the underlying state given all the observations, but not necessarily the globally optimal (in the Cramer-Rao sense) estimator. If the DSSM equations \mathbf{f} and \mathbf{h} are nonlinear and/or the state RVs are non-Gaussian, the optimal estimator will in general be a nonlinear function of the prior RVs and all of the observations (See Section 1.4). *However*, if the RVs are Gaussian and \mathbf{f} and \mathbf{h} are linear, then the expectations in Equations A.3, A.4, A.8 and A.9 can be calculated exactly in closed form¹. Under these conditions the Kalman filter will not only be the best linear estimator, it

¹The trick here is that the result of affine operations (such as expectations) on Gaussian random variables are themselves Gaussian.

will also be globally optimal for most error formulations. Furthermore it can be shown (asymptotically) that the inverse of the state error covariance \mathbf{P}_k is equal to the expected *Fisher Information Matrix* [104], implying that the linear Kalman filter is also *efficient* in the Cramer-Rao lower bound (CRLB) sense for linear systems and Gaussian RVs.

For the linear DSSM case, we can rewrite the Kalman update equations (A.11 and A.13) as,

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-) \quad (\text{A.14})$$

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \quad (\text{A.15})$$

$$= (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \mathbf{P}_{\mathbf{x}_k}^-, \quad (\text{A.16})$$

where we made use of the fact that for a linear DSSM (See Equations A.55 and A.56)

$$\hat{\mathbf{y}}_k^- = E[\mathbf{y}_k] = \mathbf{C}\hat{\mathbf{x}}_k^- \quad (\text{A.17})$$

and

$$\begin{aligned} \mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_k} &= E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{y}_k - \hat{\mathbf{y}}_k^-)^T] \\ &= E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{C}\mathbf{x}_k + \mathbf{n}_k - \mathbf{C}\hat{\mathbf{x}}_k^-)^T] \\ &= E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T] \mathbf{C}^T + E[\mathbf{x}_k \mathbf{n}_k^T] - E[\hat{\mathbf{x}}_k \mathbf{n}_k^T] \\ &= \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T. \end{aligned} \quad (\text{A.18})$$

Here we made the usual assumption that the observation noise is uncorrelated with the state and the state estimate. Under the same reasoning we can write the innovation covariance as,

$$\begin{aligned} \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k} &= E[(\mathbf{y}_k - \hat{\mathbf{y}}_k^-)(\mathbf{y}_k - \hat{\mathbf{y}}_k^-)^T] \\ &= E[(\mathbf{C}\mathbf{x}_k + \mathbf{n}_k - \mathbf{C}\hat{\mathbf{x}}_k^-)(\mathbf{C}\mathbf{x}_k + \mathbf{n}_k - \mathbf{C}\hat{\mathbf{x}}_k^-)^T] \\ &= \mathbf{C}E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T] \mathbf{C}^T + E[\mathbf{n}_k \mathbf{n}_k^T] \\ &\quad + \mathbf{C}E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)\mathbf{n}_k^T] + E[\mathbf{n}_k(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T] \mathbf{C}^T \\ &= \mathbf{C}\mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R}. \end{aligned} \quad (\text{A.19})$$

Note, we assume that the $\mathbf{D} = \mathbf{I}$ (the identity matrix) in Equation A.56 to simplify the notation. This does not affect the validity of the derivation, which can easily (but more verbosely) be extended to the more general case of a non identity matrix value for \mathbf{D} .

A.3 Gaussian MAP Derivation of Kalman Filter

The Kalman filter can also be derived from a Bayesian *maximum a posteriori* (MAP) perspective. Although this is a more constrained² framework than Kalman's original MMSE framework, it allows for very insightful analysis of Gaussian approximate recursive Bayesian estimation, especially when applied to parameter estimation.

Unlike the MMSE derivation in Section A.2, here we do not assume any prior explicit form of the optimal estimator. We do however, assume that prior state estimate (before the new observation is received) and the noisy observation of the hidden state are distributed according to a Gaussian probability density functions, *i.e.*

$$p(\mathbf{x}_k^-) = \frac{1}{\sqrt{2\pi |\mathbf{P}_{\mathbf{x}_k^-}|}} \exp \left[-(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) (\mathbf{P}_{\mathbf{x}_k^-})^{-1} (\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T \right], \quad (\text{A.20})$$

and

$$p(\mathbf{y}_k | \mathbf{x}_k) = \frac{1}{\sqrt{2\pi |\mathbf{P}_{\mathbf{x}_k^-}|}} \exp \left[-(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k)) \mathbf{R}^{-1} (\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k))^T \right], \quad (\text{A.21})$$

where $\mathbf{P}_{\mathbf{x}_k^-}$ is the prior state estimate error covariance and \mathbf{R} is the observation (measurement) noise covariance, *i.e.*

$$\mathbf{R} = E [(\mathbf{n}_k - \bar{\mathbf{n}})(\mathbf{n}_k - \bar{\mathbf{n}})^T]. \quad (\text{A.22})$$

We further assume that the noise is zero-mean, *i.e.* $\bar{\mathbf{n}} = \mathbf{0}$, such that

$$\mathbf{R} = E [(\mathbf{n}_k - \bar{\mathbf{n}})(\mathbf{n}_k - \bar{\mathbf{n}})^T] = E [\mathbf{n}_k \mathbf{n}_k^T].$$

In order to calculate the MAP estimate, we first need to factor the posterior distribution

²Here an explicit form for the true distribution of the prior and posterior random variables are assumed, viz.. multi-variate Gaussian.

of the state using Bayes Rule,

$$\begin{aligned} p(\mathbf{x}_k | \mathbf{y}_{1:k}) &= \frac{p(\mathbf{y}_{1:k} | \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{y}_{1:k})} \\ &= \frac{p(\mathbf{y}_k, \mathbf{y}_{1:k-1} | \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{y}_k, \mathbf{y}_{1:k-1})} \\ &= \frac{p(\mathbf{y}_k | \mathbf{y}_{1:k-1}, \mathbf{x}_k) p(\mathbf{y}_{1:k-1} | \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{y}_k | \mathbf{y}_{1:k-1}) p(\mathbf{y}_{1:k-1})} \end{aligned} \quad (\text{A.23})$$

$$= \frac{p(\mathbf{y}_k | \mathbf{y}_{1:k-1}, \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{y}_{1:k-1}) p(\mathbf{y}_{1:k-1}) p(\mathbf{x}_k)}{p(\mathbf{y}_k | \mathbf{y}_{1:k-1}) p(\mathbf{y}_{1:k-1}) p(\mathbf{x}_k)} \quad (\text{A.24})$$

$$= \frac{p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k | \mathbf{y}_{1:k-1})} , \quad (\text{A.25})$$

where we again made use of Bayes rule going from Equation A.23 to A.24, and used the fact that the observation sequence is conditionally independent given the current state in reducing $p(\mathbf{y}_k | \mathbf{y}_{1:k-1}, \mathbf{x}_k)$ to $p(\mathbf{y}_k | \mathbf{x}_k)$. Now, by definition $p(\mathbf{x}_k | \mathbf{y}_{1:k-1}) = p(\mathbf{x}_k^-)$, so Equation A.25 becomes,

$$p(\mathbf{x}_k | \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k^-)}{p(\mathbf{y}_k | \mathbf{y}_{1:k-1})} . \quad (\text{A.26})$$

The MAP estimate $\hat{\mathbf{x}}_k$ is found by picking the \mathbf{x}_k that maximizes Equation A.26. This is equivalent to choosing the \mathbf{x}_k that *minimizes* the negative of the logarithm of the numerator, since the denominator is not a function of \mathbf{x}_k . So

$$\hat{\mathbf{x}}_k = \operatorname{argmin} \{ \mathbf{x}_k; -\ln [p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k^-)] \} \quad (\text{A.27})$$

$$= \operatorname{argmin} \{ \mathbf{x}_k; -\ln (p(\mathbf{y}_k | \mathbf{x}_k)) - \ln (p(\mathbf{x}_k^-)) \} . \quad (\text{A.28})$$

Substituting Equations A.20 and A.21 into A.28, and setting the derivative with respect to \mathbf{x}_k equal to zero, we find the maximum a posteriori equation which must be solved for $\hat{\mathbf{x}}_k$,

$$\frac{\partial}{\partial \mathbf{x}_k} \left[(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k)) \mathbf{R}^{-1} (\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k))^T \right] + \frac{\partial}{\partial \mathbf{x}_k} \left[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) (\mathbf{P}_{\mathbf{x}_k}^-)^{-1} (\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T \right] = 0 .$$

Carrying out the differentiation results in,

$$-\frac{\partial}{\partial \mathbf{x}_k} [\mathbf{h}(\mathbf{x}_k)]^T \mathbf{R}^{-1} (\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k)) + (\mathbf{P}_{\mathbf{x}_k}^-)^{-1} (\mathbf{x}_k - \hat{\mathbf{x}}_k^-) = 0 . \quad (\text{A.29})$$

If we assume the DSSM is linear, the system observation equation can be written as

$$\mathbf{h}(\mathbf{x}_k) = \mathbf{C}\mathbf{x}_k . \quad (\text{A.30})$$

Substituting Equations A.30 into Equation A.29, differentiating and rearranging terms gives

$$\begin{aligned} (\mathbf{P}_{\mathbf{x}_k}^-)^{-1} (\mathbf{x}_k - \hat{\mathbf{x}}_k^-) &= \frac{\partial}{\partial \mathbf{x}_k} [\mathbf{C}\mathbf{x}_k]^T \mathbf{R}^{-1} (\mathbf{y}_k - \mathbf{C}\mathbf{x}_k) \\ (\mathbf{P}_{\mathbf{x}_k}^-)^{-1} (\mathbf{x}_k - \hat{\mathbf{x}}_k^-) &= \mathbf{C}^T \mathbf{R}^{-1} (\mathbf{y}_k - \mathbf{C}\mathbf{x}_k + \mathbf{C}\hat{\mathbf{x}}_k^- - \mathbf{C}\hat{\mathbf{x}}_k^-) \\ (\mathbf{P}_{\mathbf{x}_k}^-)^{-1} (\mathbf{x}_k - \hat{\mathbf{x}}_k^-) &= \mathbf{C}^T \mathbf{R}^{-1} [\mathbf{y}_k - \mathbf{C}(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) - \mathbf{C}\hat{\mathbf{x}}_k^-] \\ [(\mathbf{P}_{\mathbf{x}_k}^-)^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C}] (\mathbf{x}_k - \hat{\mathbf{x}}_k^-) &= \mathbf{C}^T \mathbf{R}^{-1} (\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-) \end{aligned} \quad (\text{A.31})$$

Solving this equation for \mathbf{x}_k gives us the MAP optimal state estimate, $\hat{\mathbf{x}}_k$. Since $\mathbf{P}_{\mathbf{x}_k}^-$ and \mathbf{R} are both square-symmetric and positive definite, $[(\mathbf{P}_{\mathbf{x}_k}^-)^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C}]$ is invertible. This allows us to rewrite the solution to Equation A.31 as

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + [(\mathbf{P}_{\mathbf{x}_k}^-)^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C}]^{-1} \mathbf{C}^T \mathbf{R}^{-1} (\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-) . \quad (\text{A.32})$$

If we define

$$\mathbf{K}_k \doteq [(\mathbf{P}_{\mathbf{x}_k}^-)^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C}]^{-1} \mathbf{C}^T \mathbf{R}^{-1} = \tilde{\mathbf{P}} \mathbf{C}^T \mathbf{R}^{-1} \quad (\text{A.33})$$

where $\tilde{\mathbf{P}}$ in turn is defined as

$$\tilde{\mathbf{P}} \doteq [(\mathbf{P}_{\mathbf{x}_k}^-)^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C}]^{-1} , \quad (\text{A.34})$$

we can write Equation A.32 as

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-) , \quad (\text{A.35})$$

which is the MAP derived Kalman state update.

Given this formulation of the state update, we can next determine the MAP derived

covariance update, i.e how $\mathbf{P}_{\mathbf{x}_k}$, the covariance of $\mathbf{x}_k - \hat{\mathbf{x}}_k$, is calculated. Using Equation A.35, we can write the covariance as

$$\begin{aligned}
\mathbf{P}_{\mathbf{x}_k} &= \text{cov}\{\mathbf{x}_k - \hat{\mathbf{x}}_k\} \\
&= E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] \\
&= E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^- - \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-))(\mathbf{x}_k - \hat{\mathbf{x}}_k^- - \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-))^T] \\
&= E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T] - E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-)^T]\mathbf{K}_k^T \\
&\quad - \mathbf{K}_k E[(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T] + \mathbf{K}_k E[(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-)(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-)^T]\mathbf{K}_k^T \\
&= \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T \mathbf{K}_k^T - \mathbf{K}_k \mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- + \mathbf{K}_k (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R}) \mathbf{K}_k^T, \tag{A.36}
\end{aligned}$$

where we made use of the fact that $\text{cov}\{\mathbf{x}_k - \hat{\mathbf{x}}_k^-\} = \mathbf{P}_{\mathbf{x}_k}^-$ and that the observation noise \mathbf{n}_k is uncorrelated with the state \mathbf{x}_k . If we now (using Equation A.33) rewrite \mathbf{K}_k as

$$\begin{aligned}
\mathbf{K}_k &= \tilde{\mathbf{P}} \mathbf{C}^T \mathbf{R}^{-1} \\
&= \tilde{\mathbf{P}} \mathbf{C}^T \mathbf{R}^{-1} (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R}) (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R})^{-1} \\
&= \tilde{\mathbf{P}} (\mathbf{C}^T \mathbf{R}^{-1} \mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{R}) (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R})^{-1} \\
&= \tilde{\mathbf{P}} \left(\mathbf{C}^T \mathbf{R}^{-1} \mathbf{C} + (\mathbf{P}_{\mathbf{x}_k}^-)^{-1} \right) \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R})^{-1} \\
&= \tilde{\mathbf{P}} \tilde{\mathbf{P}}^{-1} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R})^{-1} \\
&= \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R})^{-1} \tag{A.37}
\end{aligned}$$

and substitute into Equation A.36, we get (after rearrangement and cancellation of terms),

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R})^{-1} \mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \tag{A.38}$$

$$= \left[(\mathbf{P}_{\mathbf{x}_k}^-)^{-1} + \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C} \right]^{-1} \tag{A.39}$$

$$= \tilde{\mathbf{P}} \tag{A.40}$$

where we made use of the *matrix inversion lemma* [156] in going from Equation A.38 to A.39, and the definition of $\tilde{\mathbf{P}}$ (Equation A.34) to go from Equation A.39 to A.40.

In other words, $\tilde{\mathbf{P}}$, as defined in Equation A.34, is the covariance of the new state

estimate which can be recursively calculated by

$$\mathbf{P}_{\mathbf{x}_k} = \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{K}_k \mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \quad (\text{A.41})$$

$$= (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \mathbf{P}_{\mathbf{x}_k}^- . \quad (\text{A.42})$$

Substituting Equations A.19 and A.2 into Equation A.37, we can rewrite the gain term as,

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R})^{-1} \\ &= \mathbf{P}_{\mathbf{x}_k \tilde{\mathbf{y}}_k} \mathbf{P}_{\tilde{\mathbf{y}}_k \tilde{\mathbf{y}}_k}^{-1}, \end{aligned} \quad (\text{A.43})$$

which is identical to the form of the Kalman gain term derived under the MMSE criterion in Section A.2. This result, together with the exact same recursive forms of the state estimate update (Equations A.14 and A.35) and covariance update (Equations A.16 and A.42) proofs that the linear Kalman filter is not only the best linear *minimum-mean square-error* estimator of the hidden state, but also that it calculates the *maximum a-posteriori* estimate under a linear Gaussian assumption.

A.4 The Full Kalman Filter Algorithm

The only remaining part to derive in order to implement the full recursive Kalman filter algorithm, is the *time-update* step that is used to project the best state estimate at the previous time instance, $\hat{\mathbf{x}}_{k-1}$, forward in time in order to generate the next prior estimate, $\hat{\mathbf{x}}_k^-$. This prior is then updated with the new information contained in the latest observation, \mathbf{y}_k , during the *measurement-update*. It is this measurement-update that was derived (using two different approached) in Sections A.2 and A.3.

The time update of the prior state estimate and its covariance is found by taking the expected value of the linear process model (Equation A.55), *i.e.*

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= E[\mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{G}\mathbf{v}_k] \\ &= \mathbf{A}E[\mathbf{x}_{k-1}] + \mathbf{B}\mathbf{u}_k + \mathbf{G}E[\mathbf{v}_k] \\ &= \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{G}\bar{\mathbf{v}}, \end{aligned} \quad (\text{A.44})$$

and

$$\begin{aligned}
\mathbf{P}_{\mathbf{x}_k}^- &= E \left[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) (\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T \right] \\
&= E \left[(\mathbf{A}\mathbf{x}_{k-1} + \mathbf{G}\mathbf{v}_k - \mathbf{A}\hat{\mathbf{x}}_{k-1} - \mathbf{G}\bar{\mathbf{v}}) (\mathbf{A}\mathbf{x}_{k-1} + \mathbf{G}\mathbf{v}_k - \mathbf{A}\hat{\mathbf{x}}_{k-1} - \mathbf{G}\bar{\mathbf{v}})^T \right] \\
&= \mathbf{A}E \left[(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1})^T \right] \mathbf{A}^T \\
&\quad + \mathbf{G}E \left[(\mathbf{v}_k - \bar{\mathbf{v}}) (\mathbf{v}_k - \bar{\mathbf{v}})^T \right] \mathbf{G}^T \\
&= \mathbf{A}\mathbf{P}_{\mathbf{x}_{k-1}}\mathbf{A}^T + \mathbf{G}\mathbf{R}_{\mathbf{v}}\mathbf{G}^T,
\end{aligned} \tag{A.45}$$

where we made the usual assumption that the process noise, \mathbf{v}_k , is uncorrelated with the state and the state estimate. Combining this time-update and the measurement-update derived in the previous section, together into a *recursive predictor-corrector* [116, 62, 104] results in the well-known *linear Kalman filter* presented in Algorithm 19. Note that although the Kalman filter algorithm was derived using constant system matrices, \mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D} and \mathbf{G} , they are allowed to changed over time without affecting the validity of the derivations presented. For this reason Algorithm 19 use them in the more general (possibly) time-varying form, i.e. \mathbf{A}_k , \mathbf{B}_k , \mathbf{C}_k , \mathbf{D}_k and \mathbf{G}_k .

A.5 Alternative Forms

A.5.1 Joseph's Form of the Measurement Update

The standard form of the Kalman filter covariance update given by Equation A.54 involves subtraction of one matrix from another which can result in loss of symmetry and thus positive definiteness of the resulting updated covariance matrix due to numerical rounding errors. A numerically alternative (but algebraically equivalent) form of the covariance measurement-update can be found by manipulating Equation A.36 into the following numerically stable form:

$$\begin{aligned}
\mathbf{P}_{\mathbf{x}_k} &= \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T \mathbf{K}_k^T - \mathbf{K}_k \mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- + \mathbf{K}_k (\mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T + \mathbf{R}) \mathbf{K}_k^T \\
&= \mathbf{P}_{\mathbf{x}_k}^- - \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T \mathbf{K}_k^T - \mathbf{K}_k \mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- + \mathbf{K}_k \mathbf{C} \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}^T \mathbf{K}_k^T + \mathbf{K}_k \mathbf{R} \mathbf{K}_k^T \\
&= [\mathbf{I} - \mathbf{K}_k \mathbf{C}] \mathbf{P}_{\mathbf{x}_k}^- [\mathbf{I} - \mathbf{K}_k \mathbf{C}]^T + \mathbf{K}_k \mathbf{R} \mathbf{K}_k^T.
\end{aligned} \tag{A.57}$$

Algorithm 19 Algorithm for the Kalman filter (KF).

Initialization:

$$\hat{\mathbf{x}}_0 = E[\mathbf{x}_0] \quad (\text{A.46})$$

$$\mathbf{P}_{\mathbf{x}_0} = E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T] \quad (\text{A.47})$$

$$\mathbf{R}_{\mathbf{v}} = E[(\mathbf{v} - \bar{\mathbf{v}})(\mathbf{v} - \bar{\mathbf{v}})^T] \quad (\text{A.48})$$

$$\mathbf{R}_{\mathbf{n}} = E[(\mathbf{n} - \bar{\mathbf{n}})(\mathbf{n} - \bar{\mathbf{n}})^T] \quad (\text{A.49})$$

For $k = 1, 2, \dots, \infty$:

1. *Prediction step*:

- Compute the predicted state mean and covariance (*time-update*)

$$\hat{\mathbf{x}}_k^- = \mathbf{A}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{G}_k \bar{\mathbf{v}} \quad (\text{A.50})$$

$$\mathbf{P}_{\mathbf{x}_k}^- = \mathbf{A}_k \mathbf{P}_{\mathbf{x}_{k-1}} \mathbf{A}_k^T + \mathbf{G}_k \mathbf{R}_{\mathbf{v}} \mathbf{G}_k^T \quad (\text{A.51})$$

2. *Correction step*:

- Update estimates with latest observation (*measurement update*)

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k}^- \mathbf{A}_k^T (\mathbf{C}_k \mathbf{P}_{\mathbf{x}_k}^- \mathbf{C}_k^T + \mathbf{D}_k \mathbf{R}_{\mathbf{n}} \mathbf{D}_k^T)^{-1} \quad (\text{A.52})$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k [\mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_k^- - \mathbf{D}_k \bar{\mathbf{n}}] \quad (\text{A.53})$$

$$\mathbf{P}_{\mathbf{x}_k} = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \mathbf{P}_{\mathbf{x}_k}^- . \quad (\text{A.54})$$

The linear DSSM is given by:

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{G}_k \mathbf{v}_k \quad (\text{A.55})$$

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \mathbf{D}_k \mathbf{n}_k \quad (\text{A.56})$$

Equation A.57 is known as *Joseph's form covariance measurement update* [62, 25]. This form is numerically stable since the “subtraction term”, $\mathbf{I} - \mathbf{K}_k \mathbf{C}$, gets “squared” which preserves the symmetry and ensures positive definiteness of the updated covariance.

A.5.2 Schmidt-Kalman Filter

The traditional purpose of the *Schmidt-Kalman filter* (SKF) [172] is to reduce the computational complexity of a standard Kalman filter by eliminating states that are of no physical interest, but required to estimate other quantities such as noises and/or biases. The formulation of the Schmidt-Kalman filter begins by partitioning the state vector \mathbf{x}_k into two components,

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}^a \\ \mathbf{x}^b \end{bmatrix}_k , \quad (\text{A.58})$$

where \mathbf{x}_k^a are the states of “interest” and \mathbf{x}_k^b are the remaining states. The state covariance matrix in turn is partitioned into four quadrants, i.e.,

$$\mathbf{P}_{\mathbf{x}_k} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}^a} & \mathbf{P}_{\mathbf{x}^a \mathbf{x}^b} \\ \mathbf{P}_{\mathbf{x}^b \mathbf{x}^a} & \mathbf{P}_{\mathbf{x}^b} \end{bmatrix}_k . \quad (\text{A.59})$$

The state propagation (process) model³ and observation models are partitioned accordingly:

$$\begin{bmatrix} \mathbf{x}^a \\ \mathbf{x}^b \end{bmatrix}_{k+1} = \begin{bmatrix} \mathbf{A}^a & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^b \end{bmatrix}_k \begin{bmatrix} \mathbf{x}^a \\ \mathbf{x}^b \end{bmatrix}_k + \begin{bmatrix} \mathbf{v}^a \\ \mathbf{v}^b \end{bmatrix}_k \quad (\text{A.60})$$

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{C}^a & \mathbf{C}^b \end{bmatrix} \begin{bmatrix} \mathbf{x}^a \\ \mathbf{x}^b \end{bmatrix}_k + \mathbf{n}_k . \quad (\text{A.61})$$

After applying the partitioning of Equations A.58, A.59, A.60 and A.61 to the general Kalman filter equations (Algorithm 19), solving for each block and *setting the Kalman gain for the \mathbf{x}_k^b states to zero*, the following set of equations result [172]:

³For this derivation we assume the noise sensitivity matrices \mathbf{G}_k and \mathbf{D}_k are both equal to the identity matrix. This simplifies the notation of the derivation, but does not affect its generality or validity.

Schmidt-Kalman time update:

$$\hat{\mathbf{x}}_k^{a-} = \mathbf{A}_{k-1}^a \hat{\mathbf{x}}_{k-1}^a \quad (\text{A.62})$$

$$\mathbf{P}_{\mathbf{x}_k^a}^- = \mathbf{A}_{k-1}^a \mathbf{P}_{\mathbf{x}_{k-1}^a} \left(\mathbf{A}_{k-1}^a \right)^T + \mathbf{R}_{\mathbf{v}^a} \quad (\text{A.63})$$

$$\mathbf{P}_{\mathbf{x}_k^a \mathbf{x}_k^b}^- = \mathbf{A}_{k-1}^a \mathbf{P}_{\mathbf{x}_{k-1}^a \mathbf{x}_{k-1}^b} \left(\mathbf{A}_{k-1}^b \right)^T \quad (\text{A.64})$$

$$\mathbf{P}_{\mathbf{x}_k^b \mathbf{x}_k^a}^- = \left(\mathbf{P}_{\mathbf{x}_k^a \mathbf{x}_k^b}^- \right)^T \quad (\text{A.65})$$

$$\mathbf{P}_{\mathbf{x}_k^b}^- = \mathbf{A}_{k-1}^b \mathbf{P}_{\mathbf{x}_{k-1}^b} \left(\mathbf{A}_{k-1}^b \right)^T + \mathbf{R}_{\mathbf{v}^b} \quad (\text{A.66})$$

Schmidt-Kalman measurement update:

$$\begin{aligned} \mathbf{O}_k &= \mathbf{C}_k^a \mathbf{P}_{\mathbf{x}_k^a}^- \left(\mathbf{C}_k^a \right)^T + \mathbf{C}_k^a \mathbf{P}_{\mathbf{x}_k^a \mathbf{x}_k^b}^- \left(\mathbf{C}_k^b \right)^T + \mathbf{C}_k^b \mathbf{P}_{\mathbf{x}_k^b \mathbf{x}_k^a}^- \left(\mathbf{C}_k^a \right)^T + \\ &\quad + \mathbf{C}_k^b \mathbf{P}_{\mathbf{x}_k^b}^- \left(\mathbf{C}_k^b \right)^T + \mathbf{R}_{\mathbf{n}} \end{aligned} \quad (\text{A.67})$$

$$\mathbf{K}_k = \left[\mathbf{P}_{\mathbf{x}_k^a}^- \left(\mathbf{C}_k^a \right)^T + \mathbf{P}_{\mathbf{x}_k^a \mathbf{x}_k^b}^- \left(\mathbf{C}_k^b \right)^T \right] \mathbf{O}_k^{-1} \quad (\text{A.68})$$

$$\hat{\mathbf{x}}_k^a = \hat{\mathbf{x}}_k^{a-} + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{C}_k^a \hat{\mathbf{x}}_k^{a-} - \mathbf{C}_k^b \hat{\mathbf{x}}_0^{a-} \right) \quad (\text{A.69})$$

$$\mathbf{P}_{\mathbf{x}_k^a} = \left(\mathbf{I} - \mathbf{K}_k \mathbf{C}_k^a \right) \mathbf{P}_{\mathbf{x}_k^a}^- - \mathbf{K}_k \mathbf{C}_k^b \mathbf{P}_{\mathbf{x}_k^b \mathbf{x}_k^a}^- \quad (\text{A.70})$$

$$\mathbf{P}_{\mathbf{x}_k^a \mathbf{x}_k^b} = \left(\mathbf{I} - \mathbf{K}_k \mathbf{C}_k^a \right) \mathbf{P}_{\mathbf{x}_k^a \mathbf{x}_k^b}^- - \mathbf{K}_k \mathbf{C}_k^b \mathbf{P}_{\mathbf{x}_k^b}^- \quad (\text{A.71})$$

$$\mathbf{P}_{\mathbf{x}_k^b \mathbf{x}_k^a} = \left(\mathbf{P}_{\mathbf{x}_k^a \mathbf{x}_k^b}^- \right)^T \quad (\text{A.72})$$

$$\mathbf{P}_{\mathbf{x}_k^b} = \mathbf{P}_{\mathbf{x}_k^b}^- \quad (\text{A.73})$$

The Schmidt-Kalman filter algorithm may appear at first glance more complicated than the traditional Kalman filter; however, substantial computational savings are gained through the fact that the filter only solves for the \mathbf{x}^a states and not for the \mathbf{x}^b states which, in typical applications of this technique, is of no physical interest [51].

Appendix B

Navigation Primer

B.1 Introduction

This appendix introduces certain navigation theory related concepts which are used in Chapter 5.

B.2 Reference Frames

This section presents four geographical reference frames (see Figures B.1 and B.2) that are used to describe the position, orientation, and velocities (translational and rotational) of a free moving vehicle. These four reference frames are [17]:

ECI: The *Earth-Centered Inertial* frame is by definition an inertial frame, i.e., a non-accelerating reference frame in which Newton's laws of motion apply. The origin of the ECI coordinate space is located at the center of the earth. The axes of this frame is fixed in space, i.e. it does not rotate with the earth.

ECEF: The *Earth-Centered Earth-Fixed* frame has the same origin as the ECI frame, i.e. at the center of the earth, but its axes are fixed to the earth and hence rotate along with it, relative to the inertial frame (ECI). The rotation rate of the axes of the ECEF frame is the same as that of the earth, i.e. $\omega_e = 7.291 \times 10^{-5}$ rad/s. If a vehicle (at or close to the surface of the earth) moves relatively slowly compared to the tangential component of the earth's rotation at the earth's surface (in inertial frame), the rotation of the ECEF frame can be neglected and hence the ECEF frame can be considered inertial.

NED: The *North-East-Down* coordinate system is defined relative to the Earth's reference ellipsoid and is the coordinate system commonly referred to during everyday life. It

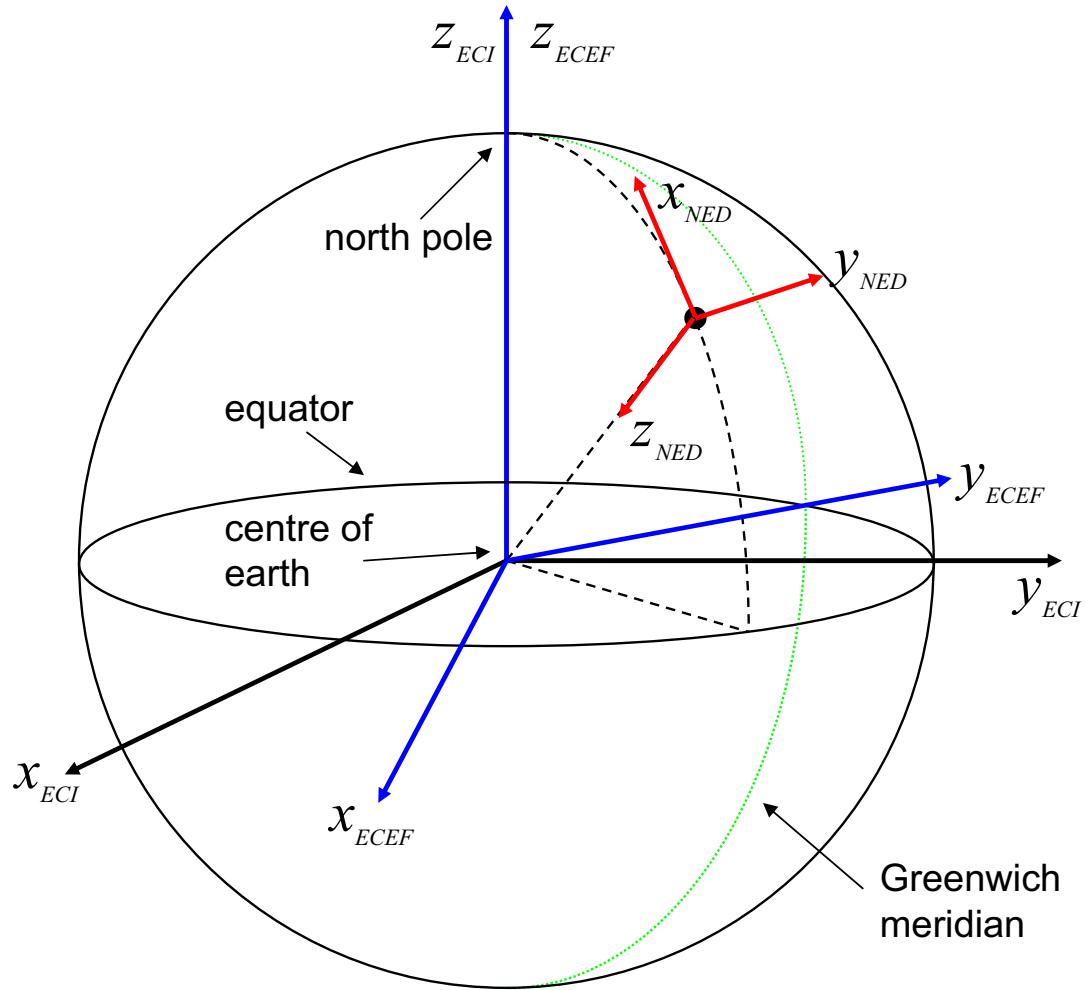


Figure B.1: Reference frames used for terrestrial navigation. The black axes are the ECI (Earth-Centered Inertial) reference frame. The blue axes (sharing a common z-axis with the ECI frame) is the ECEF (Earth-Centered Earth-Fixed frame). Both the ECI and ECEF frames have origins located at the center of the earth, but the ECEF frame is “fixed” in that it rotates with the earth. The ECI frame on the other hand is a true inertial frame in with the x-axis pointing towards the vernal equinox, the z-axis lying along the earth’s rotational axis (perpendicular to equatorial plane). The red axes are the NED (north-east-down) reference frame that is fixed to the tangential plane (at any point) of the earth-surface-ellipsoid. The x-axis of this frame points north, the y-axis points east and the z-axis points towards the center of the earth.

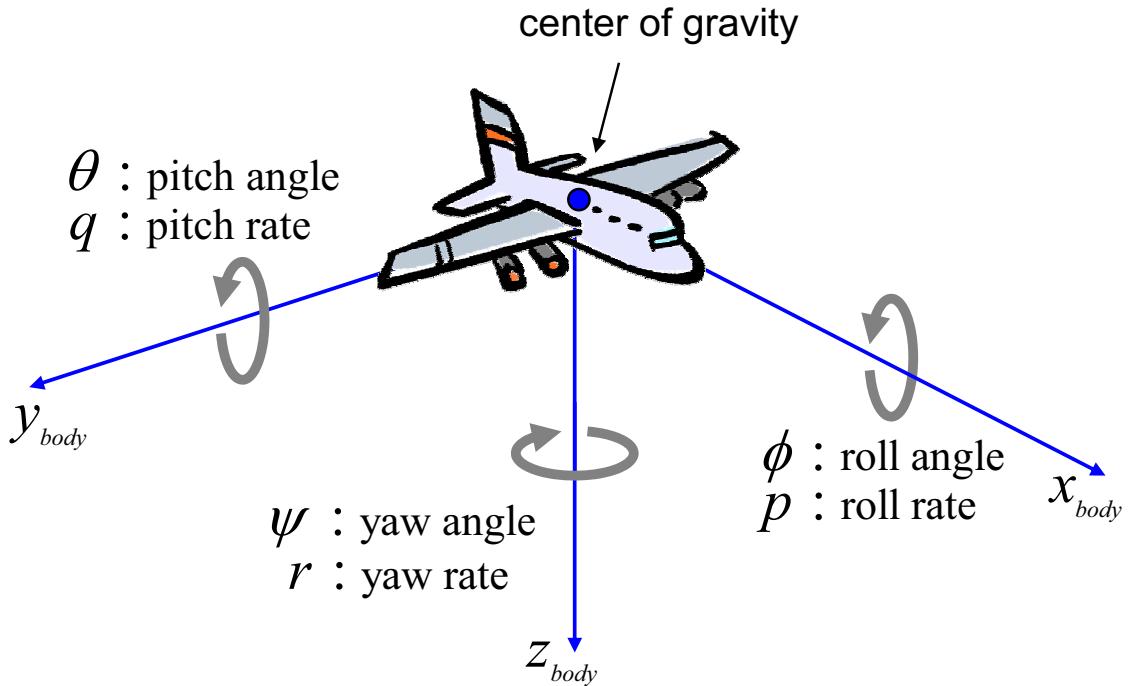


Figure B.2: Vehicle body fixed reference frame: The origin of this reference frame lies at the center of gravity of the vehicle. The x -axis points out the nose of the vehicle (along longitudinal axis), the y -axis points out the right wing (lateral axis) and the z -axis (pointing out the bottom) completes the right-handed triad. The attitude rotational angles ϕ , θ and ψ are also indicated. ϕ is the roll angle around the x -axis, θ is the pitch angle around the y -axis, and ψ is the yaw (heading) angle around the z -axis. The derivatives of the body-axis attitude angles are indicated as well: p (roll rate), q (pitch rate) and r (yaw rate).

is usually defined as a fixed tangent plane on the surface of the earth with the x -axis pointing towards true north, the y -axis towards east and z -axis pointing downwards normal to the earth's surface. The origin of this reference plane is usually chosen to be located at the intersection of the equator and the Greenwich meridian, however local origins are also commonly used if the total distance covered by the vehicle is relatively small compared to the scale of the earth's surface.

Body: The body fixed reference frame is a moving coordinate system with its origin located at the center of gravity of the vehicle. For aircraft, the x -axis points along the longitudinal axis of the vehicle (out through the nose), the y -axis points along the lateral axis (out the right wing) and the z -axis points down through the bottom of the aircraft to complete the right-handed reference frame.

B.3 Attitude Quaternion

The attitude quaternion of a rotating object in 3D space is a convenient 4 dimensional vector representation of an arbitrary rotation around any axis going through the objects center of gravity [175]. One can think of the quaternion vector as 4-tuple

$$\mathbf{e} = \begin{bmatrix} e_0 & e_1 & e_2 & e_3 \end{bmatrix}^T , \quad (\text{B.1})$$

where last three components define an axis of rotation, \mathbf{n} , in 3-space (through the objects center of gravity) and the first component specifies the angle of the rotation, α , around that axis. The exact relationship is given by,

$$\mathbf{e} = \begin{bmatrix} \cos(\alpha/2) & \mathbf{n}^T \sin(\alpha/2) \end{bmatrix}^T , \quad (\text{B.2})$$

and is often referred to as the *Euler parameter* representation of the attitude quaternion [177]. Since such an arbitrary rotation can be broken down into three consecutive rotations around the three main Euler axes (pitch, roll and yaw), there is a one-to-one correspondence between the 4 dimensional attitude quaternion vector and the three dimensional Euler angle vector, i.e.,

$$\mathbf{e} \rightleftarrows \begin{bmatrix} \phi & \theta & \psi \end{bmatrix} . \quad (\text{B.3})$$

Figure B.3 demonstrates how an arbitrary rotation is specified using a quaternion representation. At first glance it might seem that the quaternion representation is under-determined (it seems to have an extra degree of freedom compared to the Euler angle representation). By enforcing a *unity norm constraint* on the quaternion though,

$$\|\mathbf{e}\| = \sqrt{e_0^2 + e_1^2 + e_2^2 + e_3^2} \equiv 1 , \quad (\text{B.4})$$

we can ensure that all possible quaternion solutions lie on a 3D unit hypersphere (sub-manifold) embedded in the 4D quaternion space, which allows for an invertible and unique attitude relationship between Euler angles and the corresponding quaternion vector. This unique transformation is defined by Equations B.5 - B.11 on the next page.

The reason why a quaternion representation of attitude (or an attitude change, i.e.

rotation) is preferable to an Euler angle representation, is that the angular discontinuities at $\pm\pi$ radians are avoided as well as the trigonometric singularities that can occur at $\pm\frac{\pi}{2}$ radians (i.e. the infamous “gimbal lock” problem [80, 91]).

Converting from a specific quaternion representation to the related Euler angles is done using the following equations,

$$\phi = \arctan \left(\frac{2(e_2 e_3 + e_0 e_1)}{1 - 2(e_1^2 + e_2^2)} \right) \quad (\text{B.5})$$

$$\theta = \arcsin(-2(e_1 e_3 - e_0 e_2)) \quad (\text{B.6})$$

$$\psi = \arctan \left(\frac{2(e_1 e_2 + e_0 e_3)}{1 - 2(e_2^2 + e_3^2)} \right) \quad (\text{B.7})$$

(note we make use here of the *four quadrant arctangent* function, i.e., $-\pi \leq \arctan(y/x) \leq \pi$). Converting from Euler angles to a quaternion vector is given by

$$\mathbf{e}^\psi = \begin{bmatrix} \cos\left(\frac{\psi}{2}\right) & 0 & 0 & \sin\left(\frac{\psi}{2}\right) \end{bmatrix} \quad (\text{B.8})$$

$$\mathbf{e}^\theta = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & 0 & \sin\left(\frac{\theta}{2}\right) & 0 \end{bmatrix} \quad (\text{B.9})$$

$$\mathbf{e}^\phi = \begin{bmatrix} \cos\left(\frac{\phi}{2}\right) & \sin\left(\frac{\phi}{2}\right) & 0 & 0 \end{bmatrix} \quad (\text{B.10})$$

$$\mathbf{e} = \begin{bmatrix} e_0 & e_1 & e_2 & e_3 \end{bmatrix} = \mathbf{e}^\phi \otimes (\mathbf{e}^\theta \otimes \mathbf{e}^\psi), \quad (\text{B.11})$$

where \otimes is the quaternion multiplication operator defined by

$$\begin{aligned} \mathbf{c} &= \mathbf{a} \otimes \mathbf{b} \\ &\doteq \begin{bmatrix} a_0 b_0 - a_1 b_1 - a_2 b_2 - a_3 b_3 \\ a_0 b_1 + a_1 b_0 + a_2 b_3 - a_3 b_2 \\ a_0 b_2 - a_1 b_3 + a_2 b_0 + a_3 b_1 \\ a_0 b_3 + a_1 b_2 - a_2 b_1 + a_3 b_0 \end{bmatrix}, \end{aligned} \quad (\text{B.12})$$

where \mathbf{c} is the resulting unity norm product quaternion.

Another convenient characteristic of the quaternion representation is that the *directional cosine matrix* (DCM) used to transform 3D vectors from the body-frame to the NED frame¹ (and vice-versa), can be easily and directly constructed from the components

¹We make the assumption here that the scale of our movement (translation and rotation) is such that

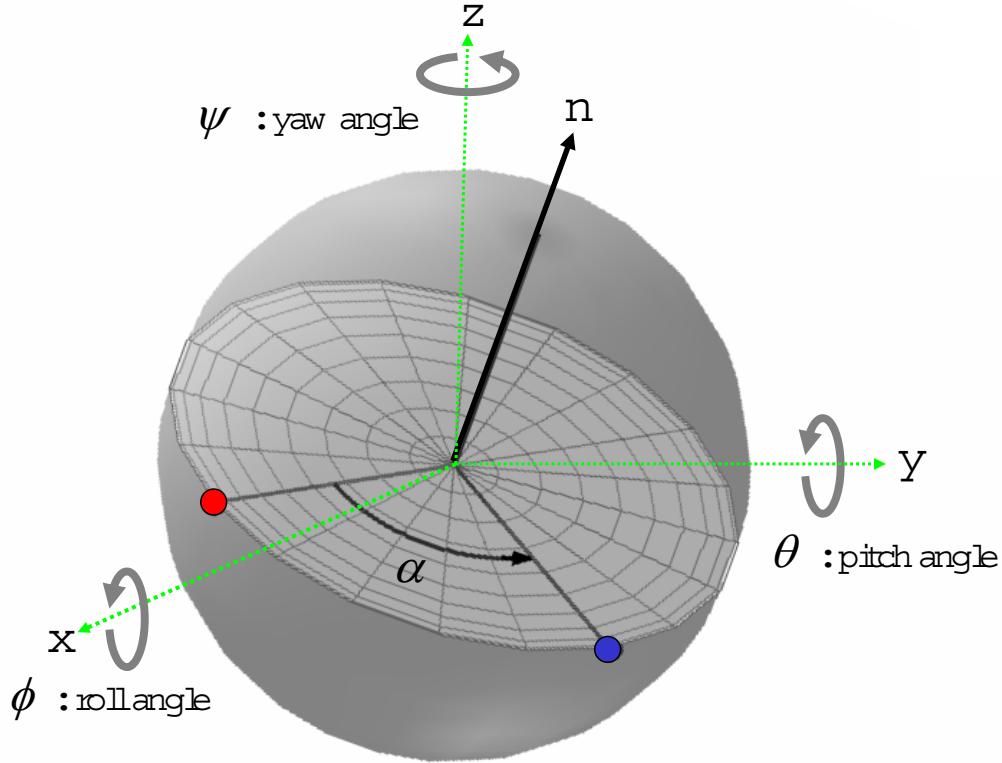


Figure B.3: Quaternion representation of an arbitrary rotation: Any point on the 3D sphere can be rotated to any other point using three consecutive Euler angle (ϕ, θ, ψ) rotations around the major ($x - y - z$) axes, or using a single rotation around an axis that is perpendicular to the plane containing the two points and the origin of the sphere. The first method requires specifying the three Euler angles. The second method (quaternion method), requires specifying the normal vector ($\mathbf{n} = [\sin(\alpha/2)]^{-1} [e_2 \ e_3 \ e_4]^T$) to the plane and the rotation angle ($\alpha = 2 \arccos(e_1)$) in the plane. In this example the red line is rotated to coincide with the blue line.

of the vehicle's attitude quaternion, i.e.,

$$\begin{aligned} \mathbf{T}_{b \rightarrow i}^{DCM} &= (\mathbf{T}_{i \rightarrow b}^{DCM})^T \\ &= 2 \begin{bmatrix} 0.5 - e_2^2 - e_3^2 & e_1 e_2 - e_0 e_3 & e_1 e_3 + e_0 e_2 \\ e_1 e_2 + e_0 e_3 & 0.5 - e_1^2 - e_3^2 & e_2 e_3 - e_0 e_1 \\ e_1 e_3 - e_0 e_2 & e_2 e_3 + e_0 e_1 & 0.5 - e_1^2 - e_2^2 \end{bmatrix}. \quad (B.13) \end{aligned}$$

the NED frame can be considered an inertial frame.

B.4 Exponentiation of 4×4 Skew-Symmetric Quaternion Update Matrix

Assume we have the following 4×4 skew-symmetric (anti-symmetric) matrix

$$\mathbf{A} = \begin{bmatrix} 0 & a_1 & a_2 & a_3 \\ -a_1 & 0 & -a_3 & a_2 \\ -a_2 & a_3 & 0 & -a_1 \\ -a_3 & -a_2 & a_1 & 0 \end{bmatrix} \quad (\text{B.14})$$

which satisfies the constraint that all rows and columns have the same L2-norm. This is the same general form as the quaternion update matrix $\tilde{\Omega}$ in Equations 5.13 and 5.19. Given \mathbf{A} , we now wish to calculate the following matrix exponentiation

$$\mathbf{B} = \exp(-\alpha \mathbf{A}) . \quad (\text{B.15})$$

In general (for arbitrary matrices) this cannot be solved in closed form and iterative approximations must be used. However, due to the special constrained skew-symmetric structure of \mathbf{A} , a closed form solution is possible. This can be derived by first rewriting Equation B.15 using the power series expansion of the matrix exponential function:

$$\exp(-\alpha \mathbf{A}) = \sum_{n=0}^{\infty} \frac{(-\alpha)^n \mathbf{A}^n}{n!} , \quad (\text{B.16})$$

where \mathbf{A}^n is the n th matrix power of \mathbf{A} , which is defined for a positive integer n as the matrix product of n copies of \mathbf{A} ,

$$\mathbf{A}^n = \underbrace{\mathbf{A}\mathbf{A}\cdots\mathbf{A}}_n . \quad (\text{B.17})$$

Now, an interesting property of skew-symmetric matrices is that their matrix powers have a very elegant self-similar property given by,

$$\mathbf{A}^n = \begin{cases} (-1)^{\frac{n}{2}} v^n \mathbf{I} & n = 2, 4, 6, \dots \quad (\text{even}) \\ (-1)^{\frac{n-1}{2}} v^{n-1} \mathbf{A} & n = 3, 5, 7, \dots \quad (\text{odd}) \end{cases} , \quad (\text{B.18})$$

where

$$v = \sqrt{\sum_{i=0}^{N-1} (\mathbf{A}_{0,i})^2} \quad (\text{B.19})$$

is the normal Euclidean vector norm of the elements of the first row of the skew-symmetric matrix \mathbf{A} treated as a vector. Here $\mathbf{A}_{0,i}$ is simply the i th column component of the zero'th row of \mathbf{A} , where N is the matrix dimension. We prove Equation B.18 through induction.

First we see if Equation B.18 holds for $n = 2$ and $n = 3$. Using Equation B.14 and Equation B.17, \mathbf{A}^2 is given by

$$\begin{aligned} \mathbf{A}^2 = \mathbf{AA} &= \begin{bmatrix} 0 & a_1 & a_2 & a_3 \\ -a_1 & 0 & -a_3 & a_2 \\ -a_2 & a_3 & 0 & -a_1 \\ -a_3 & -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} 0 & a_1 & a_2 & a_3 \\ -a_1 & 0 & -a_3 & a_2 \\ -a_2 & a_3 & 0 & -a_1 \\ -a_3 & -a_2 & a_1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} -v^2 & 0 & 0 & 0 \\ 0 & -v^2 & 0 & 0 \\ 0 & 0 & -v^2 & 0 \\ 0 & 0 & 0 & -v^2 \end{bmatrix} \end{aligned} \quad (\text{B.20})$$

$$= -v^2 \mathbf{I} \quad (\text{B.21})$$

$$= (-1)^{\frac{3}{2}} v^2 \mathbf{I}, \quad (\text{B.22})$$

where \mathbf{I} is the normal identity matrix with dimensions equal to those of \mathbf{A} and v is given by Equation B.19. In a similar fashion we calculate \mathbf{A}^3 as

$$\mathbf{A}^3 = \mathbf{A}^2 \mathbf{A} \quad (\text{B.23})$$

$$= (-v^2 \mathbf{I}) \mathbf{A} \quad (\text{B.24})$$

$$= -v^2 \mathbf{A} \quad (\text{B.25})$$

$$= (-1)^{\frac{3-1}{2}} v^{3-1} \mathbf{A}. \quad (\text{B.26})$$

For $n = k$, we get

$$\mathbf{A}^k = (-1)^{\frac{k}{2}} v^k \mathbf{I},$$

if k is even and

$$\mathbf{A}^k = (-1)^{\frac{k-1}{2}} v^{k-1} \mathbf{A} ,$$

if k is odd. Lets now calculate the value of \mathbf{A}^n for $n = k + 1$ for both the cases. If k is even, then \mathbf{A}^{k+1} is given by

$$\begin{aligned}\mathbf{A}^{k+1} &= \mathbf{A}^k \mathbf{A} \\ &= \left[(-1)^{\frac{k}{2}} v^k \mathbf{I} \right] \mathbf{A} \\ &= (-1)^{\frac{k}{2}} v^k \mathbf{A} \\ &= (-1)^{\frac{(k+1)-1}{2}} v^{(k+1)-1} \mathbf{A} .\end{aligned}$$

This is the correct form for \mathbf{A}^{k+1} according to Equation B.18 since $k+1$ is odd if k is even.

Finally, we have to test the relation for the case $n = k + 1$ if k is odd and hence $k + 1$ is even:

$$\begin{aligned}\mathbf{A}^{k+1} &= \mathbf{A}^k \mathbf{A} \\ &= \left[(-1)^{\frac{k-1}{2}} v^{k-1} \mathbf{A} \right] \mathbf{A} \\ &= (-1)^{\frac{k-1}{2}} v^{k-1} \mathbf{A}^2 \\ &= (-1)^{\frac{k-1}{2}} v^{k-1} (-v^2 \mathbf{I}) \\ &= (-1)^{\frac{k-1}{2}+1} v^{k-1+2} \mathbf{I} \\ &= (-1)^{\frac{k+1}{2}} v^{k+1} ,\end{aligned}$$

which is again the correct form based on Equation B.18. By induction this implies that Equation B.18 holds for all $n \geq 2$. Using this property, we can now expand Equation B.16

as

$$\exp(-\alpha \mathbf{A}) = \sum_{n=0}^{\infty} \frac{(-\alpha)^n \mathbf{A}^n}{n!} \quad (\text{B.27})$$

$$= \mathbf{I} - \alpha \mathbf{A} + \frac{\alpha^2}{2!} \mathbf{A}^2 - \frac{\alpha^3}{3!} \mathbf{A}^3 + \frac{\alpha^4}{4!} \mathbf{A}^4 - \frac{\alpha^5}{5!} \mathbf{A}^5 + \frac{\alpha^6}{6!} \mathbf{A}^6 - \dots \quad (\text{B.28})$$

$$= \mathbf{I} - \alpha \mathbf{A} - \frac{\alpha^2}{2!} v^2 \mathbf{I} + \frac{\alpha^3}{3!} v^2 \mathbf{A} + \frac{\alpha^4}{4!} v^4 \mathbf{I} - \frac{\alpha^5}{5!} v^4 \mathbf{A} - \frac{\alpha^6}{6!} v^6 \mathbf{I} + \dots \quad (\text{B.29})$$

$$= \mathbf{I} - \frac{\alpha^2}{2!} v^2 \mathbf{I} + \frac{\alpha^4}{4!} v^4 \mathbf{I} - \frac{\alpha^6}{6!} v^6 \mathbf{I} + \dots \quad (\text{B.30})$$

$$- \alpha \mathbf{A} + \frac{\alpha^3}{3!} v^2 \mathbf{A} - \frac{\alpha^5}{5!} v^4 \mathbf{A} + \frac{\alpha^7}{7!} v^7 \mathbf{A} - \dots \quad (\text{B.31})$$

$$= \mathbf{I} \sum_{n=0}^{\infty} \frac{(-1)^n \alpha^{2n}}{(2n)!} v^{2n} - \alpha \mathbf{A} \sum_{n=1}^{\infty} \frac{(-1)^{n-1} \alpha^{2n-2}}{(2n-1)!} v^{2n-2} \quad (\text{B.32})$$

$$= \mathbf{I} \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} (\alpha v)^{2n} - \alpha (\alpha v)^{-1} \mathbf{A} \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{(2n-1)!} (\alpha v)^{2n-1} \quad (\text{B.33})$$

$$= \mathbf{I} \cos(\alpha v) - \alpha \mathbf{A} \frac{\sin(\alpha v)}{\alpha v} \quad (\text{B.34})$$

$$= \mathbf{I} \cos(s) - \alpha \mathbf{A} \text{sinc}(s), \quad (\text{B.35})$$

where $s = \alpha v$, and Equation B.33 is simplified to Equation B.34 using the power series expansions of the cosine and sine functions. These are given by

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{1}{(2n)!} x^{2n}$$

and

$$\sin(x) = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{1}{(2n-1)!} x^{2n-1}.$$

This concludes the derivation and proof. In summary, for any 4×4 skew-symmetric matrix \mathbf{A} with constrained structure as given by Equation B.14, the matrix exponentiation function is given by the following closed form solution:

$$\mathbf{B} = \exp(-\alpha \mathbf{A}) = \mathbf{I} \cos(s) - \alpha \mathbf{A} \text{sinc}(s), \quad (\text{B.36})$$

where $s = \alpha v$ and v is given by Equation B.19.

Appendix C

ReBEL Toolkit

C.1 A Recursive Bayesian Estimation Library for Matlab®

ReBEL is a Matlab® toolkit of functions and scripts, designed to facilitate sequential Bayesian inference (estimation) in general state space models. This software consolidates most of the algorithmic research presented in this thesis and is available for free download for academic research. Numerous commercial entities have also licensed the code for commercial service and/or product development.

ReBEL was developed over a period of about six months and is still considered a 'work in progress' or beta-release quality. The toolkit currently contains most of the following functional units which can be used for state-, parameter- and joint-estimation: Kalman filter, Extended Kalman filter, all Sigma-point Kalman filters (SPKFs) presented in this thesis, including all the square root forms, all particle filters presented in this thesis. The code is designed to be as general, modular and extensible as possible, while at the same time trying to be as computationally efficient as possible. It has been tested with Matlab 6.1 (R12.1) and Matlab 6.5 (R13).

The toolkit can be downloaded for free from the main ReBEL website at <http://choosh.bme.ogi.edu/rebel> under a "free for academic use" license. This not only allows others to benefit directly from this work, either through general application to probabilistic inference problems or extension of the algorithms presented in this thesis, but also allows for the external verification of most of the experimental results presented in this thesis. ReBEL has also been licensed by a number of commercial companies through OHSU's Technology and Research Collaborations (TRC) office, resulting in the award of an OHSU

Commercialization Award. To date¹, the toolkit has been downloaded by more than 900 different researchers and research groups with a wide variety of *intended uses*. Here are but a short selection of some of these applications²:

- Numerous downloads for use as teaching aid in undergraduate and graduate classes in adaptive signal processing, image processing, machine learning, etc.
- Probabilistic inference for decision support systems.
- Bearing only tracking for submerged aquatic vehicles.
- Study of the probabilistic evolution of the Bosnian conflict using events data extracted from newswire³.
- Classification of biomedical time series data.
- Satellite orbit and attitude determination.
- Simultaneous localization and mapping (SLAM) for mobile robots.
- Speech enhancement and robust speech & speaker recognition.
- State and parameter estimation for waste -water treatment plant and hydrogeological modeling.
- Analysis of human brain imaging data.
- Fault detection in fault tolerant control systems
- Parameter estimation for nonlinear stochastic-dynamic models of simple cells in the visual cortex of primates.
- Echo cancellation

¹The initial version of **ReBEL** was released in May 2002, followed by a number of bug-fix releases as well as major updates. By the time of this publication, October 2003, more than 900 unique downloads of the toolkit for academic use has taken place.

²This data was collected using the ReBEL toolkit website. One of the requirements of the academic license is that the “user” has to provide information regarding their intended use of the toolkit.

³This must be one of the strangest applications of the ReBEL toolkit found to date by the author.

- Econometrics and financial time series analysis.
- Signal processing for cochlear implants.
- Stochastic image filtering for magnetic resonance imaging (MRI) applications.
- Non-rigid structure and motion from video.
- Data mining
- Smoothing of microwave remote sensing data.
- VLSI implementation of particle filters.
- Delay and channel estimation in CDMA communication channels.
- Target tracking problems in neuroscience: Exploring optimal algorithms that weakly-electric fish might use for detecting and tracking prey.
- Biomedical signal processing
- Neural network training for neural based approximate dynamic programming for flight control augmentation.
- Environmental modeling: Parameter estimation of the ocean mixed layer model (turbulence model in the ocean upper layer).
- Geolocation of mobile users in indoor environments.
- Modeling of astronomical time series, quasars, etc.
- Computational biology applications, i.e., bioinformatics.
- Blind signal separation
- Tracking of micro-organisms from microscopic image sequences.
- etc. ...

ReBEL : Recursive Bayesian Estimation Library

- Matlab toolkit that provides a unified framework for recursive Bayesian inference
 - ◆ State-, Parameter- and Dual-estimation
 - ◆ Algorithms:
 - Kalman filter
 - Extended Kalman filter
 - Sigma-point Kalman filters (SPKF)
 - Unscented Kalman filter (UKF)
 - Central difference Kalman filter (CDKF)
 - Square-root SPKFs
 - Other SPKF extensions
 - Particle Filters
 - Generic particle filter (Condensation algorithm, Bootstrap-filter)
 - Sigma-point particle filter (SPPF)
 - Gaussian mixture sigma-point particle filter (GMSPPF)
 - ◆ General, modular and extensible.
 - ◆ Will possibly be ported to C/C++ (and then rewrapped for Matlab via Mex code) in the future.

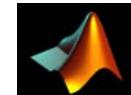
The logo for ReBEL consists of the word "ReBEL" in a bold, green, sans-serif font. The letters are slightly slanted to the right.

Figure C.1: The *ReBEL Toolkit* : general information

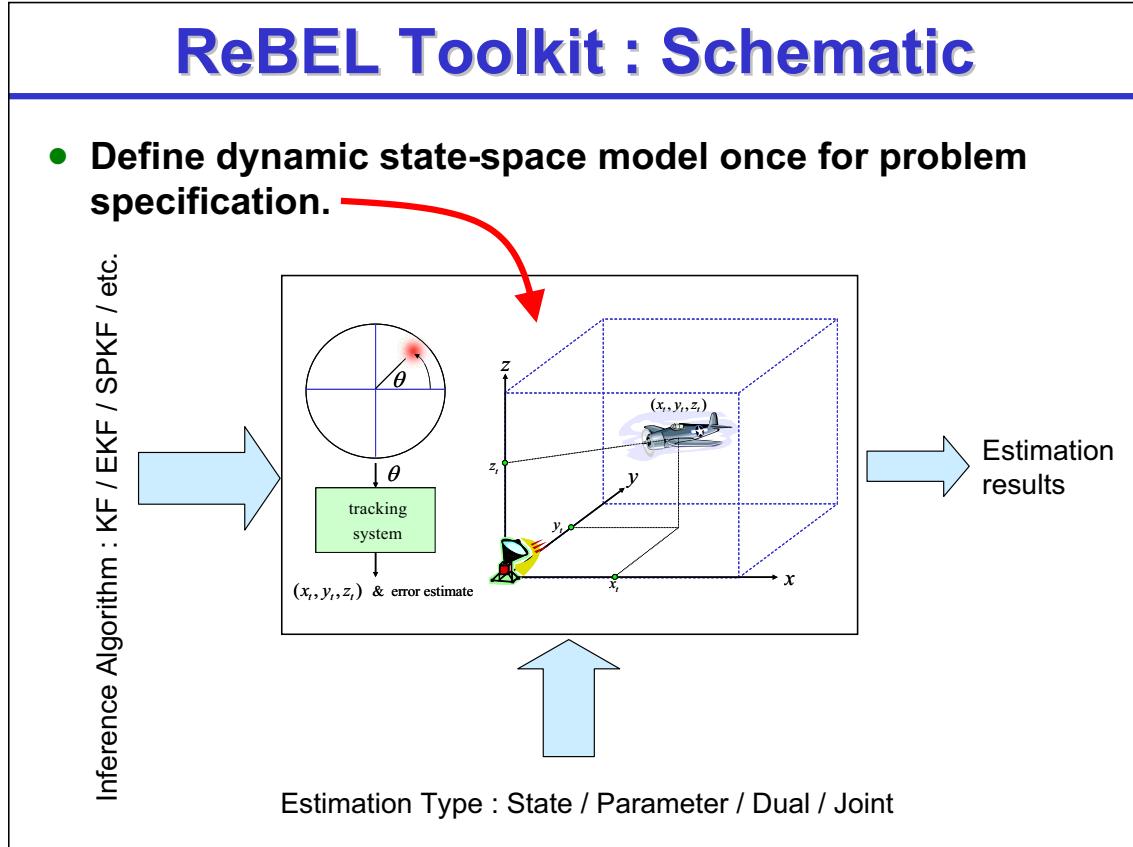


Figure C.2: The *ReBEL Toolkit* provides a general, modular, unified framework for probabilistic inference in generic DSSMs. The DSSM is defined once using a standard descriptor file format after which any of the inference algorithms can be applied for either state-, parameter- or dual/joint estimation. No reformulation of the DSSM needs to be done by hand. The toolkit automatically builds the needed data- and algorithmic structures to adapt the DSSM and inference algorithm to the inference task at hand.

ReBEL Toolkit : Code Example

```
|  
|  
model = gssm_helicopter('init');  
  
Arg.type = 'state'; % inference type (state estimation)  
Arg.tag = 'State estimation for helicopter system.';  
Arg.model = model; % arbitrary ID tag  
Arg.DSSM = InfDS; % DSSM data structure of external  
% system  
  
InfDS = geninfds(Arg); % Create inference data structure  
  
[pNoise, oNoise, InfDS] = gensysnoiseds(InfDS, 'CDKF'); % generate process and  
% observation noise sources  
  
[Xh, Px] = srckf(Xh(:,1), Px, pNoise, oNoise, y, [], [], InfDS); % Call SR-CDKF estimator  
  
|  
|
```

Figure C.3: The *ReBEL Toolkit* : code example

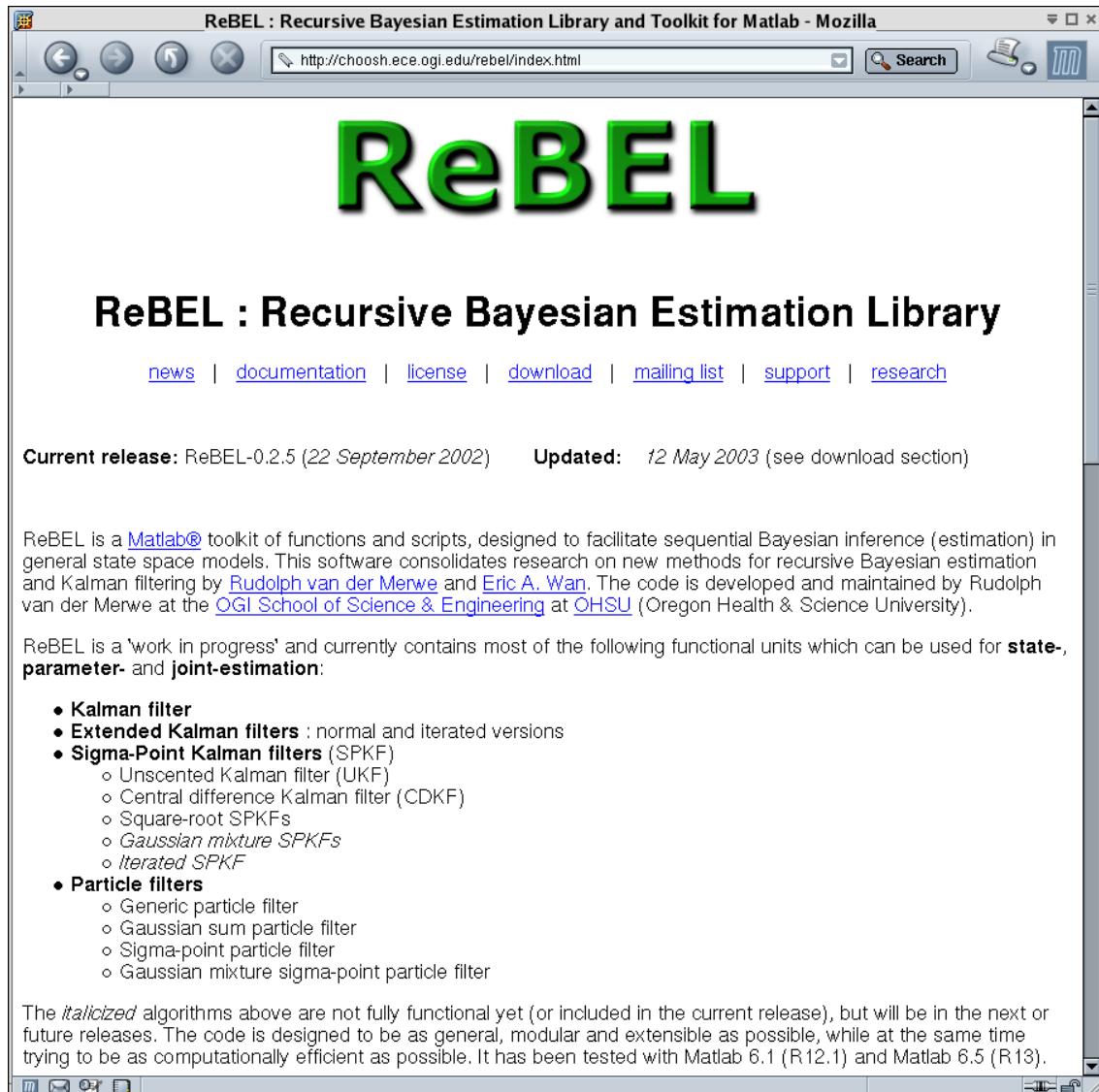


Figure C.4: The *ReBEL Toolkit* website : <http://choosh.bme.ogi.edu/rebel>

Biographical Note

Rudolph van der Merwe was born in Bellville, South Africa on November 2, 1973. He grew up in the greater Cape Town area, attended D. F. Malan High School and graduated first in his class in 1991. In 1995, he earned a B.Ing (Bachelors of Engineering) degree in Electrical and Electronic Engineering from the University of Stellenbosch with honors, *magna cum laude*. He continued on to graduate school, receiving a M.Ing (Masters of Engineering) degree in Electronic Engineering (*magna cum laude*) in 1997, also from the University of Stellenbosch. In 1998, he was awarded both a Fulbright Fellowship to further his studies in the United States of America, as well as a scholarship to join the doctoral program at Cambridge University in the United Kingdom. After much deliberation, introspection and toil, he finally decided to broaden his horizons to the far side of the Atlantic, accepted the Fulbright Fellowship and moved to America. He joined the Ph.D. program at the Oregon Graduate Institute of Science and Technology in the fall of 1998 where he pursued a doctoral degree in Electrical and Computer Engineering until he graduated in 2004. During this time of higher academic pursuits, he not only made new friends, explored the Pacific Northwest and made peace with the incessant rain, he also met his future wife, Zoë Chambers, whom he married in December 2002. Rudolph's other awards and honors include:

- Recipient of Oregon Health & Science University *Commercialization Award*, for IP licensing revenue resulting from commercialization of software developed during his Ph.D., 2003.
- Recipient of the *Paul Clayton Student Achievement Award* for excellence in Ph.D. studies and social service, OGI School of Science & Engineering, Oregon Health & Science University, 2002.
- First place award in annual *Student Paper Competition*, OGI School of Science & Engineering, Oregon Health & Science University, 2001-2002.
- *Fulbright Fellow*, awarded in 1998.
- Recipient of the *FRD Merit Scholarship for Graduate Studies*, University of Stellenbosch, 1997.

His broad range of interests include (but are not limited to) neural, adaptive, and machine learning approaches to signal processing, pattern recognition, robotics and autonomous control. Rudolph van der Merwe is author of the following publications:

- VAN DER MERWE, R. Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic-State Space Models. In *Proc. of Workshop on Advances in Machine Learning* (Montreal, Jun 2003).
- VAN DER MERWE, R., AND WAN, E. A. Gaussian Mixture Sigma-Point Particle Filters for Sequential Probabilistic Inference in Dynamic State-Space Models. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing* (Hong Kong, Apr 2003), IEEE.
- VAN DER MERWE, R., AND WAN, E. A. The Square-Root Unscented Kalman Filter for State- and Parameter-Estimation. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing* (Salt Lake City, May 2001), IEEE.
- VAN DER MERWE, R., AND WAN, E. A. Efficient Derivative-Free Kalman Filters for Online Learning. In *Proc. of ESANN* (Bruges, Apr 2001).
- WAN, E. A., AND VAN DER MERWE, R. *Kalman Filtering and Neural Networks*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. Wiley, 2001, Ch. 7 - The Unscented Kalman Filter.
- VAN DER MERWE, R., DE FREITAS, N., DOUCET, A., AND WAN, E. A. The Unscented Particle Filter. In *Advances in Neural Information Processing Systems (NIPS) 13* (Denver, Nov 2001).
- VAN DER MERWE, R., DE FREITAS, N., DOUCET, A., AND WAN, E. A. The Unscented Particle Filter. Tech. Rep. CUED/F-INFENG/TR 380, Cambridge University Engineering Department, Aug 2000.
- WAN, E. A., AND VAN DER MERWE, R. The Unscented Kalman Filter for Nonlinear Estimation. In *Proc. of IEEE Symposium 2000 (AS-SPCC)* (Lake Louise, Oct 2000).
- WAN, E. A., VAN DER MERWE, R., AND NELSON, A. T. Dual Estimation and the Unscented Transformation. In *Advances in Neural Information Processing Systems (NIPS) 12* (Denver, Nov 2000).
- WAN, E. A., AND VAN DER MERWE, R. Noise-Regularized Adaptive Filtering for Speech Enhancement. In *Proc. of EuroSpeech* (Budapest, Sep 1999).
- VAN DER MERWE, R., AND DU PREEZ, J. A. Hybrid Combination of Knowledge- and Cepstral-based Features for Phoneme Recognition. In *Proc. of COMSIG* (Cape Town, Sep 1998), IEEE.
- VAN DER MERWE, R. *Variations on Statistical Phoneme Recognition: A Hybrid Approach*. Masters thesis. University of Stellenbosch, Dec 1997.

*“And if the cloud bursts, thunder in your ear
You shout and no one seems to hear
And if the band you’re in starts playing different tunes
I’ll see you on the dark side of the moon.”*

- Pink Floyd, “Brain Damage”