ims Textbooks

Bayesian Filtering and Smoothing

Simo Särkkä

CAMBRIDGE

CAMBRIDGE more information - www.cambridge.org/9781107030657

Bayesian Filtering and Smoothing

Filtering and smoothing methods are used to produce an accurate estimate of the state of a time-varying system based on multiple observational inputs (data). Interest in these methods has exploded in recent years, with numerous applications emerging in fields such as navigation, aerospace engineering, telecommunications, and medicine.

This compact, informal introduction for graduate students and advanced undergraduates presents the current state-of-the-art filtering and smoothing methods in a unified Bayesian framework. Readers learn what non-linear Kalman filters and particle filters are, how they are related, and their relative advantages and disadvantages. They also discover how state-of-the-art Bayesian parameter estimation methods can be combined with state-of-the-art filtering and smoothing algorithms.

The book's practical and algorithmic approach assumes only modest mathematical prerequisites. Examples include MATLAB computations, and the numerous end-of-chapter exercises include computational assignments. MATLAB/GNU Octave source code is available for download at www.cambridge.org/sarkka, promoting hands-on work with the methods.

SIMO SÄRKKÄ worked, from 2000 to 2010, with Nokia Ltd., Indagon Ltd., and the Nalco Company in various industrial research projects related to telecommunications, positioning systems, and industrial process control. Currently, he is a Senior Researcher with the Department of Biomedical Engineering and Computational Science at Aalto University, Finland, and Adjunct Professor with Tampere University of Technology and Lappeenranta University of Technology. In 2011 he was a visiting scholar with the Signal Processing and Communications Laboratory of the Department of Engineering at the University of Cambridge. His research interests are in state and parameter estimation in stochastic dynamic systems and, in particular, Bayesian methods in signal processing, machine learning, and inverse problems with applications to brain imaging, positioning systems, computer vision, and audio signal processing. He is a Senior Member of the IEEE.

INSTITUTE OF MATHEMATICAL STATISTICS
TEXTBOOKS

IMS Textbooks give introductory accounts of topics of current concern suitable for advanced courses at master's level, for doctoral students and for individual study. They are typically shorter than a fully developed textbook, often arising from material created for a topical course. Lengths of 100–290 pages are envisaged. The books typically contain exercises.

Bayesian Filtering and Smoothing

SIMO SÄRKKÄ
Aalto University, Finland

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town,
Singapore, São Paulo, Delhi, Mexico City

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9781107030657

First published 2013

Printed and bound in the United Kingdom by the MPG Books Group

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication data

ISBN 978-1-107-03065-7 Hardback
ISBN 978-1-107-61928-9 Paperback

Additional resources for this publication at www.cambridge.org/sarkka

Contents

Preface

The aim of this book is to give a concise introduction to non-linear Kalman filtering and smoothing, particle filtering and smoothing, and to the related parameter estimation methods. Although the book is intended to be an introduction, the mathematical ideas behind all the methods are carefully explained, and a mathematically inclined reader can get quite a deep understanding of the methods by reading the book. The book is purposely kept short for quick reading.

The book is mainly intended for advanced undergraduate and graduate students in applied mathematics and computer science. However, the book is suitable also for researchers and practitioners (engineers) who need a concise introduction to the topic on a level that enables them to implement or use the methods. The assumed background is linear algebra, vector calculus, Bayesian inference, and MATLAB® programming skills.

As implied by the title, the mathematical treatment of the models and algorithms in this book is Bayesian, which means that all the results are treated as being approximations to certain probability distributions or their parameters. Probability distributions are used both to represent uncertainties in the models and for modeling the physical randomness. The theories of non-linear filtering, smoothing, and parameter estimation are formulated in terms of Bayesian inference, and both the classical and recent algorithms are derived using the same Bayesian notation and formalism. This Bayesian approach to the topic is far from new. It was pioneered by Stratonovich in the 1950s and 1960s – even before Kalman's seminal article in 1960. Thus the theory of non-linear filtering has been Bayesian from the beginning (see Jazwinski, 1970).

Chapter 1 is a general introduction to the idea and applications of Bayesian filtering and smoothing. The purpose of Chapter 2 is to briefly review the basic concepts of Bayesian inference as well as the basic numerical methods used in Bayesian computations. Chapter 3 starts with a step-by-step introduction to recursive Bayesian estimation via solving a

linear regression problem in a recursive manner. The transition to Bayesian filtering and smoothing theory is explained by extending and generalizing the problem. The first Kalman filter of the book is also encountered in this chapter.

The Bayesian filtering theory starts in Chapter 4 where we derive the general Bayesian filtering equations and, as their special case, the celebrated Kalman filter. Non-linear extensions of the Kalman filter, the extended Kalman filter (EKF), the statistically linearized filter (SLF), and the unscented Kalman filter (UKF) are presented in Chapter 5. Chapter 6 generalizes these filters into the framework of Gaussian filtering. The Gauss–Hermite Kalman filter (GHKF) and cubature Kalman filter (CKF) are then derived from the general framework. Sequential Monte Carlo (SMC) based particle filters (PF) are explained in Chapter 7 by starting from the basic SIR filter and ending with Rao–Blackwellized particle filters (RBPF).

Chapter 8 starts with a derivation of the general (fixed-interval) Bayesian smoothing equations and then continues to a derivation of the Rauch–Tung–Striebel (RTS) smoother as their special case. In that chapter we also briefly discuss two-filter smoothing. The extended RTS smoother (ERTSS), statistically linearized RTS smoother (SLRTSS), and the unscented RTS smoother (URTSS) are presented in Chapter 9. The general Gaussian smoothing framework is presented in Chapter 10, and the Gauss–Hermite RTS smoother (GHRTSS) and the cubature RTS smoother (CRTSS) are derived as its special cases. We also discuss Gaussian fixed-point and fixed-lag smoothing in the same chapter. In Chapter 11 we start by showing how the basic SIR particle filter can be used to approximate the smoothing solutions with a small modification. We then introduce the numerically better backward-simulation particle smoother and the reweighting (or marginal) particle smoother. Finally, we discuss the implementation of Rao–Blackwellized particle smoothers.

Chapter 12 is an introduction to parameter estimation in state space models concentrating on optimization and expectation–maximization (EM) based computation of maximum likelihood (ML) and maximum a posteriori (MAP) estimates, as well as to Markov chain Monte Carlo (MCMC) methods. We start by presenting the general methods and then show how Kalman filters and RTS smoothers, non-linear Gaussian filters and RTS smoothers, and finally particle filters and smoothers, can be used to compute or approximate the quantities needed in implementation of parameter estimation methods. This leads to, for example, classical EM algorithms for state space models, as well as to particle EM and

particle MCMC methods. We also discuss how Rao–Blackwellization can sometimes be used to help parameter estimation.

Chapter 13 is an epilogue where we give some general advice on the selection of different methods for different purposes. We also discuss and give references to various technical points and related topics that are important, but did not fit into this book.

Each of the chapters ends with a range of exercises, which give the reader hands-on experience in implementing the methods and in selecting the appropriate method for a given purpose. The MATLAB® source code needed in the exercises as well as various other material can be found on the book's web page at `www.cambridge.org/sarkka`.

This book is an outgrowth of lecture notes of courses that I gave during the years 2009–2012 at Helsinki University of Technology, Aalto University, and Tampere University of Technology, Finland. Most of the text was written while I was working at the Department of Biomedical Engineering and Computational Science (BECS) of Aalto University (formerly Helsinki University of Technology), but some of the text was written during my visit to the Department of Engineering at the University of Cambridge, UK. I am grateful to the former Centre of Excellence in Computational Complex Systems Research of the Academy of Finland, BECS, and Aalto University School of Science for providing me with the research funding which made this book possible.

I would like to thank Jouko Lampinen and Aki Vehtari from BECS for giving me the opportunity to do the research and for co-operation which led to this book. Arno Solin, Robert Piché, Juha Sarmavuori, Thomas Schön, Pete Bunch, and Isambi S. Mbalawata deserve thanks for careful checking of the book and for giving a lot of useful suggestions for improving the text. I am also grateful to Jouni Hartikainen, Ville Väänänen, Heikki Haario, and Simon Godsill for research co-operation that led to improvement of my understanding of the topic as well as to the development of some of the methods which now are explained in this book. I would also like to thank Diana Gillooly from Cambridge University Press and series editor Susan Holmes for suggesting the publication of my lecture notes in book form. Finally, I am grateful to my wife Susanne for her support and patience during the writing of this book.

Simo Särkkä
Vantaa, Finland

Symbols and abbreviations

General notation

$a, b, c, x, t, \alpha, \beta$	Scalars
$\mathbf{a}, \mathbf{f}, \mathbf{s}, \mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta}$	Vectors
$\mathbf{A}, \mathbf{F}, \mathbf{S}, \mathbf{X}, \mathbf{Y}$	Matrices
$\mathcal{A}, \mathcal{F}, \mathcal{S}, \mathcal{X}, \mathcal{Y}$	Sets
$\mathbb{A}, \mathbb{F}, \mathbb{S}, \mathbb{X}, \mathbb{Y}$	Spaces

Notational conventions

$\mathbf{A}^\mathsf{T}$	Transpose of matrix
$\mathbf{A}^{-1}$	Inverse of matrix
$\mathbf{A}^{-\mathsf{T}}$	Inverse of transpose of matrix
$[\mathbf{A}]_i$	ith column of matrix $\mathbf{A}$
$[\mathbf{A}]_{ij}$	Element at ith row and jth column of matrix $\mathbf{A}$
$\lvert a \rvert$	Absolute value of scalar a
$\lvert \mathbf{A} \rvert$	Determinant of matrix $\mathbf{A}$
$\mathrm{d}\mathbf{x}/\mathrm{d}t$	Time derivative of $\mathbf{x}(t)$
$\frac{\partial g_i(\mathbf{x})}{\partial x_j}$	Partial derivative of g_i with respect to x_j
$(a_1, \ldots, a_n)$	Column vector with elements $a_1, \ldots, a_n$
$(a_1 \ \cdots \ a_n)$	Row vector with elements $a_1, \ldots, a_n$
$(a_1 \ \cdots \ a_n)^\mathsf{T}$	Column vector with elements $a_1, \ldots, a_n$
$\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}$	Gradient (column vector) of scalar function g
$\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}$	Jacobian matrix of vector valued function $\mathbf{x} \to \mathbf{g}(\mathbf{x})$
$\mathrm{Cov}[\mathbf{x}]$	Covariance $\mathrm{Cov}[\mathbf{x}] = \mathrm{E}[(\mathbf{x} - \mathrm{E}[\mathbf{x}])\,(\mathbf{x} - \mathrm{E}[\mathbf{x}])^\mathsf{T}]$ of the random variable $\mathbf{x}$
$\mathrm{diag}(a_1, \ldots, a_n)$	Diagonal matrix with diagonal values $a_1, \ldots, a_n$
$\sqrt{\mathbf{P}}$	Matrix such that $\mathbf{P} = \sqrt{\mathbf{P}}\,\sqrt{\mathbf{P}}^\mathsf{T}$
$\mathrm{E}[\mathbf{x}]$	Expectation of $\mathbf{x}$
$\mathrm{E}[\mathbf{x} \mid \mathbf{y}]$	Conditional expectation of $\mathbf{x}$ given $\mathbf{y}$

$\int f(\mathbf{x})\, \mathrm{d}\mathbf{x}$	Lebesgue integral of $f(\mathbf{x})$ over the space $\mathbb{R}^n$
$p(\mathbf{x})$	Probability density of continuous random variable $\mathbf{x}$ or probability of discrete random variable $\mathbf{x}$
$p(\mathbf{x} \mid \mathbf{y})$	Conditional probability density or conditional probability of $\mathbf{x}$ given $\mathbf{y}$
$p(\mathbf{x}) \propto q(\mathbf{x})$	$p(\mathbf{x})$ is proportional to $q(\mathbf{x})$, that is, there exists a constant c such that $p(\mathbf{x}) = c\, q(\mathbf{x})$ for all values of $\mathbf{x}$
$\mathrm{tr}\, \mathbf{A}$	Trace of matrix $\mathbf{A}$
$\mathrm{Var}[x]$	Variance $\mathrm{Var}[x] = \mathrm{E}[(x - \mathrm{E}[x])^2]$ of the scalar random variable x
$x \gg y$	x is much greater than y
$x_{i,k}$	ith component of vector $\mathbf{x}_k$
$\mathbf{x} \sim p(\mathbf{x})$	Random variable $\mathbf{x}$ has the probability density or probability distribution $p(\mathbf{x})$
$\mathbf{x} \triangleq \mathbf{y}$	$\mathbf{x}$ is defined to be equal to $\mathbf{y}$
$\mathbf{x} \approx \mathbf{y}$	$\mathbf{x}$ is approximately equal to $\mathbf{y}$
$\mathbf{x} \simeq \mathbf{y}$	$\mathbf{x}$ is assumed to be approximately equal to $\mathbf{y}$
$\mathbf{x}_{0:k}$	Set or sequence containing the vectors $\{\mathbf{x}_0, \ldots, \mathbf{x}_k\}$
$\dot{\mathbf{x}}$	Time derivative of $\mathbf{x}(t)$

Symbols

α	Parameter of the unscented transform or pendulum angle
α_i	Acceptance probability in an MCMC method
$\bar{\alpha}_*$	Target acceptance rate in an adaptive MCMC
β	Parameter of the unscented transform
$\delta(\cdot)$	Dirac delta function
$\delta\mathbf{x}$	Difference of $\mathbf{x}$ from the mean $\delta\mathbf{x} = \mathbf{x} - \mathbf{m}$
Δt	Sampling period
Δt_k	Length of the time interval $\Delta t_k = t_{k+1} - t_k$
ε_k	Measurement error at the time step k
$\boldsymbol{\varepsilon}_k$	Vector of measurement errors at the time step k
$\boldsymbol{\theta}$	Vector of parameters
$\boldsymbol{\theta}_k$	Vector of parameters at the time step k
$\boldsymbol{\theta}^{(n)}$	Vector of parameters at iteration n of the EM-algorithm
$\boldsymbol{\theta}^{(i)}$	Vector of parameters at iteration i of the MCMC-algorithm
$\boldsymbol{\theta}^*$	Candidate point in the MCMC-algorithm
$\hat{\boldsymbol{\theta}}^{\mathrm{MAP}}$	Maximum a posteriori (MAP) estimate of parameter $\boldsymbol{\theta}$
κ	Parameter of the unscented transform

λ	Parameter of the unscented transform
λ'	Parameter of the unscented transform
λ''	Parameter of the unscented transform
$\boldsymbol{\mu}_k$	Predicted mean of measurement $\mathbf{y}_k$ in a Kalman/Gaussian filter at the time step k
$\boldsymbol{\mu}_\mathrm{L}$	Mean in the linear approximation of a non-linear transform
$\boldsymbol{\mu}_\mathrm{M}$	Mean in the Gaussian moment matching approximation
$\boldsymbol{\mu}_\mathrm{Q}$	Mean in the quadratic approximation
$\boldsymbol{\mu}_\mathrm{S}$	Mean in the statistical linearization approximation
$\boldsymbol{\mu}_\mathrm{U}$	Mean in the unscented approximation
$\pi(\cdot)$	Importance distribution
σ^2	Variance
σ_i^2	Variance of noise component i
$\boldsymbol{\Sigma}$	Auxiliary matrix needed in the EM-algorithm
$\boldsymbol{\Sigma}_i$	Proposal distribution covariance in the Metropolis algorithm
$\varphi_k(\boldsymbol{\theta})$	Energy function at the time step k
$\Phi(\cdot)$	A function returning the lower triangular part of its argument
$\boldsymbol{\Phi}$	An auxiliary matrix needed in the EM-algorithm
ξ	Unit Gaussian random variable
$\xi^{(i)}$	ith scalar unit sigma point
$\boldsymbol{\xi}$	Vector of unit Gaussian random variables
$\boldsymbol{\xi}^{(i)}$	ith unit sigma point vector
$\boldsymbol{\xi}^{(i_1,\ldots,i_n)}$	Unit sigma point in the multivariate Gauss–Hermite cubature
$\mathbf{a}$	Action in decision theory, or a part of a mean vector
$\mathbf{a}_\mathrm{o}$	Optimal action
$\mathbf{a}(t)$	Acceleration
$\mathbf{A}$	Dynamic model matrix in a linear time-invariant model, the lower triangular Cholesky factor of a covariance matrix, the upper left block of a covariance matrix, a coefficient in statistical linearization, or an arbitrary matrix
$\mathbf{A}_k$	Dynamic model matrix (i.e., transition matrix) of the jump from step k to step $k+1$
$\mathbf{b}$	The lower part of a mean vector, the offset term in statistical linearization, or an arbitrary vector
$\mathbf{B}$	Lower right block of a covariance matrix, an auxiliary matrix needed in the EM-algorithm, or an arbitrary matrix
$\mathbf{B}_{j\|k}$	Gain matrix in a fixed-point or fixed-lag Gaussian smoother
c	Scalar constant
$C(\cdot)$	Cost or loss function

$\mathbf{C}$	The upper right block of a covariance matrix, an auxiliary matrix needed in the EM-algorithm, or an arbitrary matrix
$\mathbf{C}_k$	Cross-covariance matrix in a non-linear Kalman filter
$\mathbf{C}_\mathrm{L}$	Cross-covariance in the linear approximation of a non-linear transform
$\mathbf{C}_\mathrm{M}$	Cross-covariance in the Gaussian moment matching approximation of a non-linear transform
$\mathbf{C}_\mathrm{Q}$	Cross-covariance in the quadratic approximation
$\mathbf{C}_\mathrm{S}$	Cross-covariance in the statistical linearization approximation
$\mathbf{C}_\mathrm{U}$	Cross-covariance in the unscented approximation
d	Positive integer, usually dimensionality of the parameters
d_i	Order of a monomial
$\mathrm{d}t$	Differential of time variable t
$\mathrm{d}\mathbf{x}$	Differential of vector $\mathbf{x}$
$\mathbf{D}$	Derivative of the Cholesky factor, an auxiliary matrix needed in the EM-algorithm, or an arbitrary matrix
$\mathbf{D}_k$	Cross-covariance matrix in a non-linear RTS smoother or an auxiliary matrix used in derivations
$\mathbf{e}_i$	Unit vector in the direction of the coordinate axis i
$\mathbf{f}(\cdot)$	Dynamic transition function in a state space model
$\mathbf{F}_\mathbf{x}(\cdot)$	Jacobian matrix of the function $\mathbf{x} \to \mathbf{f}(\mathbf{x})$
$\mathbf{F}$	Feedback matrix of a continuous-time linear state space model
$\mathbf{F}^{(i)}_{\mathbf{xx}}(\cdot)$	Hessian matrix of $\mathbf{x} \to f_i(\mathbf{x})$
$F[\cdot]$	An auxiliary functional needed in the derivation of the EM-algorithm
g	Gravitation acceleration
$g(\cdot)$	An arbitrary function
$g_i(\cdot)$	An arbitrary function
$\mathbf{g}(\cdot)$	An arbitrary function
$\mathbf{g}^{-1}(\cdot)$	Inverse function of $\mathbf{g}(\cdot)$
$\tilde{\mathbf{g}}(\cdot)$	Augmented function with elements $(\mathbf{x}, \mathbf{g}(\cdot))$
$\mathbf{G}_k$	Gain matrix in an RTS smoother
$\mathbf{G}_\mathbf{x}(\cdot)$	Jacobian matrix of the function $\mathbf{x} \to \mathbf{g}(\mathbf{x})$
$\mathbf{G}^{(i)}_{\mathbf{xx}}(\cdot)$	Hessian matrix of $\mathbf{x} \to g_i(\mathbf{x})$
$H_p(\cdot)$	pth order Hermite polynomial
$\mathbf{H}$	Measurement model matrix in a linear Gaussian model, or a Hessian matrix
$\mathbf{H}_k$	Measurement model matrix at the time step k in a linear Gaussian model

$\mathbf{H}_{\mathbf{x}}(\cdot)$	Jacobian matrix of the function $\mathbf{x} \rightarrow \mathbf{h}(\mathbf{x})$
$\mathbf{H}_{\mathbf{xx}}^{(i)}(\cdot)$	Hessian matrix of $\mathbf{x} \rightarrow h_i(\mathbf{x})$
$\mathbf{h}(\cdot)$	Measurement model function in a state space model
i	Integer valued index variable
$\mathbf{I}$	Identity matrix
$I_i(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})$	An integral term needed in the EM-algorithm
$\mathbf{J}(\cdot)$	Jacobian matrix
k	Time step number
$\mathbf{K}_k$	Gain matrix of a Kalman/Gaussian filter
$\mathbf{L}$	Noise coefficient (i.e., dispersion) matrix of a continuous-time linear state space model
$\mathcal{L}(\cdot)$	Likelihood function
m	Dimensionality of a measurement, mean of the univariate Gaussian distribution, or the mass
$\mathbf{m}$	Mean of a Gaussian distribution
$\tilde{\mathbf{m}}$	Mean of an augmented random variable
$\mathbf{m}_k$	Mean of a Kalman/Gaussian filter at the time step k
$\mathbf{m}_k^{(i)}$	Mean of the Kalman filter in the particle i of RBPF at the time step k
$\mathbf{m}_{0:T}^{(i)}$	History of means of the Kalman filter in the particle i of RBPF
$\tilde{\mathbf{m}}_k$	Augmented mean at the time step k or an auxiliary variable used in derivations
$\mathbf{m}_k^-$	Predicted mean of a Kalman/Gaussian filter at the time step k just before the measurement $\mathbf{y}_k$
$\mathbf{m}_k^{-(i)}$	Predicted mean of the Kalman filter in the particle i of RBPF at the time step k
$\tilde{\mathbf{m}}_k^-$	Augmented predicted mean at the time step k
$\mathbf{m}_k^{\mathrm{s}}$	Mean computed by a Gaussian fixed-interval (RTS) smoother for the time step k
$\mathbf{m}_{0:T}^{\mathrm{s},(i)}$	History of means of the RTS smoother in the particle i of RBPS
$\mathbf{m}_{k\|n}$	Conditional mean of $\mathbf{x}_k$ given $\mathbf{y}_{1:n}$
n	Positive integer, usually the dimensionality of the state
n'	Augmented state dimensionality in a non-linear transform
n''	Augmented state dimensionality in a non-linear transform
N	Positive integer, usually the number of Monte Carlo samples
$\mathrm{N}(\cdot)$	Gaussian distribution (i.e., normal distribution)

p	Order of a Hermite polynomial
P	Variance of the univariate Gaussian distribution
$\mathbf{P}$	Covariance of the Gaussian distribution
$\tilde{\mathbf{P}}$	Covariance of an augmented random variable
$\mathbf{P}_k$	Covariance of a Kalman/Gaussian filter at the time step k
$\mathbf{P}_k^{(i)}$	Covariance of the Kalman filter in the particle i of RBPF at the time step k
$\mathbf{P}_{0:T}^{(i)}$	History of covariances of the Kalman filter in the particle i of RBPF
$\tilde{\mathbf{P}}_k$	Augmented covariance at the time step k or an auxiliary variable used in derivations
$\mathbf{P}_k^-$	Predicted covariance of a Kalman/Gaussian filter at the time step k just before the measurement $\mathbf{y}_k$
$\tilde{\mathbf{P}}_k^-$	Augmented predicted covariance at the time step k
$\mathbf{P}_k^{-(i)}$	Predicted covariance of the Kalman filter in the particle i of RBPF at the time step k
$\mathbf{P}_k^{\mathrm{s}}$	Covariance computed by a Gaussian fixed-interval (RTS) smoother for the time step k
$\mathbf{P}_{0:T}^{\mathrm{s},(i)}$	History of covariances of the RTS smoother in the particle i of RBPS
$\mathbf{P}_{k\|n}$	Conditional covariance of $\mathbf{x}_k$ given $\mathbf{y}_{1:n}$
q^{c}	Spectral density of a white noise process
q_i^{c}	Spectral density of component i of a white noise process
$q(\cdot)$	Proposal distribution in the MCMC algorithm, or an arbitrary distribution in the derivation of the EM-algorithm
$q^{(n)}$	Distribution approximation on the nth step of the EM-algorithm
$\mathbf{q}$	Gaussian random vector
$\mathbf{q}_k$	Gaussian process noise
Q	Variance of scalar process noise
$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})$	An auxiliary function needed in the EM-algorithm
$\mathbf{Q}$	Covariance of the process noise in a time-invariant model
$\mathbf{Q}_k$	Covariance of the process noise at the jump from step k to $k+1$
r_k	Scalar Gaussian measurement noise
$\mathbf{r}_k$	Vector of Gaussian measurement noises
R	Variance of scalar measurement noise
$\mathbf{R}$	Covariance matrix of the measurement in a time-invariant model

$\mathbf{R}_k$	Covariance matrix of the measurement at the time step k
$\mathbb{R}$	Space of real numbers
$\mathbb{R}^n$	n-dimensional space of real numbers
$\mathbb{R}^{n \times m}$	Space of real $n \times m$ matrices
S	Number of backward-simulation draws
$\mathbf{S}_k$	Innovation covariance of a Kalman/Gaussian filter at step k
$\mathbf{S}_\mathrm{L}$	Covariance in the linear approximation of a non-linear transform
$\mathbf{S}_\mathrm{M}$	Covariance in the Gaussian moment matching approximation of a non-linear transform
$\mathbf{S}_\mathrm{Q}$	Covariance in the quadratic approximation of a non-linear transform
$\mathbf{S}_\mathrm{S}$	Covariance in the statistical linearization approximation of a non-linear transform
$\mathbf{S}_\mathrm{U}$	Covariance in the unscented approximation of a non-linear transform
t	Time variable $t \in [0, \infty)$
t_k	Time of the step k (usually time of the measurement y_k)
T	Index of the last time step, the final time of a time interval
$\mathcal{T}_k$	Sufficient statistics
u	Uniform random variable
$\mathbf{u}_k$	Latent (non-linear) variable in a Rao–Blackwellized particle filter or smoother
$\mathbf{u}_k^{(i)}$	Latent variable value in particle i
$\mathbf{u}_{0:k}^{(i)}$	History of latent variable values in particle i
$U(\cdot)$	Utility function
$\mathrm{U}(\cdot)$	Uniform distribution
$v_k^{(i)}$	Unnormalized weight in an SIR particle filter based likelihood evaluation
$\mathbf{v}_k$	Innovation vector of a Kalman/Gaussian filter at step k
$w^{(i)}$	Normalized weight of the particle i in importance sampling
$\tilde{w}^{(i)}$	Weight of the particle i in importance sampling
$w^{*(i)}$	Unnormalized weight of the particle i in importance sampling
$w_k^{(i)}$	Normalized weight of the particle i on step k of a particle filter
$w_{k\|n}^{(i)}$	Normalized weight of a particle smoother
w_i	Weight i in a regression model
$\mathbf{w}_k$	Vector of weights at the time step k in a regression model
$\mathbf{w}(t)$	Gaussian white noise process
W	Weight in the cubature or unscented approximation

W_i	ith weight in sigma-point approximation or in Gauss–Hermite quadrature
$W_i^{(\mathrm{m})}$	Mean weight of the unscented transform
$W_i^{(\mathrm{m})'}$	Mean weight of the unscented transform
$W_i^{(\mathrm{c})}$	Covariance weight of the unscented transform
$W_i^{(\mathrm{c})'}$	Covariance weight of the unscented transform
$W_{i_1,\ldots,i_n}$	Weight in multivariate Gauss–Hermite cubature
x	Scalar random variable or state, sometimes regressor variable, or a generic scalar variable
$\mathbf{x}$	Random variable or state
$\mathbf{x}^{(i)}$	ith Monte Carlo draw from the distribution of $\mathbf{x}$
$\mathbf{x}_k$	State at the time step k
$\tilde{\mathbf{x}}_k$	Augmented state at the time step k
$\mathbf{x}_{0:k}$	Set containing the state vectors $\{\mathbf{x}_0, \ldots, \mathbf{x}_k\}$
$\mathbf{x}_{0:k}^{(i)}$	The history of the states in the particle i
$\tilde{\mathbf{x}}_{0:T}^{(j)}$	State trajectory simulated by a backward-simulation particle smoother
$\mathbf{X}$	Matrix of regressors
$\mathbf{X}_k$	Matrix of regressors up to the time step k
$\mathcal{X}^{(\cdot)}$	Sigma point of $\mathbf{x}$
$\tilde{\mathcal{X}}^{(\cdot)}$	Augmented sigma point of $\mathbf{x}$
$\mathcal{X}_k^{(\cdot)}$	Sigma point of the state $\mathbf{x}_k$
$\tilde{\mathcal{X}}_k^{(\cdot)}$	Augmented sigma point of the state $\mathbf{x}_k$
$\hat{\mathcal{X}}_k^{(\cdot)}$	Predicted sigma point of the state $\mathbf{x}_k$
$\mathcal{X}_k^{-(\cdot)}$	Sigma point of the predicted state $\mathbf{x}_k$
$\tilde{\mathcal{X}}_k^{-(\cdot)}$	Augmented sigma point of the predicted state $\mathbf{x}_k$
$\mathbf{y}$	Random variable or measurement
$\mathbf{y}_k$	Measurement at the time step k
$\mathbf{y}_{1:k}$	Set containing the measurement vectors $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$
$\mathcal{Y}^{(\cdot)}$	Sigma point of $\mathbf{y}$
$\tilde{\mathcal{Y}}^{(\cdot)}$	Augmented sigma point of $\mathbf{y}$
$\hat{\mathcal{Y}}_k^{(\cdot)}$	ith predicted sigma point of the measurement $\mathbf{y}_k$ at step k
Z	Normalization constant
Z_k	Normalization constant at the time step k
∞	Infinity

Abbreviations

ADF	Assumed density filter
AM	Adaptive Metropolis (algorithm)
AMCMC	Adaptive Markov chain Monte Carlo
AR	Autoregressive (model)
ARMA	Autoregressive moving average (model)
ASIR	Auxiliary sequential importance resampling
BS-PS	Backward-simulation particle smoother
CDKF	Central differences Kalman filter
CKF	Cubature Kalman filter
CLT	Central limit theorem
CPF	Cubature particle filter
CRLB	Cramér–Rao lower bound
DLM	Dynamic linear model
DOT	Diffuse optical tomography
DSP	Digital signal processing
EC	Expectation correction
EEG	Electroencephalography
EKF	Extended Kalman filter
EM	Expectation–maximization
EP	Expectation propagation
ERTSS	Extended Rauch–Tung–Striebel smoother
FHKF	Fourier–Hermite Kalman filter
FHRTSS	Fourier–Hermite Rauch–Tung–Striebel smoother
fMRI	Functional magnetic resonance imaging
GHKF	Gauss–Hermite Kalman filter
GHPF	Gauss–Hermite particle filter
GHRTSS	Gauss–Hermite Rauch–Tung–Striebel smoother
GPB	Generalized pseudo-Bayesian
GPS	Global positioning system
HMC	Hamiltonian (or hybrid) Monte Carlo
HMM	Hidden Markov model
IMM	Interacting multiple model (algorithm)
INS	Inertial navigation system
IS	Importance sampling
InI	Inverse imaging
KF	Kalman filter
LMS	Least mean squares
LQG	Linear quadratic Gaussian (regulator)

LS	Least squares
MA	Moving average (model)
MAP	Maximum a posteriori
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
MEG	Magnetoencephalography
MH	Metropolis–Hastings
MKF	Mixture Kalman filter
ML	Maximum likelihood
MLP	Multi-layer perceptron
MMSE	Minimum mean squared error
MNE	Minimum norm estimate
MSE	Mean squared error
PF	Particle filter
PMCMC	Particle Markov chain Monte Carlo
PMMH	Particle marginal Metropolis–Hastings
PS	Particle smoother
QKF	Quadrature Kalman filter
RAM	Robust adaptive Metropolis (algorithm)
RBPF	Rao–Blackwellized particle filter
RBPS	Rao–Blackwellized particle smoother
RMSE	Root mean squared error
RTS	Rauch–Tung–Striebel
RTSS	Rauch–Tung–Striebel smoother
SDE	Stochastic differential equation
SIR	Sequential importance resampling
SIR-PS	Sequential importance resampling particle smoother
SIS	Sequential importance sampling
SLDS	Switching linear dynamic system
SLF	Statistically linearized filter
SLRTSS	Statistically linearized Rauch–Tung–Striebel smoother
SMC	Sequential Monte Carlo
TVAR	Time-varying autoregressive (model)
UKF	Unscented Kalman filter
UPF	Unscented particle filter
URTSS	Unscented Rauch–Tung–Striebel smoother
UT	Unscented transform

1

What are Bayesian filtering and smoothing?

The term *optimal filtering* traditionally refers to a class of methods that can be used for estimating the state of a time-varying system which is indirectly observed through noisy measurements. The term *optimal* in this context refers to statistical optimality. *Bayesian filtering* refers to the Bayesian way of formulating optimal filtering. In this book we use these terms interchangeably and always mean Bayesian filtering.

In optimal, Bayesian, and Bayesian optimal filtering the *state* of the system refers to the collection of dynamic variables such as position, velocity, orientation, and angular velocity, which fully describe the system. The *noise* in the measurements means that they are uncertain; even if we knew the true system state the measurements would not be deterministic functions of the state, but would have a distribution of possible values. The time evolution of the state is modeled as a dynamic system which is perturbed by a certain *process noise*. This noise is used for modeling the uncertainties in the system dynamics. In most cases the system is not truly stochastic, but stochasticity is used for representing the model uncertainties.

Bayesian smoothing (or optimal smoothing) is often considered to be a class of methods within the field of Bayesian filtering. While Bayesian filters in their basic form only compute estimates of the current state of the system given the history of measurements, Bayesian smoothers can be used to reconstruct states that happened before the current time. Although the term *smoothing* is sometimes used in a more general sense for methods which generate a smooth (as opposed to rough) representation of data, in the context of Bayesian filtering the term (Bayesian) smoothing has this more definite meaning.

1.1 Applications of Bayesian filtering and smoothing

Phenomena which can be modeled as time-varying systems of the above type are very common in engineering applications. This kind of model

can be found, for example, in navigation, aerospace engineering, space engineering, remote surveillance, telecommunications, physics, audio signal processing, control engineering, finance, and many other fields. Examples of such applications are the following.

- *Global positioning system (GPS)* (Kaplan, 1996) is a widely used satellite navigation system, where the GPS receiver unit measures arrival times of signals from several GPS satellites and computes its position based on these measurements (see Figure 1.1). The GPS receiver typically uses an extended Kalman filter (EKF) or some other optimal filtering algorithm[1] for computing the current position and velocity such that the measurements and the assumed dynamics (laws of physics) are taken into account. Also the ephemeris information, which is the satellite reference information transmitted from the satellites to the GPS receivers, is typically generated using optimal filters.

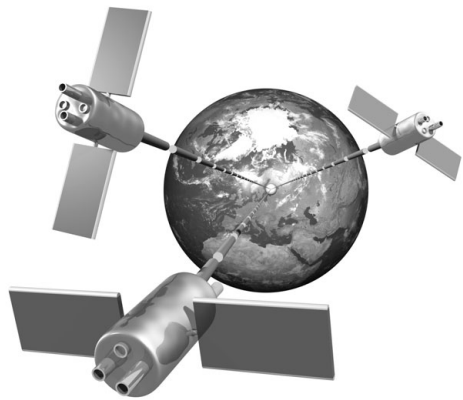


Figure 1.1 In the GPS system, the measurements are time delays of satellite signals and the optimal filter (e.g., extended Kalman filter, EKF) computes the position and the accurate time.

- *Target tracking* (Bar-Shalom et al., 2001; Crassidis and Junkins, 2004; Challa et al., 2011) refers to the methodology where a set of sensors such as active or passive radars, radio frequency sensors, acoustic arrays,

[1] Strictly speaking, the EKF is only an approximate optimal filtering algorithm, because it uses a Taylor series based Gaussian approximation to the non-Gaussian optimal filtering solution.

infrared sensors, and other types of sensors are used for determining the position and velocity of a remote target (see Figure 1.2). When this tracking is done continuously in time, the dynamics of the target and measurements from the different sensors are most naturally combined using an optimal filter or smoother. The target in this (single) target tracking case can be, for example, a robot, a satellite, a car or an airplane.

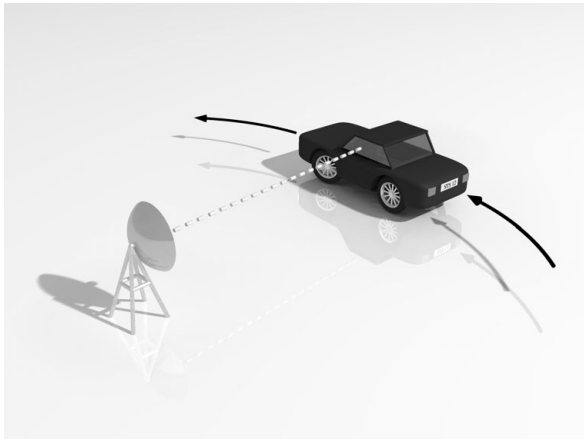


Figure 1.2 In target tracking, a sensor (e.g., radar) generates measurements (e.g., angle and distance measurements) of the target, and the purpose is to determine the target trajectory.

- *Multiple target tracking* (Bar-Shalom and Li, 1995; Blackman and Popoli, 1999; Stone et al., 1999; Särkkä et al., 2007b) systems are used for remote surveillance in the cases where there are multiple targets moving at the same time in the same geographical area (see Figure 1.3). This introduces the concept of data association (which measurement was from which target?) and the problem of estimating the number of targets. Multiple target tracking systems are typically used in remote surveillance for military purposes, but their civil applications are, for example, monitoring of car tunnels, automatic alarm systems, and people tracking in buildings.
- *Inertial navigation* (Titterton and Weston, 1997; Grewal et al., 2001) uses inertial sensors such as accelerometers and gyroscopes for computing the position and velocity of a device such as a car, an airplane, or a missile. When the inaccuracies in sensor measurements are taken into account the natural way of computing the estimates is by using an op-

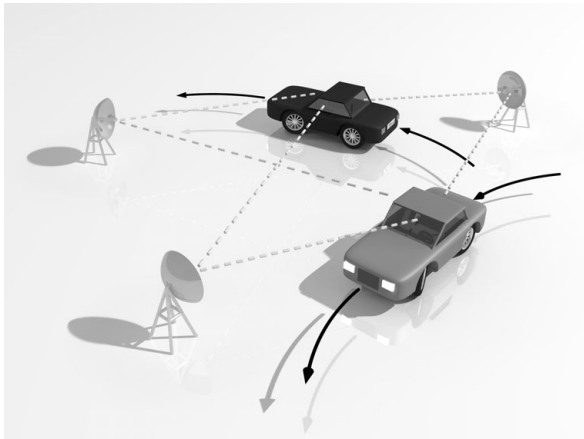


Figure 1.3 In multiple target tracking the data association problem has to be solved, because it is impossible to know without any additional information which target produced which measurement.

timal filter or smoother. Also, in sensor calibration, which is typically done in a time-varying environment, optimal filters and smoothers can be applied.

- *Integrated inertial navigation* (Grewal et al., 2001; Bar-Shalom et al., 2001) combines the good sides of unbiased but inaccurate sensors, such as altimeters and landmark trackers, and biased but locally accurate inertial sensors. A combination of these different sources of information is most naturally performed using an optimal filter such as the extended Kalman filter. This kind of approach was used, for example, in the guidance system of the Apollo 11 lunar module (Eagle), which landed on the moon in 1969.
- *GPS/INS navigation* (Grewal et al., 2001; Bar-Shalom et al., 2001) is a form of integrated inertial navigation where the inertial navigation system (INS) is combined with a GPS receiver unit. In a GPS/INS navigation system the short term fluctuations of the GPS can be compensated by the inertial sensors and the inertial sensor biases can be compensated by the GPS receiver. An additional advantage of this approach is that it is possible to temporarily switch to pure inertial navigation when the GPS receiver is unable to compute its position (i.e., has no fix) for some reason. This happens, for example, indoors, in tunnels and in other cases

when there is no direct line-of-sight between the GPS receiver and the satellites.

- *Brain imaging* methods such as electroencephalography (EEG), magnetoencephalography (MEG), parallel functional magnetic resonance imaging (fMRI) and diffuse optical tomography (DOT) (see Figure 1.4) are based on reconstruction of the source field in the brain from noisy sensor data by using minimum norm estimates (MNE) and its variants (Hauk, 2004; Tarantola, 2004; Kaipio and Somersalo, 2005; Lin et al., 2006). The minimum norm solution can also be interpreted in the Bayesian sense as a problem of estimating the field with certain prior structure from Gaussian observations. With that interpretation the estimation problem becomes equivalent to a *statistical inversion* or generalized *Gaussian process regression problem* (Tarantola, 2004; Kaipio and Somersalo, 2005; Rasmussen and Williams, 2006; Särkkä, 2011). Including dynamical priors then leads to a linear or non-linear spatio-temporal estimation problem, which can be solved with Kalman filters and smoothers (see Hiltunen et al., 2011; Särkkä et al., 2012b). The same can be done in inversion based approaches to parallel fMRI such as inverse imaging (InI, Lin et al., 2006).

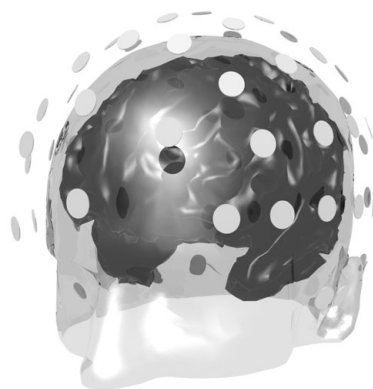


Figure 1.4 Brain imaging methods such as EEG and MEG are based on estimating the state of the brain from sensor readings. In dynamic case the related inversion problem can be solved with an optimal filter or smoother.

- *Spread of infectious diseases* (Keeling and Rohani, 2007) can often be modeled as differential equations for the number of susceptible, infected, and recovered/dead individuals. When uncertainties are introduced into the dynamic equations, and when the measurements are not perfect, the estimation of the spread of the disease can be formulated as an optimal filtering problem (see, e.g., Särkkä and Sottinen, 2008).
- *Biological processes* (Murray, 1993) such as population growth, predator–prey models, and several other dynamic processes in biology can also be modeled as (stochastic) differential equations. Estimation of the states of these processes from inaccurate measurements can be formulated as an optimal filtering and smoothing problem.
- *Telecommunications* is also a field where optimal filters are traditionally used. For example, optimal receivers, signal detectors, and phase locked loops can be interpreted to contain optimal filters (Van Trees, 1968, 1971; Proakis, 2001) as components. Also the celebrated Viterbi algorithm (Viterbi, 1967) can be seen as a method for computing the maximum a posteriori (MAP) Bayesian smoothing solution for the underlying hidden Markov model (HMM).
- *Audio signal processing* applications such as audio restoration (Godsill and Rayner, 1998) and audio signal enhancement (Fong et al., 2002) often use TVAR (time-varying autoregressive) models as the underlying audio signal models. These kinds of model can be efficiently estimated using optimal filters and smoothers.
- *Stochastic optimal control* (Maybeck, 1982a; Stengel, 1994) considers control of time-varying stochastic systems. Stochastic controllers can typically be found in, for example, airplanes, cars, and rockets. Optimal, in addition to the statistical optimality, means that the control signal is constructed to minimize a performance cost, such as the expected time to reach a predefined state, the amount of fuel consumed, or the average distance from a desired position trajectory. When the state of the system is observed through a set of sensors, as it usually is, optimal filters are needed for reconstructing the state from them.
- *Learning systems* or adaptive systems can often be mathematically formulated in terms of optimal filters and smoothers (Haykin, 2001) and they have a close relationship with Bayesian non-parametric modeling, machine learning, and neural network modeling (Bishop, 2006). Methods similar to the data association methods in multiple target tracking are also applicable to on-line adaptive classification (Andrieu et al., 2002). The connection between Gaussian process regression (Rasmussen and Williams, 2006) and optimal filtering has also been recently discussed

in Särkkä et al. (2007a), Hartikainen and Särkkä (2010) and Särkkä and Hartikainen (2012).

- *Physical systems* which are time-varying and measured through non-ideal sensors can sometimes be formulated as stochastic state space models, and the time evolution of the system can be estimated using optimal filters (Kaipio and Somersalo, 2005). These kinds of problem are often called *inverse problems* (Tarantola, 2004), and optimal filters and smoothers can be seen as the Bayesian solutions to time-varying inverse problems.

1.2 Origins of Bayesian filtering and smoothing

The roots of Bayesian analysis of time-dependent behavior are in the field of optimal linear filtering. The idea of constructing mathematically optimal recursive estimators was first presented for linear systems due to their mathematical simplicity, and the most natural optimality criterion in both the mathematical and modeling points of view was the least squares optimality. For linear systems the optimal Bayesian solution (with minimum mean squared error, MMSE, loss) coincides with the least squares solution, that is, the optimal least squares solution is exactly the posterior mean.

The history of optimal filtering starts from the *Wiener filter* (Wiener, 1950), which is a frequency domain solution to the problem of least squares optimal filtering of stationary Gaussian signals. The Wiener filter is still important in communication applications (Proakis, 2001), digital signal processing (Hayes, 1996) and image processing (Gonzalez and Woods, 2008). The disadvantage of the Wiener filter is that it can only be applied to stationary signals.

The success of optimal linear filtering in engineering applications is mostly due to the seminal article of Kalman (1960b), which describes the recursive solution to the optimal discrete-time (sampled) linear filtering problem. One reason for the success is that the *Kalman filter* can be understood and applied with very much lighter mathematical machinery than the Wiener filter. Also, despite its mathematical simplicity and generality, the Kalman filter (or actually the Kalman–Bucy filter; Kalman and Bucy, 1961) contains the Wiener filter as its limiting special case.

In the early stages of its history, the Kalman filter was soon discovered to belong to the class of Bayesian filters (Ho and Lee, 1964; Lee, 1964; Jazwinski, 1966, 1970). The corresponding Bayesian smoothers (Rauch, 1963; Rauch et al., 1965; Leondes et al., 1970) were also developed soon after the invention of the Kalman filter. An interesting historical detail is

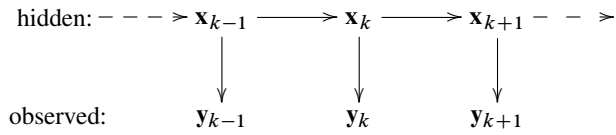


Figure 1.5 In optimal filtering and smoothing problems a sequence of hidden states $\mathbf{x}_k$ is indirectly observed through noisy measurements $\mathbf{y}_k$.

that while Kalman and Bucy were formulating the linear theory in the United States, Stratonovich was doing the pioneering work on the probabilistic (Bayesian) approach in Russia (Stratonovich, 1968; Jazwinski, 1970).

As discussed in the book of West and Harrison (1997), in the 1960s, Kalman filter like recursive estimators were also used in the Bayesian community and it is not clear whether the theory of Kalman filtering or the theory of *dynamic linear models* (DLM) came first. Although these theories were originally derived from slightly different starting points, they are equivalent. Because of the Kalman filter's useful connection to the theory and history of stochastic optimal control, this book approaches the Bayesian filtering problem from the Kalman filtering point of view.

Although the original derivation of the *Kalman filter* was based on the least squares approach, the same equations can be derived from pure probabilistic Bayesian analysis. The Bayesian analysis of Kalman filtering is well covered in the classical book of Jazwinski (1970) and more recently in the book of Bar-Shalom et al. (2001). Kalman filtering, mostly because of its least squares interpretation, has widely been used in stochastic optimal control. A practical reason for this is that the inventor of the Kalman filter, Rudolph E. Kalman, has also made several contributions (Kalman, 1960a) to the theory of *linear quadratic Gaussian* (LQG) regulators, which are fundamental tools of stochastic optimal control (Stengel, 1994; Maybeck, 1982a).

1.3 Optimal filtering and smoothing as Bayesian inference

In mathematical terms, optimal filtering and smoothing are considered to be statistical inversion problems, where the unknown quantity is a vector valued time series $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots\}$ which is observed through a set of

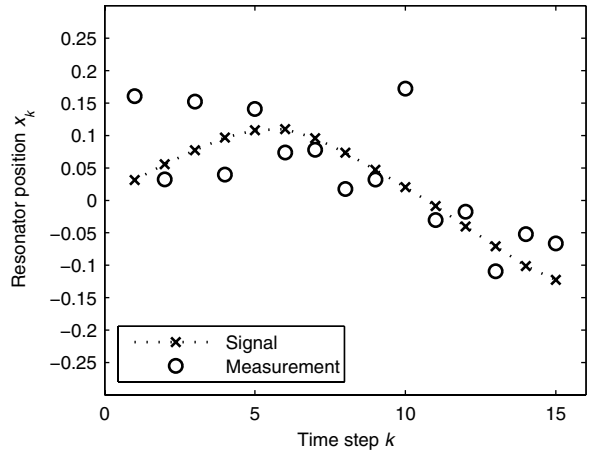


Figure 1.6 An example of time series, which models a discrete-time resonator. The actual resonator state (signal) is hidden and only observed through the noisy measurements.

noisy measurements $\{\mathbf{y}_1, \mathbf{y}_2, \ldots\}$ as illustrated in Figure 1.5. An example of this kind of time series is shown in Figure 1.6. The process shown is a noisy resonator with a known angular velocity. The state $\mathbf{x}_k = (x_k \ \dot{x}_k)^\mathsf{T}$ is two dimensional and consists of the position of the resonator x_k and its time derivative $\dot{x}_k$. The measurements y_k are scalar observations of the resonator position (signal) and they are corrupted by measurement noise.

The purpose of the *statistical inversion* at hand is to estimate the hidden states $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \ldots, \mathbf{x}_T\}$ from the observed measurements $\mathbf{y}_{1:T} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$, which means that in the Bayesian sense we want to compute the joint *posterior distribution* of all the states given all the measurements. In principle, this can be done by a straightforward application of Bayes' rule

$$p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T})\, p(\mathbf{x}_{0:T})}{p(\mathbf{y}_{1:T})}, \tag{1.1}$$

where

- $p(\mathbf{x}_{0:T})$, is the *prior distribution* defined by the dynamic model,
- $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T})$ is the likelihood model for the measurements,

- $p(\mathbf{y}_{1:T})$ is the normalization constant defined as

$$p(\mathbf{y}_{1:T}) = \int p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T})\, p(\mathbf{x}_{0:T})\, \mathrm{d}\mathbf{x}_{0:T}. \tag{1.2}$$

Unfortunately, this full posterior formulation has the serious disadvantage that each time we obtain a new measurement, the full posterior distribution would have to be recomputed. This is particularly a problem in dynamic estimation (which is exactly the problem we are solving here!), where measurements are typically obtained one at a time and we would want to compute the best possible estimate after each measurement. When the number of time steps increases, the dimensionality of the full posterior distribution also increases, which means that the computational complexity of a single time step increases. Thus eventually the computations will become intractable, no matter how much computational power is available. Without additional information or restrictive approximations, there is no way of getting over this problem in the full posterior computation.

However, the above problem only arises when we want to compute the *full* posterior distribution of the states at each time step. If we are willing to relax this a bit and be satisfied with selected marginal distributions of the states, the computations become an order of magnitude lighter. To achieve this, we also need to restrict the class of dynamic models to probabilistic Markov sequences, which is not as restrictive as it may at first seem. The model for the states and measurements will be assumed to be of the following type.

- **An initial distribution** specifies the *prior probability distribution* $p(\mathbf{x}_0)$ of the hidden state $\mathbf{x}_0$ at the initial time step $k = 0$.
- **A dynamic model** describes the system dynamics and its uncertainties as a *Markov sequence*, defined in terms of the transition probability distribution $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$.
- **A measurement model** describes how the measurement $\mathbf{y}_k$ depends on the current state $\mathbf{x}_k$. This dependence is modeled by specifying the conditional probability distribution of the measurement given the state, which is denoted as $p(\mathbf{y}_k \mid \mathbf{x}_k)$.

Thus a general probabilistic *state space model* is usually written in the following form:

$$\begin{aligned} \mathbf{x}_0 &\sim p(\mathbf{x}_0), \\ \mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}), \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k). \end{aligned} \tag{1.3}$$

Because computing the full joint distribution of the states at all time steps is computationally very inefficient and unnecessary in real-time applications, in *Bayesian filtering and smoothing* the following marginal distributions are considered instead (see Figure 1.7).

- *Filtering distributions* computed by the *Bayesian filter* are the marginal distributions of *the current state* $\mathbf{x}_k$ given *the current and previous measurements* $\mathbf{y}_{1:k} = \{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}), \qquad k = 1, \ldots, T. \tag{1.4}$$

  The result of applying the Bayesian filter to the resonator time series in Figure 1.6 is shown in Figure 1.8.
- *Prediction distributions* which can be computed with the *prediction step of the Bayesian filter* are the marginal distributions of the *future state* $\mathbf{x}_{k+n}$, n steps after the current time step:

$$p(\mathbf{x}_{k+n} \mid \mathbf{y}_{1:k}), \qquad k = 1, \ldots, T, \quad n = 1, 2, \ldots. \tag{1.5}$$

- *Smoothing distributions* computed by the *Bayesian smoother* are the marginal distributions of the state $\mathbf{x}_k$ given a certain interval $\mathbf{y}_{1:T} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ of measurements with $T > k$:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}), \qquad k = 1, \ldots, T. \tag{1.6}$$

  The result of applying the Bayesian smoother to the resonator time series is shown in Figure 1.9.

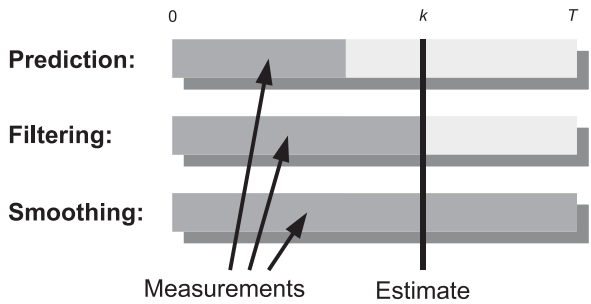


Figure 1.7 State estimation problems can be divided into optimal prediction, filtering, and smoothing depending on the time span of the measurements available with respect to the time of the estimated state.

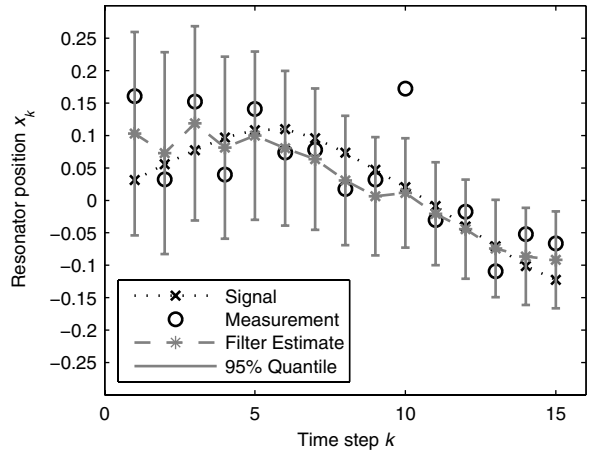


Figure 1.8 The result of computing the filtering distributions for the discrete-time resonator model. The *estimates* are the means of the filtering distributions and the quantiles are the 95% quantiles of the filtering distributions.

Computing filtering, prediction, and smoothing distributions require only a constant number of computations per time step, and thus the problem of processing arbitrarily long time series is solved.

1.4 Algorithms for Bayesian filtering and smoothing

There exist a few classes of filtering and smoothing problems which have closed form solutions.

- *The Kalman filter* (KF) is a closed form solution to the linear Gaussian filtering problem. Due to linear Gaussian model assumptions the posterior distribution is exactly Gaussian and no numerical approximations are needed.
- *The Rauch–Tung–Striebel smoother* (RTSS) is the corresponding closed form smoother for linear Gaussian state space models.
- *Grid filters and smoothers* are solutions to Markov models with finite state spaces.

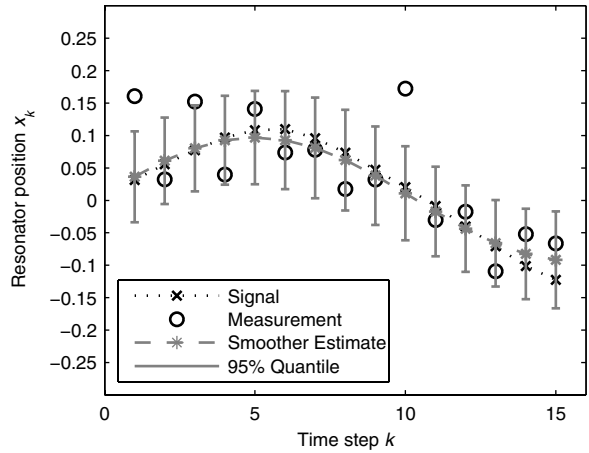


Figure 1.9 The result of computing the smoothing distributions for the discrete-time resonator model. The *estimates* are the means of the smoothing distributions and the quantiles are the 95% quantiles of the smoothing distributions.

But because the Bayesian optimal filtering and smoothing equations are generally computationally intractable, many kinds of numerical approximation methods have been developed, for example:

- *The extended Kalman filter* (EKF) approximates the non-linear and non-Gaussian measurement and dynamic models by linearization, that is, by forming a Taylor series expansion at the nominal (or maximum a posteriori, MAP) solution. This results in a Gaussian approximation to the filtering distribution.
- *The extended Rauch–Tung–Striebel smoother* (ERTSS) is the approximate non-linear smoothing algorithm corresponding to EKF.
- *The unscented Kalman filter* (UKF) approximates the propagation of densities through the non-linearities of measurement and noise processes using the *unscented transform*. This also results in a Gaussian approximation.
- *The unscented Rauch–Tung–Striebel smoother* (URTSS) is the approximate non-linear smoothing algorithm corresponding to UKF.
- *Sequential Monte Carlo methods* or *particle filters and smoothers* represent the posterior distribution as a weighted set of Monte Carlo samples.

- *The unscented particle filter* (UPF) and *local linearization* based particle filtering methods use UKFs and EKFs, respectively, for approximating the optimal importance distributions in particle filters.
- *Rao–Blackwellized particle filters and smoothers* use closed form integration (e.g., Kalman filters and RTS smoothers) for some of the state variables and Monte Carlo integration for others.
- *Grid based approximation methods* approximate the filtering and smoothing distributions as discrete distributions on a finite grid.
- *Other methods* also exist, for example, based on Gaussian mixtures, series expansions, describing functions, basis function expansions, exponential family of distributions, variational Bayesian methods, and batch Monte Carlo (e.g., Markov chain Monte Carlo, MCMC, methods).

1.5 Parameter estimation

In state space models of dynamic systems, there are often *unknown or uncertain parameters* $\boldsymbol{\theta}$ which should be estimated along with the state itself. For example, in a stochastic resonator model, the frequency of the resonator might be unknown. Also the noise variances might be only known approximately or they can be completely unknown. Although, formally, we can always augment unknown parameters as part of the state, in practice it is often useful to consider parameter estimation separately.

In a Bayesian setting, the proper way to estimate the parameters is by setting a prior distribution on the parameters $p(\boldsymbol{\theta})$ and treating them as additional random variables in the model. When unknown parameters are present, the state space model in Equation (1.3) becomes

$$\begin{aligned} \boldsymbol{\theta} &\sim p(\boldsymbol{\theta}), \\ \mathbf{x}_0 &\sim p(\mathbf{x}_0 \mid \boldsymbol{\theta}), \\ \mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}), \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}). \end{aligned} \tag{1.7}$$

The full Bayesian solution to this problem would require the computation of the full *joint posterior distribution of states and parameters* $p(\mathbf{x}_{0:T}, \boldsymbol{\theta} \mid \mathbf{y}_{1:T})$. Unfortunately, computing this joint posterior of the states and parameters is even harder than computation of the joint distribution of states alone, and thus this task is intractable.

Fortunately, when run with fixed parameters $\boldsymbol{\theta}$, the Bayesian filter algorithm produces the sequence of distributions $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})$ for $k = 1, \ldots, T$ as side products. Once we have these, we can form the *marginal*

posterior distribution of parameters as follows:

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{k=1}^{T} p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}), \tag{1.8}$$

where we have denoted $p(\mathbf{y}_1 \mid \mathbf{y}_{1:0}, \boldsymbol{\theta}) \triangleq p(\mathbf{y}_1 \mid \boldsymbol{\theta})$ for notational convenience. When combined with the smoothing distributions, we can form all the marginal joint distributions of states and parameters as follows:

$$p(\mathbf{x}_k, \boldsymbol{\theta} \mid \mathbf{y}_{1:T}) = p(\mathbf{x}_k \mid \mathbf{y}_{1:T}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}), \tag{1.9}$$

for $k = 1, \ldots, T$, where $p(\mathbf{x}_k \mid \mathbf{y}_{1:T}, \boldsymbol{\theta})$ is the smoothing distribution of the states with fixed model parameters $\boldsymbol{\theta}$. However, we cannot compute the full joint posterior distribution of states and parameters, which is the price of only using a constant number of computations per time step.

Although here we use the term *parameter estimation*, it might sometimes be the case that we are not actually interested in the values of the parameters as such, but we just do not know the values of them. In that case the proper Bayesian approach is to *integrate out* the parameters. For example, to compute the smoothing distributions in the presence of unknown parameters we can integrate out the parameters from the joint distribution in Equation (1.9):

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) &= \int p(\mathbf{x}_k, \boldsymbol{\theta} \mid \mathbf{y}_{1:T})\, \mathrm{d}\boldsymbol{\theta} \\ &= \int p(\mathbf{x}_k \mid \mathbf{y}_{1:T}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T})\, \mathrm{d}\boldsymbol{\theta}. \end{aligned} \tag{1.10}$$

Many of the Bayesian methods for parameter estimation indeed allow this to be done (approximately). For example, by using the parameter samples produced by a Markov chain Monte Carlo (MCMC) method, it is possible to form a Monte Carlo approximation to the above integral.

1.6 Exercises

1.1 Find the seminal article of Kalman (1960b) from the Internet (or, e.g., from a library) and investigate the orthogonal projections approach that is taken in the article. How would you generalize the approach to the non-linear/non-Gaussian case? Is it possible?

1.2 An alternative to Bayesian estimation would be to formulate the state estimation problem as maximum likelihood (ML) estimation. This would amount

to estimating the state sequence as the ML-estimate

$$\hat{\mathbf{x}}_{0:T} = \arg\max_{\mathbf{x}_{0:T}} p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T}). \tag{1.11}$$

Do you see any problem with this approach? *Hint:* where is the dynamic model?

1.3 Assume that in an electronics shop the salesperson decides to give you a chance to win a brand new GPS receiver. He lets you choose one of three packages of which one contains the GPS receiver and two others are empty. After you have chosen the package, the salesperson opens one of the packages that you have not chosen – and that package turns out to be empty. He gives you a chance to switch to the other yet unopened package. Is it advantageous for you to do that?

2

Bayesian inference

This chapter provides a brief presentation of the philosophical and mathematical foundations of Bayesian inference. The connections to classical statistical inference are also briefly discussed.

2.1 Philosophy of Bayesian inference

The purpose of *Bayesian inference* (Bernardo and Smith, 1994; Gelman et al., 2004) is to provide a mathematical machinery that can be used for modeling systems, where the uncertainties of the system are taken into account and the decisions are made according to rational principles. The tools of this machinery are the probability distributions and the rules of probability calculus.

If we compare the so-called frequentist philosophy of statistical analysis to Bayesian inference the difference is that in Bayesian inference the probability of an event does not mean the proportion of the event in an infinite number of trials, but the uncertainty of the event in a single trial. Because models in Bayesian inference are formulated in terms of probability distributions, the probability axioms and computation rules of the probability theory (see, e.g., Shiryaev, 1996) also apply in Bayesian inference.

2.2 Connection to maximum likelihood estimation

Consider a situation where we know the conditional distribution $p(\mathbf{y}_k \mid \boldsymbol{\theta})$ of conditionally independent random variables (measurements) $\mathbf{y}_{1:T} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$, but the parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ is unknown. The classical statistical method for estimating the parameter is the *maximum likelihood method* (Milton and Arnold, 1995), where we maximize the joint probability of the

measurements, also called the likelihood function

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{k=1}^{T} p(\mathbf{y}_k \mid \boldsymbol{\theta}). \tag{2.1}$$

The maximum of the likelihood function with respect to $\boldsymbol{\theta}$ gives the *maximum likelihood estimate* (ML-estimate)

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{2.2}$$

The difference between the Bayesian inference and the maximum likelihood method is that the starting point of Bayesian inference is to formally consider the parameter $\boldsymbol{\theta}$ as a random variable. Then the posterior distribution of the parameter $\boldsymbol{\theta}$ can be computed by using *Bayes' rule*

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathbf{y}_{1:T})}, \tag{2.3}$$

where $p(\boldsymbol{\theta})$ is the prior distribution which models the prior beliefs on the parameter before we have seen any data, and $p(\mathbf{y}_{1:T})$ is a normalization term which is independent of the parameter $\boldsymbol{\theta}$. This normalization constant is often left out and if the measurements $\mathbf{y}_{1:T}$ are conditionally independent given $\boldsymbol{\theta}$, the posterior distribution of the parameter can be written as

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{k=1}^{T} p(\mathbf{y}_k \mid \boldsymbol{\theta}). \tag{2.4}$$

Because we are dealing with a distribution, we might now choose the most probable value of the random variable, the *maximum a posteriori* (MAP) estimate, which is given by the maximum of the posterior distribution. The optimal estimate in the mean squared sense is the posterior mean of the parameter (MMSE-estimate). There are an infinite number of other ways of choosing the point estimate from the distribution and the best way depends on the assumed loss or cost function (or utility function). The ML-estimate can be seen as a MAP-estimate with uniform prior $p(\boldsymbol{\theta}) \propto 1$ on the parameter $\boldsymbol{\theta}$.

We can also interpret Bayesian inference as a convenient method for including regularization terms into maximum likelihood estimation. The basic ML-framework does not have a self-consistent method for including regularization terms or prior information into statistical models. However, this regularization interpretation of Bayesian inference is quite limited, because Bayesian inference is much more than this.

2.3 The building blocks of Bayesian models

The basic blocks of a Bayesian model are the *prior model* containing the preliminary information on the parameter and the *measurement model* determining the stochastic mapping from the parameter to the measurements. Using combination rules, namely Bayes' rule, it is possible to infer an estimate of the parameters from the measurements. The probability distribution of the parameters, conditional on the observed measurements, is called the *posterior distribution* and it is the distribution representing the state of knowledge about the parameters when all the information in the observed measurements and the model is used. The *predictive posterior distribution* is the distribution of new (not yet observed) measurements when all the information in the observed measurements and the model is used.

- **Prior model**
  The prior information consists of subjective experience based beliefs about the possible and impossible parameter values and their relative likelihoods before anything has been observed. The prior distribution is a mathematical representation of this information:

$$p(\boldsymbol{\theta}) = \text{information on parameter } \boldsymbol{\theta} \text{ before seeing any observations.} \tag{2.5}$$

  The lack of prior information can be expressed by using a non-informative prior. The non-informative prior distribution can be selected in various different ways (Gelman et al., 2004).
- **Measurement model**
  Between the true parameters and the measurements there is often a causal, but inaccurate or noisy relationship. This relationship is mathematically modeled using the measurement model:

$$p(\mathbf{y} \mid \boldsymbol{\theta}) = \text{distribution of observation } \mathbf{y} \text{ given the parameters } \boldsymbol{\theta}. \tag{2.6}$$

- **Posterior distribution**
  The posterior distribution is the conditional distribution of the parameters given the observations. It represents the information we have after the measurement $\mathbf{y}$ has been obtained. It can be computed by using Bayes' rule

$$p(\boldsymbol{\theta} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathbf{y})} \propto p(\mathbf{y} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta}), \tag{2.7}$$

where the normalization constant is given as

$$p(\mathbf{y}) = \int p(\mathbf{y} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}. \tag{2.8}$$

In the case of multiple measurements $\mathbf{y}_{1:T}$, if the measurements are conditionally independent, the joint likelihood of all measurements is the product of distributions of individual measurements and the posterior distribution is

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{k=1}^{T} p(\mathbf{y}_k \mid \boldsymbol{\theta}), \tag{2.9}$$

where the normalization term can be computed by integrating the right-hand side over $\boldsymbol{\theta}$. If the random variable is discrete the integration is replaced by summation.

- **Predictive posterior distribution**
  The predictive posterior distribution is the distribution of new measurements $\mathbf{y}_{T+1}$ given the observed measurements

$$p(\mathbf{y}_{T+1} \mid \mathbf{y}_{1:T}) = \int p(\mathbf{y}_{T+1} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T})\, \mathrm{d}\boldsymbol{\theta}. \tag{2.10}$$

  Thus, after obtaining the measurements $\mathbf{y}_{1:T}$ the predictive posterior distribution can be used for computing the probability distribution for measurement index $T + 1$ which has not yet been observed.

In the case of tracking, we could imagine that the parameter is the sequence of dynamic states of a target, where the state contains the position and velocity. The measurements could be, for example, noisy distance and direction measurements produced by a radar. In this book we will divide the parameters into two classes: the dynamic state of the system and the static parameters of the model. But from the Bayesian estimation point of view both the states and static parameters are unknown (random) parameters of the system.

2.4 Bayesian point estimates

Distributions alone have no use in many practical applications; we need finite-dimensional summaries (point estimates). This selection of a point based on observed values of random variables is a statistical decision, and therefore this selection procedure is most naturally formulated in terms of *statistical decision theory* (Berger, 1985; Bernardo and Smith, 1994; Raiffa and Schlaifer, 2000).

Definition 2.1 (Loss function) *A loss function or cost function* $C(\boldsymbol{\theta}, \mathbf{a})$ *is a scalar valued function which determines the loss of taking the* action $\mathbf{a}$ *when the true parameter value is* $\boldsymbol{\theta}$*. The action (or control) is the statistical decision to be made based on the currently available information.*

Instead of loss functions it is also possible to work with utility functions $U(\boldsymbol{\theta}, \mathbf{a})$, which determine the reward from taking the action $\mathbf{a}$ with parameter values $\boldsymbol{\theta}$. Loss functions can be converted to utility functions and vice versa by defining $U(\boldsymbol{\theta}, \mathbf{a}) = -C(\boldsymbol{\theta}, \mathbf{a})$.

If the value of the parameter $\boldsymbol{\theta}$ is not known, but the knowledge of the parameter can be expressed in terms of the posterior distribution $p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T})$, then the natural choice is the action which gives the *minimum (maximum) of the expected loss (utility)* (Berger, 1985)

$$\mathrm{E}[C(\boldsymbol{\theta}, \mathbf{a}) \mid \mathbf{y}_{1:T}] = \int C(\boldsymbol{\theta}, \mathbf{a})\, p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T})\, \mathrm{d}\boldsymbol{\theta}. \tag{2.11}$$

Commonly used loss functions are the following.

- *Quadratic error loss*. If the loss function is quadratic

$$C(\boldsymbol{\theta}, \mathbf{a}) = (\boldsymbol{\theta} - \mathbf{a})^{\mathsf{T}}(\boldsymbol{\theta} - \mathbf{a}), \tag{2.12}$$

  then the optimal choice $\mathbf{a}_{\mathrm{o}}$ is the *mean* of the posterior distribution of $\boldsymbol{\theta}$

$$\mathbf{a}_{\mathrm{o}} = \int \boldsymbol{\theta}\, p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T})\, \mathrm{d}\boldsymbol{\theta}. \tag{2.13}$$

  This posterior mean based estimate is often called the *minimum mean squared error (MMSE)* estimate of the parameter $\boldsymbol{\theta}$. The quadratic loss is the most commonly used loss function, because it is easy to handle mathematically and because in the case of Gaussian posterior distribution the MAP estimate and the median coincide with the posterior mean.
- *Absolute error loss*. The loss function of the form

$$C(\boldsymbol{\theta}, \mathbf{a}) = \sum_i |\theta_i - a_i| \tag{2.14}$$

  is called an absolute error loss and in this case the optimal choice is the *median* of the distribution (the medians of the marginal distributions in the multi-dimensional case).
- *0–1 loss*. If the loss function is of the form

$$C(\boldsymbol{\theta}, \mathbf{a}) = -\delta(\mathbf{a} - \boldsymbol{\theta}), \tag{2.15}$$

  where $\delta(\cdot)$ is the Dirac's delta function, then the optimal choice is the maximum (mode) of the posterior distribution, that is, the *maximum a*

posterior (MAP) estimate of the parameter. If the random variable $\boldsymbol{\theta}$ is discrete the corresponding loss function can be defined as

$$C(\boldsymbol{\theta}, \mathbf{a}) = \begin{cases} 0, & \text{if } \boldsymbol{\theta} = \mathbf{a}, \\ 1, & \text{if } \boldsymbol{\theta} \neq \mathbf{a}. \end{cases} \tag{2.16}$$

2.5 Numerical methods

In principle, Bayesian inference provides the equations for computing the posterior distributions and point estimates for any model once the model specification has been set up. However, the practical difficulty is that computation of the integrals involved in the equations can rarely be performed analytically and numerical methods are needed. Here we briefly describe numerical methods which are also applicable in higher-dimensional problems: Gaussian approximations, multi-dimensional quadratures, Monte Carlo methods, and importance sampling.

- *Gaussian approximations* (Gelman et al., 2004) are very common, and in them the posterior distribution is approximated with a Gaussian distribution (see Section A.1)

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \simeq \mathrm{N}(\boldsymbol{\theta} \mid \mathbf{m}, \mathbf{P}). \tag{2.17}$$

  The mean $\mathbf{m}$ and covariance $\mathbf{P}$ of the Gaussian approximation can be computed either by matching the first two moments of the posterior distribution, or by using the mode of the distribution as the approximation of $\mathbf{m}$ and by approximating $\mathbf{P}$ using the curvature of the posterior at the mode. Note that above we have introduced the notation $\simeq$ which here means that the left-hand side is *assumed* to be approximately equal to the right-hand side, even though we know that it will not be true in most practical situations nor can we control the approximation error in any practical way.
- *Multi-dimensional quadrature or cubature integration methods* such as Gauss–Hermite quadrature can also be used if the dimensionality of the integral is moderate. The idea is to deterministically form a representative set of sample points $\{\boldsymbol{\theta}^{(i)} : i = 1, \ldots, N\}$ (sometimes called *sigma points*) and form the approximation of the integral as the weighted average

$$\mathrm{E}[\mathbf{g}(\boldsymbol{\theta}) \mid \mathbf{y}_{1:T}] \approx \sum_{i=1}^{N} W_i \, \mathbf{g}(\boldsymbol{\theta}^{(i)}), \tag{2.18}$$

where the numerical values of the weights W_i are determined by the algorithm. The sample points and weights can be selected, for example, to give exact answers for polynomials up to certain degree or to account for the moments up to certain degree. Above we have used the notation $\approx$ to mean that the expressions are approximately equal in some suitable limit (here $N \to \infty$) or in some verifiable conditions.

- In direct *Monte Carlo methods* a set of N samples from the posterior distribution is randomly drawn

$$\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}), \qquad i = 1, \ldots, N, \tag{2.19}$$

and expectation of any function $\mathbf{g}(\cdot)$ can be then approximated as the sample average

$$\mathrm{E}[\mathbf{g}(\boldsymbol{\theta}) \mid \mathbf{y}_{1:T}] \approx \frac{1}{N} \sum_i \mathbf{g}(\boldsymbol{\theta}^{(i)}). \tag{2.20}$$

Another interpretation of this is that Monte Carlo methods form an approximation of the posterior density of the form

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^{(i)}), \tag{2.21}$$

where $\delta(\cdot)$ is the Dirac delta function. The convergence of Monte Carlo approximation is guaranteed by the *central limit theorem (CLT)* (see, e.g., Liu, 2001) and the error term is, at least in theory, under certain ideal conditions, independent of the dimensionality of $\boldsymbol{\theta}$. The rule of thumb is that the error should decrease like the square root of the number of samples, regardless of the dimensions.

- Efficient methods for generating Monte Carlo samples are the *Markov chain Monte Carlo* (MCMC) methods (see, e.g., Gilks et al., 1996; Liu, 2001; Brooks et al., 2011). In MCMC methods, a Markov chain is constructed such that it has the target distribution as its stationary distribution. By simulating the Markov chain, samples from the target distribution can be generated.
- *Importance sampling* (see, e.g., Liu, 2001) is a simple algorithm for generating *weighted* samples from the target distribution. The difference between this and direct Monte Carlo sampling and MCMC is that each of the particles has an associated weight, which corrects for the difference between the actual target distribution and the approximate importance distribution $\pi(\cdot)$ from which the sample was drawn.

An importance sampling estimate can be formed by drawing N samples from the *importance distribution*

$$\boldsymbol{\theta}^{(i)} \sim \pi(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}), \qquad i = 1, \ldots, N. \tag{2.22}$$

The *importance weights* are then computed as

$$\tilde{w}^{(i)} = \frac{1}{N} \frac{p(\boldsymbol{\theta}^{(i)} \mid \mathbf{y}_{1:T})}{\pi(\boldsymbol{\theta}^{(i)} \mid \mathbf{y}_{1:T})}, \tag{2.23}$$

and the expectation of any function $\mathbf{g}(\cdot)$ can be then approximated as

$$\mathrm{E}[\mathbf{g}(\boldsymbol{\theta}) \mid \mathbf{y}_{1:T}] \approx \sum_{i=1}^{N} \tilde{w}^{(i)}\, \mathbf{g}(\boldsymbol{\theta}^{(i)}), \tag{2.24}$$

or alternatively as

$$\mathrm{E}[\mathbf{g}(\boldsymbol{\theta}) \mid \mathbf{y}_{1:T}] \approx \frac{\sum_{i=1}^{N} \tilde{w}^{(i)}\, \mathbf{g}(\boldsymbol{\theta}^{(i)})}{\sum_{i=1}^{N} \tilde{w}^{(i)}}. \tag{2.25}$$

2.6 Exercises

2.1 Prove that median of distribution $p(\theta)$ minimizes the expected value of the absolute error loss function

$$\mathrm{E}[|\theta - a|] = \int |\theta - a|\, p(\theta)\, \mathrm{d}\theta. \tag{2.26}$$

2.2 Find the optimal point estimate $\mathbf{a}$ which minimizes the expected value of the loss function

$$C(\boldsymbol{\theta}, \mathbf{a}) = (\boldsymbol{\theta} - \mathbf{a})^{\mathsf{T}}\, \mathbf{R}\, (\boldsymbol{\theta} - \mathbf{a}), \tag{2.27}$$

where $\mathbf{R}$ is a positive definite matrix, and the distribution of the parameter is $\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T})$.

2.3 Assume that we have obtained T measurement pairs (x_k, y_k) from the linear regression model

$$y_k = \theta_1\, x_k + \theta_2, \qquad k = 1, 2, \ldots, T. \tag{2.28}$$

The purpose is now to derive estimates of the parameters θ_1 and θ_2 such that the following error is minimized (least squares estimate):

$$E(\theta_1, \theta_2) = \sum_{k=1}^{T} (y_k - \theta_1\, x_k - \theta_2)^2. \tag{2.29}$$

(a) Define $\mathbf{y} = (y_1 \ \ldots \ y_T)^\mathsf{T}$ and $\boldsymbol{\theta} = (\theta_1 \ \theta_2)^\mathsf{T}$. Show that the set of Equations (2.28) can be written in matrix form as

$$\mathbf{y} = \mathbf{X}\,\boldsymbol{\theta},$$

with a suitably defined matrix $\mathbf{X}$.

(b) Write the error function in Equation (2.29) in matrix form in terms of $\mathbf{y}$, $\mathbf{X}$, and $\boldsymbol{\theta}$.

(c) Compute the gradient of the matrix form error function and solve the least squares estimate of the parameter $\boldsymbol{\theta}$ by finding the point where the gradient is zero.

2.4 Assume that in the linear regression model above (Equation (2.28)) we set independent Gaussian priors for the parameters θ_1 and θ_2 as follows:

$$\theta_1 \sim \mathrm{N}(0, \sigma^2),$$
$$\theta_2 \sim \mathrm{N}(0, \sigma^2),$$

where the variance σ^2 is known. The measurements y_k are modeled as

$$y_k = \theta_1 x_k + \theta_2 + \varepsilon_k, \qquad k = 1, 2, \ldots, T,$$

where the terms ε_k are independent Gaussian errors with mean 0 and variance 1, that is, $\varepsilon_k \sim \mathrm{N}(0, 1)$. The values x_k are fixed and known. The posterior distribution can be now written as

$$p(\boldsymbol{\theta} \mid y_1, \ldots, y_T)$$
$$\propto \exp\left(-\frac{1}{2}\sum_{k=1}^{T}(y_k - \theta_1 x_k - \theta_2)^2\right) \exp\left(-\frac{1}{2\sigma^2}\theta_1^2\right) \exp\left(-\frac{1}{2\sigma^2}\theta_2^2\right).$$

The posterior distribution can be seen to be Gaussian and your task is to derive its mean and covariance.

(a) Write the exponent of the posterior distribution in matrix form as in Exercise 2.3 (in terms of $\mathbf{y}$, $\mathbf{X}$, $\boldsymbol{\theta}$, and σ^2).

(b) Because a Gaussian distribution is always symmetric, its mean $\mathbf{m}$ is at the maximum of the distribution. Find the posterior mean by computing the gradient of the exponent and finding where it vanishes.

(c) Find the covariance of the distribution by computing the second derivative matrix (Hessian matrix) $\mathbf{H}$ of the exponent. The posterior covariance is then $\mathbf{P} = -\mathbf{H}^{-1}$ (why?).

(d) What is the resulting posterior distribution? What is the relationship with the least squares estimate in Exercise 2.3?

2.5 Implement an importance sampling based approximation for the Bayesian linear regression problem in the above Exercise 2.4. Use a suitable Gaussian distribution as the importance distribution for the parameters $\boldsymbol{\theta}$. Check that the posterior mean and covariance (approximately) coincide with the exact values computed in Exercise 2.4.

3

Batch and recursive Bayesian estimation

In order to understand the meaning and applicability of Bayesian filtering and its relationship to recursive estimation, it is useful to go through an example where we solve a simple and familiar linear regression problem in a recursive manner. After that we generalize this concept to include a dynamic model in order to illustrate the differences in dynamic and batch estimation.

3.1 Batch linear regression

Consider the *linear regression model*

$$y_k = \theta_1 + \theta_2 t_k + \varepsilon_k, \tag{3.1}$$

where we assume that the measurement noise is zero mean Gaussian with a given variance $\varepsilon_k \sim \mathrm{N}(0, \sigma^2)$ and the prior distribution of the parameters $\boldsymbol{\theta} = (\theta_1 \;\; \theta_2)^\mathsf{T}$ is Gaussian with known mean and covariance $\boldsymbol{\theta} \sim \mathrm{N}(\mathbf{m}_0, \mathbf{P}_0)$. In the classical linear regression problem we want to estimate the parameters $\boldsymbol{\theta}$ from a set of measurement data $\mathcal{D} = \{(t_1, y_1), \ldots, (t_T, y_T)\}$. The measurement data and the true linear function used in simulation are illustrated in Figure 3.1.

In compact *probabilistic notation* the linear regression model can be written as

$$\begin{aligned} p(y_k \mid \boldsymbol{\theta}) &= \mathrm{N}(y_k \mid \mathbf{H}_k \boldsymbol{\theta}, \sigma^2) \\ p(\boldsymbol{\theta}) &= \mathrm{N}(\boldsymbol{\theta} \mid \mathbf{m}_0, \mathbf{P}_0), \end{aligned} \tag{3.2}$$

where we have introduced the row vector $\mathbf{H}_k = (1 \;\; t_k)$ and $\mathrm{N}(\cdot)$ denotes the Gaussian probability density function (see Section A.1). Note that we denote the row vector $\mathbf{H}_k$ in matrix notation, because it generally is a matrix (when the measurements are vector valued) and we want to avoid using different notations for scalar and vector measurements. The likelihood of y_k is conditional on the regressors t_k also (or equivalently $\mathbf{H}_k$), but because

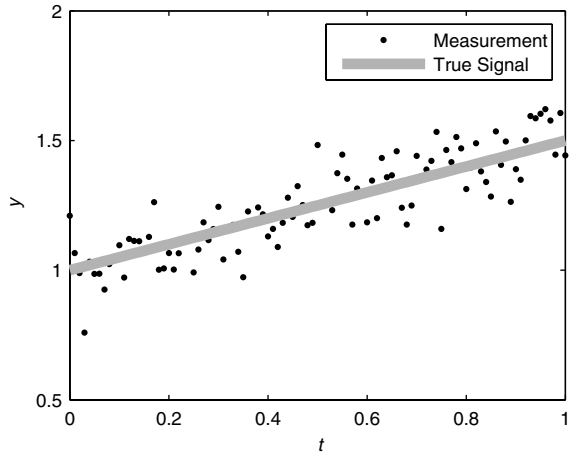


Figure 3.1 The underlying truth and the measurement data in the simple linear regression problem.

the regressors are assumed to be known, to simplify the notation we will not denote this dependence explicitly and from now on this dependence is assumed to be understood from the context.

The *batch solution* to the linear regression problem in Equation (3.2) can be obtained by a straightforward application of Bayes' rule

$$\begin{aligned} p(\boldsymbol{\theta} \mid y_{1:T}) &\propto p(\boldsymbol{\theta}) \prod_{k=1}^{T} p(y_k \mid \boldsymbol{\theta}) \\ &= \mathrm{N}(\boldsymbol{\theta} \mid \mathbf{m}_0, \mathbf{P}_0) \prod_{k=1}^{T} \mathrm{N}(y_k \mid \mathbf{H}_k \boldsymbol{\theta}, \sigma^2). \end{aligned}$$

In the *posterior distribution* above, we assume the conditioning on t_k and $\mathbf{H}_k$, but will not denote it explicitly. Thus the posterior distribution is denoted to be conditional on $y_{1:T}$, and not on the data set $\mathcal{D}$ also containing the regressor values t_k. The reason for this simplification is that the simplified notation will also work in more general filtering problems, where there is no natural way of defining the associated regressor variables.

Because the prior and likelihood are Gaussian, the *posterior distribution* will also be Gaussian:

$$p(\boldsymbol{\theta} \mid y_{1:T}) = \mathrm{N}(\boldsymbol{\theta} \mid \mathbf{m}_T, \mathbf{P}_T). \tag{3.3}$$

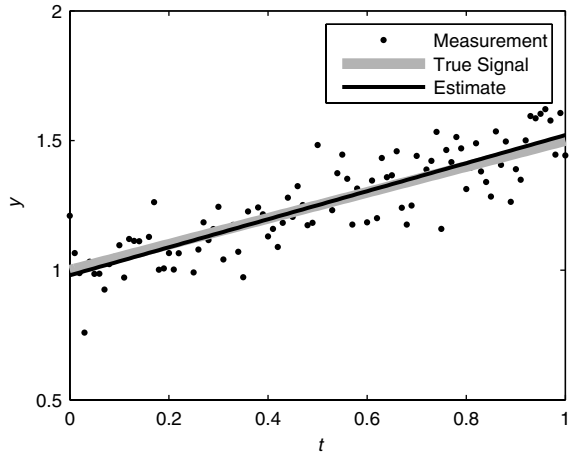


Figure 3.2 The result of simple linear regression with a slight regularization prior used for the regression parameters. For simplicity, the variance was assumed to be known.

The mean and covariance can be obtained by completing the quadratic form in the exponent, which gives:

$$\mathbf{m}_T = \left[\mathbf{P}_0^{-1} + \frac{1}{\sigma^2}\mathbf{H}^\mathsf{T}\mathbf{H}\right]^{-1}\left[\frac{1}{\sigma^2}\mathbf{H}^\mathsf{T}\mathbf{y} + \mathbf{P}_0^{-1}\mathbf{m}_0\right],$$
$$\mathbf{P}_T = \left[\mathbf{P}_0^{-1} + \frac{1}{\sigma^2}\mathbf{H}^\mathsf{T}\mathbf{H}\right]^{-1}, \tag{3.4}$$

where $\mathbf{H}_k = (1\ t_k)$ and

$$\mathbf{H} = \begin{pmatrix}\mathbf{H}_1\\ \vdots \\ \mathbf{H}_T\end{pmatrix} = \begin{pmatrix}1 & t_1\\ \vdots & \vdots \\ 1 & t_T\end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix}y_1\\ \vdots \\ y_T\end{pmatrix}. \tag{3.5}$$

Figure 3.2 shows the result of batch linear regression, where the posterior mean parameter values are used as the linear regression parameters.

3.2 Recursive linear regression

A *recursive solution* to the regression problem (3.2) can be obtained by assuming that we already have obtained the *posterior distribution*

conditioned on the previous measurements $1, \ldots, k-1$ as follows:

$$p(\boldsymbol{\theta} \mid y_{1:k-1}) = \mathrm{N}(\boldsymbol{\theta} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1}).$$

Now assume that we have obtained a new measurement y_k and we want to compute the posterior distribution of $\boldsymbol{\theta}$ given the old measurements $y_{1:k-1}$ *and* the new measurement y_k. According to the model specification the new measurement has the likelihood

$$p(y_k \mid \boldsymbol{\theta}) = \mathrm{N}(y_k \mid \mathbf{H}_k\, \boldsymbol{\theta}, \sigma^2).$$

Using the batch version equations such that we interpret the *previous posterior* as the *prior*, we can calculate the distribution

$$\begin{aligned} p(\boldsymbol{\theta} \mid y_{1:k}) &\propto p(y_k \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid y_{1:k-1}) \\ &\propto \mathrm{N}(\boldsymbol{\theta} \mid \mathbf{m}_k, \mathbf{P}_k), \end{aligned} \tag{3.6}$$

where the Gaussian distribution parameters are

$$\begin{aligned} \mathbf{m}_k &= \left[\mathbf{P}_{k-1}^{-1} + \frac{1}{\sigma^2}\,\mathbf{H}_k^{\mathsf{T}}\,\mathbf{H}_k\right]^{-1} \left[\frac{1}{\sigma^2}\,\mathbf{H}_k^{\mathsf{T}}\, y_k + \mathbf{P}_{k-1}^{-1}\,\mathbf{m}_{k-1}\right], \\ \mathbf{P}_k &= \left[\mathbf{P}_{k-1}^{-1} + \frac{1}{\sigma^2}\,\mathbf{H}_k^{\mathsf{T}}\,\mathbf{H}_k\right]^{-1}. \end{aligned} \tag{3.7}$$

By using the *matrix inversion lemma*, the covariance calculation can be written as

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \mathbf{P}_{k-1}\,\mathbf{H}_k^{\mathsf{T}} \left[\mathbf{H}_k\,\mathbf{P}_{k-1}\,\mathbf{H}_k^{\mathsf{T}} + \sigma^2\right]^{-1} \mathbf{H}_k\,\mathbf{P}_{k-1}.$$

By introducing temporary variables S_k and $\mathbf{K}_k$ the calculation of the mean and covariance can be written in the form

$$\begin{aligned} S_k &= \mathbf{H}_k\,\mathbf{P}_{k-1}\,\mathbf{H}_k^{\mathsf{T}} + \sigma^2, \\ \mathbf{K}_k &= \mathbf{P}_{k-1}\,\mathbf{H}_k^{\mathsf{T}}\,S_k^{-1}, \\ \mathbf{m}_k &= \mathbf{m}_{k-1} + \mathbf{K}_k\,[y_k - \mathbf{H}_k\,\mathbf{m}_{k-1}], \\ \mathbf{P}_k &= \mathbf{P}_{k-1} - \mathbf{K}_k\,S_k\,\mathbf{K}_k^{\mathsf{T}}. \end{aligned} \tag{3.8}$$

Note that $S_k = \mathbf{H}_k\,\mathbf{P}_{k-1}\,\mathbf{H}_k^{\mathsf{T}} + \sigma^2$ is scalar because the measurements are scalar and thus no matrix inversion is required.

The equations above actually are special cases of the Kalman filter update equations. Only the update part of the equations (as opposed to the prediction and update) is required, because the estimated parameters are assumed to be constant, that is, there is no stochastic dynamics model for the parameters $\boldsymbol{\theta}$. Figures 3.3 and 3.4 illustrate the convergence of the means and variances of the parameters during the recursive estimation.

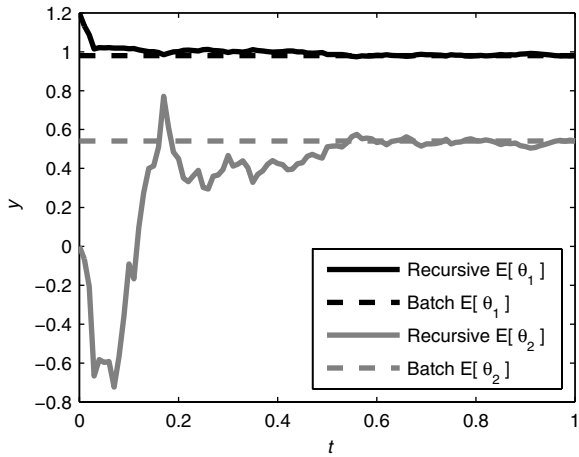


Figure 3.3 Convergence of the recursive linear regression parameters means. The final value is exactly the same as that obtained with batch linear regression.

3.3 Batch versus recursive estimation

In this section we generalize the recursion idea used in the previous section to general probabilistic models. The underlying idea is simply that at each measurement we treat the *posterior distribution of the previous time step* as the *prior for the current time step*. This way we can compute the same solution in a recursive manner that we would obtain by direct application of Bayes' rule to the whole (batch) data set.

The *batch Bayesian solution* to a statistical estimation problem can be formulated as follows.

1 Specify the likelihood model of measurements $p(\mathbf{y}_k \mid \boldsymbol{\theta})$ given the parameter $\boldsymbol{\theta}$. Typically the measurements $\mathbf{y}_k$ are assumed to be conditionally independent such that

$$p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) = \prod_{k=1}^{T} p(\mathbf{y}_k \mid \boldsymbol{\theta}).$$

2 The prior information about the parameter $\boldsymbol{\theta}$ is encoded into the prior distribution $p(\boldsymbol{\theta})$.

3 The observed data set is $\mathcal{D} = \{(t_1, \mathbf{y}_1), \ldots, (t_T, \mathbf{y}_T)\}$, or if we drop the explicit conditioning on t_k, the data is $\mathcal{D} = \mathbf{y}_{1:T}$.

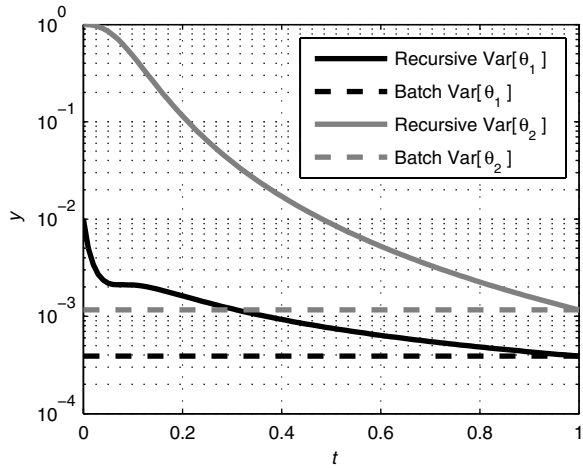


Figure 3.4 Convergence of the variances of linear regression parameters plotted on logarithmic scale. As can be seen, every measurement brings more information and the uncertainty decreases monotonically. The final values are the same as the variances obtained from the batch solution.

4 The batch Bayesian solution to the statistical estimation problem can be computed by applying Bayes' rule:

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) = \frac{1}{Z} p(\boldsymbol{\theta}) \prod_{k=1}^{T} p(\mathbf{y}_k \mid \boldsymbol{\theta}),$$

where Z is the *normalization constant*

$$Z = \int p(\boldsymbol{\theta}) \prod_{k=1}^{T} p(\mathbf{y}_k \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}.$$

For example, the batch solution of the above kind to the linear regression problem (3.2) was given by Equations (3.3) and (3.4).

The *recursive Bayesian solution* to the above statistical estimation problem can be formulated as follows.

1 The distribution of measurements is again modeled by the likelihood function $p(\mathbf{y}_k \mid \boldsymbol{\theta})$ and the measurements are assumed to be conditionally independent.
2 In the beginning of estimation (i.e., at step 0), all the information about the parameter $\boldsymbol{\theta}$ we have is contained in the prior distribution $p(\boldsymbol{\theta})$.

3 The measurements are assumed to be obtained one at a time, first $\mathbf{y}_1$, then $\mathbf{y}_2$ and so on. At each step we use the posterior distribution from the previous time step as the current prior distribution:

$$
\begin{aligned}
p(\boldsymbol{\theta} \mid \mathbf{y}_1) &= \frac{1}{Z_1} p(\mathbf{y}_1 \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}), \\
p(\boldsymbol{\theta} \mid \mathbf{y}_{1:2}) &= \frac{1}{Z_2} p(\mathbf{y}_2 \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}_1), \\
p(\boldsymbol{\theta} \mid \mathbf{y}_{1:3}) &= \frac{1}{Z_3} p(\mathbf{y}_3 \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}_{1:2}), \\
&\vdots \\
p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) &= \frac{1}{Z_T} p(\mathbf{y}_T \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T-1}).
\end{aligned}
$$

It is easy to show that the posterior distribution at the final step above is exactly the posterior distribution obtained by the batch solution. Also, reordering of measurements does not change the final solution.

For example, Equations (3.6) and (3.7) give the one step update rule for the linear regression problem in Equation (3.2).

The recursive formulation of Bayesian estimation has many useful properties.

- The recursive solution can be considered as the *on-line learning* solution to the Bayesian learning problem. That is, the information on the parameters is updated in an on-line manner using new pieces of information as they arrive.
- Because each step in the recursive estimation is a full Bayesian update step, *batch* Bayesian inference is a *special case of recursive* Bayesian inference.
- Due to the sequential nature of estimation we can also model the effect of time on the parameters. That is, we can model what happens to the parameter $\boldsymbol{\theta}$ between the measurements – this is actually the *basis of filtering theory*, where time behavior is modeled by assuming the parameter to be a time-dependent stochastic process $\boldsymbol{\theta}(t)$.

3.4 Drift model for linear regression

Assume that we have a similar linear regression model as in Equation (3.2), but the parameter $\boldsymbol{\theta}$ is allowed to perform a *Gaussian random walk* be-

tween the measurements:

$$\begin{aligned} p(y_k \mid \boldsymbol{\theta}_k) &= \mathrm{N}(y_k \mid \mathbf{H}_k\, \boldsymbol{\theta}_k, \sigma^2), \\ p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1}) &= \mathrm{N}(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1}, \mathbf{Q}), \\ p(\boldsymbol{\theta}_0) &= \mathrm{N}(\boldsymbol{\theta}_0 \mid \mathbf{m}_0, \mathbf{P}_0), \end{aligned} \tag{3.9}$$

where $\mathbf{Q}$ is the covariance of the random walk. Now, given the distribution

$$p(\boldsymbol{\theta}_{k-1} \mid y_{1:k-1}) = \mathrm{N}(\boldsymbol{\theta}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1}),$$

the joint distribution of $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}_{k-1}$ is[1]

$$p(\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k-1} \mid y_{1:k-1}) = p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1})\, p(\boldsymbol{\theta}_{k-1} \mid y_{1:k-1}).$$

The distribution of $\boldsymbol{\theta}_k$ given the measurement history up to time step $k-1$ can be calculated by integrating over $\boldsymbol{\theta}_{k-1}$:

$$p(\boldsymbol{\theta}_k \mid y_{1:k-1}) = \int p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1})\, p(\boldsymbol{\theta}_{k-1} \mid y_{1:k-1})\, \mathrm{d}\boldsymbol{\theta}_{k-1}.$$

This relationship is sometimes called the *Chapman–Kolmogorov equation*. Because $p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1})$ and $p(\boldsymbol{\theta}_{k-1} \mid y_{1:k-1})$ are Gaussian, the result of the marginalization is Gaussian,

$$p(\boldsymbol{\theta}_k \mid y_{1:k-1}) = \mathrm{N}(\boldsymbol{\theta}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-),$$

where

$$\begin{aligned} \mathbf{m}_k^- &= \mathbf{m}_{k-1}, \\ \mathbf{P}_k^- &= \mathbf{P}_{k-1} + \mathbf{Q}. \end{aligned}$$

By using this as the prior distribution for the measurement likelihood $p(y_k \mid \boldsymbol{\theta}_k)$ we get the parameters of the posterior distribution

$$p(\boldsymbol{\theta}_k \mid y_{1:k}) = \mathrm{N}(\boldsymbol{\theta}_k \mid \mathbf{m}_k, \mathbf{P}_k),$$

which are given by Equations (3.8), when $\mathbf{m}_{k-1}$ and $\mathbf{P}_{k-1}$ are replaced by $\mathbf{m}_k^-$ and $\mathbf{P}_k^-$:

$$\begin{aligned} S_k &= \mathbf{H}_k\, \mathbf{P}_k^-\, \mathbf{H}_k^\mathsf{T} + \sigma^2, \\ \mathbf{K}_k &= \mathbf{P}_k^-\, \mathbf{H}_k^\mathsf{T}\, S_k^{-1}, \\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\, [y_k - \mathbf{H}_k\, \mathbf{m}_k^-], \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\, S_k\, \mathbf{K}_k^\mathsf{T}. \end{aligned} \tag{3.10}$$

[1] Note that this formula is correct only for Markovian dynamic models, where $p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1}, y_{1:k-1}) = p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1})$.

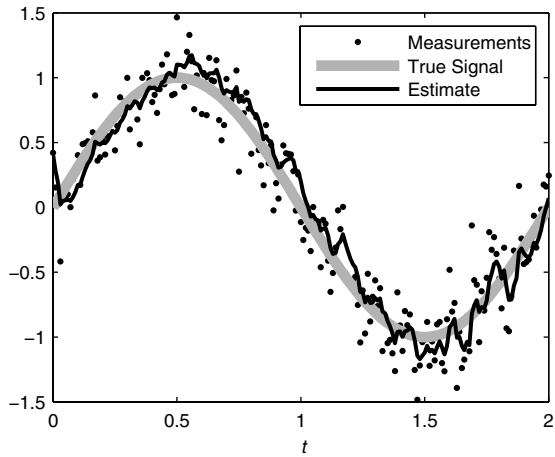


Figure 3.5 Example of tracking a sine signal with linear model with drift, where the parameters are allowed to vary according to the Gaussian random walk model.

This recursive computational algorithm for the time-varying linear regression weights is again a special case of the Kalman filter algorithm. Figure 3.5 shows the result of recursive estimation of a sine signal assuming a small diagonal Gaussian drift model for the parameters.

At this point we change from the *regression notation* used so far into *state space model notation*, which is commonly used in Kalman filtering and related dynamic estimation literature. Because this notation easily causes confusion to people who have got used to regression notation, this point is emphasized.

- In *state space notation* $\mathbf{x}$ means the unknown state of the system, that is, the vector of *unknown parameters in the system*. It is *not* the regressor, covariate or input variable of the system.
- For example, the time-varying linear regression model with drift presented in this section can be transformed into the more standard *state space model notation* by replacing the variable $\boldsymbol{\theta}_k = (\theta_{1,k}\ \theta_{2,k})^\mathsf{T}$ with the variable $\mathbf{x}_k = (x_{1,k}\ x_{2,k})^\mathsf{T}$:

$$\begin{aligned} p(y_k \mid \mathbf{x}_k) &= \mathrm{N}(y_k \mid \mathbf{H}_k\, \mathbf{x}_k, \sigma^2), \\ p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{Q}), \\ p(\mathbf{x}_0) &= \mathrm{N}(\mathbf{x}_0 \mid \mathbf{m}_0, \mathbf{P}_0). \end{aligned} \tag{3.11}$$

From now on, the symbol $\boldsymbol{\theta}$ is reserved for denoting the static parameters of the state space model. Although there is no fundamental difference between states and static parameters of the model (we can always augment the parameters as part of the state), it is useful to treat them separately.

3.5 State space model for linear regression with drift

The linear regression model with drift in the previous section had the disadvantage that the covariates t_k occurred explicitly in the model specification. The problem with this is that when we get more and more measurements, the parameter t_k grows without bound. Thus the conditioning of the problem also gets worse in time. For practical reasons it also would be desirable to have a *time-invariant model*, that is, a model which is not dependent on the absolute time, but only on the relative positions of states and measurements in time.

The alternative state space formulation of the linear regression model with drift, without using explicit covariates, can be done as follows. Let's denote the time difference between consecutive times as $\Delta t_{k-1} = t_k - t_{k-1}$. The idea is that if the underlying phenomenon (signal, state, parameter) x_k was exactly linear, the difference between adjacent time points could be written exactly as

$$x_k - x_{k-1} = \dot{x}\, \Delta t_{k-1}, \tag{3.12}$$

where $\dot{x}$ is the derivative, which is constant in the exactly linear case. The divergence from the exactly linear function can be modeled by assuming that the above equation does not hold exactly, but there is a small *noise term* on the right hand side. The derivative can also be assumed to perform a small *random walk* and thus not be exactly constant. This model can be written as follows:

$$\begin{aligned} x_{1,k} &= x_{1,k-1} + \Delta t_{k-1} x_{2,k-1} + q_{1,k-1}, \\ x_{2,k} &= x_{2,k-1} + q_{2,k-1}, \\ y_k &= x_{1,k} + r_k, \end{aligned} \tag{3.13}$$

where the signal is the first components of the state, $x_{1,k} \triangleq x_k$, and the derivative is the second, $x_{2,k} \triangleq \dot{x}_k$. The noises are $r_k \sim \mathrm{N}(0, \sigma^2)$ and $(q_{2,k-1}, q_{2,k-1}) \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$. The model can also be written in the form

$$\begin{aligned} p(y_k \mid \mathbf{x}_k) &= \mathrm{N}(y_k \mid \mathbf{H}\, \mathbf{x}_k, \sigma^2), \\ p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{A}_{k-1}\, \mathbf{x}_{k-1}, \mathbf{Q}), \end{aligned} \tag{3.14}$$

where

$$\mathbf{A}_{k-1} = \begin{pmatrix} 1 & \Delta t_{k-1} \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} 1 & 0 \end{pmatrix}.$$

With a suitable $\mathbf{Q}$ this model is actually equivalent to model (3.9), but in this formulation we explicitly estimate the state of the signal (point on the regression line) instead of the linear regression parameters.

We could now explicitly derive the recursion equations in the same manner as we did in the previous sections. However, we can also use the *Kalman filter*, which is a readily derived recursive solution to generic linear Gaussian models of the form

$$p(\mathbf{y}_k \mid \mathbf{x}_k) = \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}_k \, \mathbf{x}_k, \mathbf{R}_k),$$
$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{A}_{k-1} \, \mathbf{x}_{k-1}, \mathbf{Q}_{k-1}).$$

Our alternative linear regression model in Equation (3.13) can be seen to be a special case of these models. The *Kalman filter equations* are often expressed as *prediction and update steps* as follows.

1 *Prediction step:*

$$\mathbf{m}_k^- = \mathbf{A}_{k-1} \, \mathbf{m}_{k-1},$$
$$\mathbf{P}_k^- = \mathbf{A}_{k-1} \, \mathbf{P}_{k-1} \, \mathbf{A}_{k-1}^\mathsf{T} + \mathbf{Q}_{k-1}.$$

2 *Update step:*

$$\mathbf{S}_k = \mathbf{H}_k \, \mathbf{P}_k^- \, \mathbf{H}_k^\mathsf{T} + \mathbf{R}_k,$$
$$\mathbf{K}_k = \mathbf{P}_k^- \, \mathbf{H}_k^\mathsf{T} \, \mathbf{S}_k^{-1},$$
$$\mathbf{m}_k = \mathbf{m}_k^- + \mathbf{K}_k \, [\mathbf{y}_k - \mathbf{H}_k \, \mathbf{m}_k^-],$$
$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \, \mathbf{S}_k \, \mathbf{K}_k^\mathsf{T}.$$

The result of tracking the sine signal with Kalman filter is shown in Figure 3.6. All the mean and covariance calculation equations given in this book so far have been special cases of the above equations, including the batch solution to the scalar measurement case (which is a one-step solution). The Kalman filter recursively computes the mean and covariance of the posterior distributions of the form

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k).$$

Note that the estimates of $\mathbf{x}_k$ derived from this distribution are non-anticipative in the sense that they are only conditional on the measurements obtained before and at the time step k. However, after we have obtained the measurements $\mathbf{y}_1, \ldots, \mathbf{y}_k$, we could compute estimates of

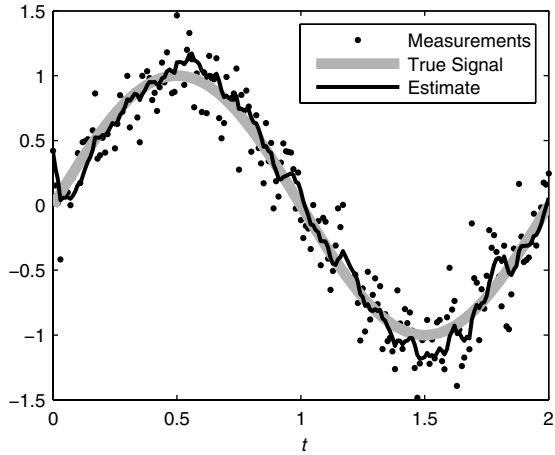


Figure 3.6 Example of tracking a sine signal with Kalman filter using the locally linear state space model. The result differs a bit from the random walk parameter model, because of slightly different choice of process noise. It could be made equivalent if desired.

$\mathbf{x}_{k-1}, \mathbf{x}_{k-2}, \ldots$, which are also conditional to the measurements after the corresponding state time steps. Because more measurements and more information is available for the estimator, these estimates can be expected to be more accurate than the non-anticipative measurements computed by the filter.

The above mentioned problem of computing estimates of the state by conditioning not only on previous measurements, but also on future measurements, is called *Bayesian smoothing* as already mentioned in Section 1.3. The Bayesian smoothing solution to the linear Gaussian state space models is given by the *Rauch–Tung–Striebel (RTS) smoother*. The full Bayesian theory of smoothing will be presented in Chapter 8. The result of tracking the sine signal with the RTS smoother is shown in Figure 3.7.

It is also possible to predict the time behavior of the state in the future that we have not yet measured. This procedure is called *optimal prediction*. Because optimal prediction can always be done by iterating the prediction step of the optimal filter, no specialized algorithms are needed for this. The result of prediction of the future values of the sine signal is shown in Figure 3.8.

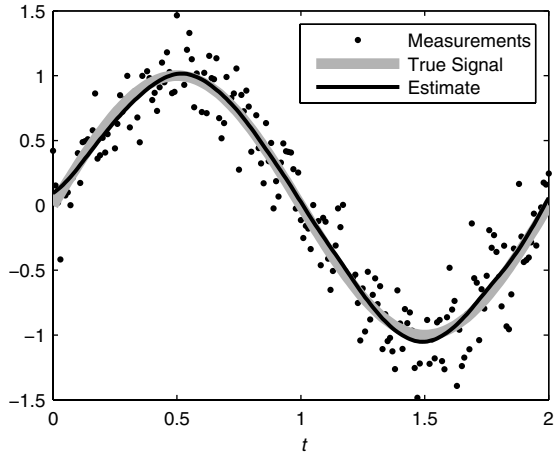


Figure 3.7 Example of tracking a sine signal with the Rauch–Tung–Striebel (RTS) smoother using the locally linear state space model. The result is much "smoother" and more accurate than the result of the Kalman filter.

3.6 Examples of state space models

In the previous sections we saw that a simple one-dimensional linear regression problem can be converted into a state space model which can be solved using the Kalman filter. In an analogous manner, more general linear-in-parameters models can be converted into state space models. This is demonstrated in the following example.

Example 3.1 (Linear-in-parameters regression model I) *Consider the following parametric model, where $g_1(t_k), \ldots, g_d(t_k)$ are some given functions of time t_k:*

$$y_k = w_0 + w_1\, g_1(t_k) + \cdots + w_d\, g_d(t_k) + \varepsilon_k, \tag{3.15}$$

and ε_k is a Gaussian measurement noise. The problem of determining the weights $w_0, \ldots, w_d$ from a set of measurements $\{(t_1, y_1), \ldots, (t_T, y_T)\}$ can be converted into a Kalman filtering problem as follows. If we define $\mathbf{H}_k = (1\ g_1(t_k)\ \cdots\ g_d(t_k))$ and $\mathbf{x}_k = \mathbf{x} = (w_0\ w_1\ \cdots\ w_d)^\mathsf{T}$, we can rewrite the model as a linear Gaussian state space model:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{x}_{k-1}, \\ y_k &= \mathbf{H}_k\, \mathbf{x}_k + \varepsilon_k. \end{aligned} \tag{3.16}$$

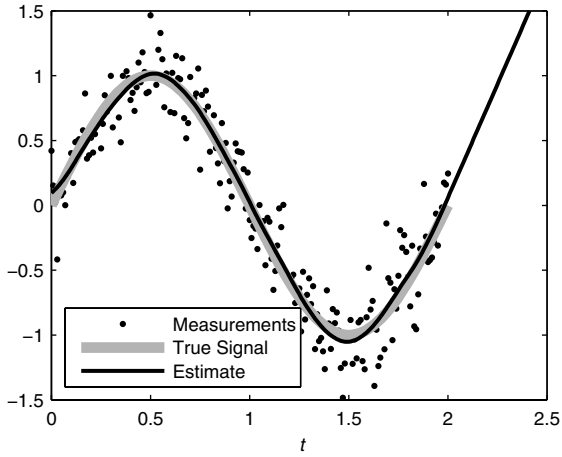


Figure 3.8 Example of prediction of a sine signal with optimal linear predictor (the Kalman filter prediction step) using the locally linear state space model. The prediction is a straight line extending to infinity as the model states.

Because the model is a linear Gaussian state space model, we can use linear Kalman filter for estimating the parameters.

There is no reason why we should select the functions g_i above to be explicit functions of time. Instead, they can be functions of arbitrary regressors as is illustrated in the following example.

Example 3.2 (Linear-in-parameters regression model II) *Consider the following parametric model, where* $g_1(\mathbf{s}), \ldots, g_d(\mathbf{s})$ *are some given functions of a regressor variable* $\mathbf{s} \in \mathbb{R}^n$*:*

$$y_k = w_0 + w_1\, g_1(\mathbf{s}_k) + \cdots + w_d\, g_d(\mathbf{s}_k) + \varepsilon_k, \tag{3.17}$$

where the weights $w_0, \ldots, w_d$ *are to be estimated from a set of measurements* $\{(\mathbf{s}_1, y_1), \ldots, (\mathbf{s}_T, y_T)\}$*. Analogously to the previous example, we can convert the problem into a linear Gaussian state space model by defining* $\mathbf{H}_k = (1\ g_1(\mathbf{s}_k)\ \cdots\ g_d(\mathbf{s}_k))$ *and* $\mathbf{x}_k = \mathbf{x} = (w_0\ w_1\ \cdots\ w_d)^\mathsf{T}$*.*

Linearity of the state space models in the above examples resulted from the property that the models are linear in parameters. Generalized linear models involving non-linear link functions will lead to non-linear state space models, as is illustrated with the following example.

Example 3.3 (Generalized linear model) *An example of a generalized linear model is*

$$y_k = g(w_0 + \mathbf{w}^\mathsf{T} \mathbf{s}_k) + \varepsilon_k, \tag{3.18}$$

where $g(\cdot)$ is some given non-linear link function and $\mathbf{s}_k$ is a vector of regressors. If we define the state as $\mathbf{x} = (w_0, \mathbf{w})$ and $h_k(\mathbf{x}) \triangleq g(w_0 + \mathbf{w}^\mathsf{T} \mathbf{s}_k)$, we get the following non-linear Gaussian state space model:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{x}_{k-1}, \\ y_k &= h_k(\mathbf{x}_k) + \varepsilon_k. \end{aligned} \tag{3.19}$$

Because the state space model is non-linear, instead of the linear Kalman filter, we need to use non-linear Kalman filters such as the extended Kalman filter (EKF) or the unscented Kalman filter (UKF) to cope with the non-linearity.

One general class of non-linear regression models, which can be converted into state space models using an analogous approach to the above, is the multi-layer perceptron (MLP) neural networks (see, e.g., Bishop, 2006). Using a non-linear Kalman filter is indeed one way to train (to estimate the parameters of) such models (Haykin, 2001). However, non-linear regression models of this kind arise in various others contexts as well.

In digital signal processing (DSP), an important class of models is linear signal models such as autoregressive (AR) models, moving average (MA) models, autoregressive moving average models (ARMA) and their generalizations (see, e.g., Hayes, 1996). In those models one is often interested in performing adaptive filtering, which refers to the methodology where the parameters of the signal model are estimated from data. These kinds of adaptive filtering problem can often be formulated as Kalman filtering problems, as is illustrated in the following example.

Example 3.4 (Autoregressive (AR) model) *An autoregressive (AR) model of order d has the form*

$$y_k = w_1 y_{k-1} + \cdots + w_d y_{k-d} + \varepsilon_k, \tag{3.20}$$

where ε_k is a white Gaussian noise process. The problem of adaptive filtering is to estimate the weights $w_1, \ldots, w_d$ given the observed signal $y_1, y_2, y_3, \ldots$. If we let $\mathbf{H}_k = (y_{k-1} \ \cdots \ y_{k-d})$ and define the state as $\mathbf{x}_k = (w_1 \ \cdots \ w_d)^\mathsf{T}$, we get a linear Gaussian state space model

$$\begin{aligned} \mathbf{x}_k &= \mathbf{x}_{k-1}, \\ y_k &= \mathbf{H}_k \mathbf{x}_k + \varepsilon_k. \end{aligned} \tag{3.21}$$

Thus the adaptive filtering problem can be solved with a linear Kalman filter.

The classical algorithm for an adaptive filtering is called the least mean squares (LMS) algorithm, and it can be interpreted as an approximate version of the above Kalman filter. However, in LMS algorithms it is common to allow the model to change in time, which in the state space context corresponds to setting up a dynamic model for the model parameters. This kind of model is illustrated in the next example.

Example 3.5 (Time-varying autoregressive (TVAR) model) *In a time-varying autoregressive (TVAR) model the weights are assumed to depend on the time step number k as*

$$y_k = w_{1,k}\, y_{k-1} + \cdots + w_{d,k}\, y_{k-d} + \varepsilon_k. \tag{3.22}$$

A typical model for the time dependence of weights is the random walk model

$$w_{i,k} = w_{i,k-1} + q_{i,k-1}, \quad q_{i,k-1} \sim \mathrm{N}(0, Q_i), \quad i = 1, \ldots, d. \tag{3.23}$$

If we define the state as $\mathbf{x}_k = (w_{1,k-1} \ \cdots \ w_{d,k-1})^\mathsf{T}$*, this model can be written as a linear Gaussian state space model with process noise* $\mathbf{q}_{k-1} = (q_{1,k-1} \ \cdots \ q_{d,k-1})^\mathsf{T}$*:*

$$\begin{aligned} \mathbf{x}_k &= \mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ y_k &= \mathbf{H}_k\, \mathbf{x}_k + \varepsilon_k. \end{aligned} \tag{3.24}$$

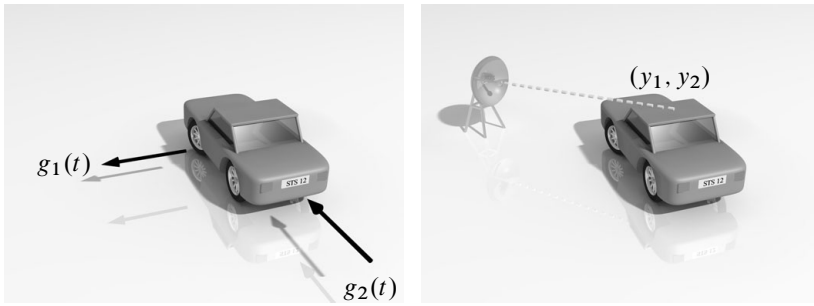
More general (TV)ARMA models can be handled similarly.

In tracking and navigation (Bar-Shalom et al., 2001) the dynamic models are often built based on Newtonian physics and classical mechanics. This procedure also leads to linear and non-linear state space models. In the following example we build a simple dynamic model for a car and its measurement process.

Example 3.6 (State space model of a car) *The dynamics of the car in 2d* (x_1, x_2) *are governed by* Newton's law *(see Figure 3.9(a))*

$$\mathbf{g}(t) = m\, \mathbf{a}(t), \tag{3.25}$$

where $\mathbf{a}(t)$ *is the acceleration, m is the mass of the car, and* $\mathbf{g}(t)$ *is a vector of (unknown) forces acting on the car. Let's now model* $\mathbf{g}(t)/m$ *as a two-*



(a) (b)

Figure 3.9 Illustration of car's (a) dynamic and (b) measurement models. In the dynamic model, the unknown forces $g_1(t)$ and $g_2(t)$ are modeled as white noise processes. The measurements (y_1, y_2) are modeled as noise corrupted observations of the car's position.

dimensional white random process

$$\begin{aligned} \frac{\mathrm{d}^2 x_1}{\mathrm{d}t^2} &= w_1(t), \\ \frac{\mathrm{d}^2 x_2}{\mathrm{d}t^2} &= w_2(t). \end{aligned} \tag{3.26}$$

If we define $x_3(t) = \mathrm{d}x_1/\mathrm{d}t$, $x_4(t) = \mathrm{d}x_2/\mathrm{d}t$, *then the model can be written as a first order* system of differential equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{\mathbf{F}} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}}_{\mathbf{L}} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}. \tag{3.27}$$

In shorter matrix form *this can be written as a continuous-time linear dynamic model of the form*

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{F}\,\mathbf{x} + \mathbf{L}\,\mathbf{w}.$$

If the state of the car is measured (sampled) with sampling period Δt *it suffices to consider the state of the car only at the time instants* $t \in$

$\{0, \Delta t, 2\Delta t, \ldots\}$. *The dynamic model can be discretized, which leads to*

$$\begin{pmatrix} x_{1,k} \\ x_{2,k} \\ x_{3,k} \\ x_{4,k} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} x_{1,k-1} \\ x_{2,k-1} \\ x_{3,k-1} \\ x_{4,k-1} \end{pmatrix} + \mathbf{q}_k, \tag{3.28}$$

where $\mathbf{q}_k$ *is a discrete-time Gaussian noise process. This can be seen to be a (discrete-time) linear dynamic model of the form*

$$\mathbf{x}_k = \mathbf{A}\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1},$$

where $\mathbf{x}_k = \mathbf{x}(t_k)$*, and* $\mathbf{A}$ *is the transition matrix.*

Assume that the position of the car (x_1, x_2) *is measured and the measurements are corrupted by Gaussian measurement noise* $(r_{1,k}, r_{2,k})$ *(see Figure 3.9(b)):*

$$\begin{aligned} y_{1,k} &= x_{1,k} + r_{1,k}, \\ y_{2,k} &= x_{2,k} + r_{2,k}. \end{aligned} \tag{3.29}$$

The measurement model can now be written as

$$\mathbf{y}_k = \mathbf{H}\,\mathbf{x}_k + \mathbf{r}_k, \qquad \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{3.30}$$

The dynamic and measurement models of the car form a linear Gaussian state space model

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{H}\,\mathbf{x}_k + \mathbf{r}_k, \end{aligned}$$

where $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$ *and* $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R})$*. The state of the car from the noisy measurements can be estimated using the Kalman filter.*

The above car model is a linear Gaussian state space model, because the model was based on a linear differential equation, and the measured quantities were linear functions of the state. However, if either the dynamic or measurement model is non-linear, we get a non-linear state space model, as is illustrated in the following example.

Example 3.7 (Noisy pendulum model) *The differential equation for a simple pendulum (see Figure 3.10) with unit length and mass can be written as*

$$\frac{\mathrm{d}^2\alpha}{\mathrm{d}t^2} = -g\,\sin(\alpha) + w(t), \tag{3.31}$$

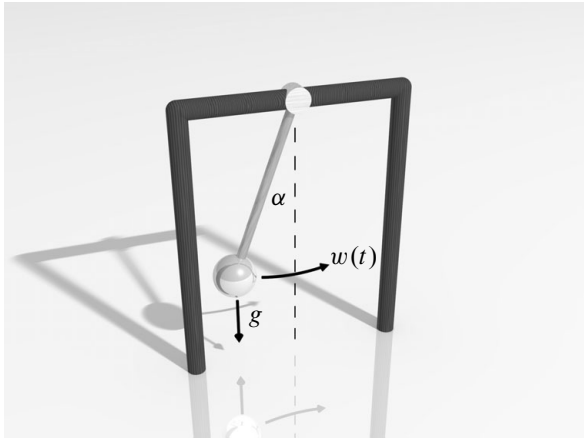


Figure 3.10 Illustration of pendulum example. In addition to the gravitation g there is an additional unknown force component $w(t)$, which is modeled as white noise.

where α is the angle, g is the gravitational acceleration and $w(t)$ is a random noise process. This model can be converted into the following state space model:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -g\ \sin(x_1) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} w(t), \tag{3.32}$$

where $x_1 = \alpha$ and $x_2 = \mathrm{d}\alpha/\,\mathrm{d}t$. This can be seen to be a special case of continuous-time non-linear dynamic models of the form

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}) + \mathbf{L}\,\mathbf{w}, \tag{3.33}$$

where $\mathbf{f}(\mathbf{x})$ is a non-linear function. We can then, for example, consider measuring the angular position of the pendulum, which leads to the linear Gaussian measurement model

$$y_k = \alpha(t_k) + \mathit{noise}. \tag{3.34}$$

However, if we measured only the horizontal position of the pendulum, we would get a non-linear measurement model

$$y_k = \sin(\alpha(t_k)) + \mathit{noise}. \tag{3.35}$$

By a suitable numerical integration scheme we can convert the non-linear dynamic model into a discrete-time non-linear dynamic model, which then

results in a model of the generic non-linear state space form

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}), \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{r}_k), \end{aligned} \tag{3.36}$$

where $\mathbf{y}_k$ *is the vector of measurements,* $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$*, and* $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R})$*. Estimation of the pendulum state can be now implemented using, for example, the extended Kalman filter (EKF), unscented Kalman filter (UKF) or particle filter.*

3.7 Exercises

3.1 In this exercise your task is to implement a simple Kalman filter using ready-made codes and analyze the results.

(a) Download and install the EKF/UKF toolbox[2] to some MATLAB® computer. Run the following demonstrations:

```
demos/kf_sine_demo/kf_sine_demo.m
demos/kf_cwpa_demo/kf_cwpa_demo.m
```

After running them read the contents of these files and try to understand how they have been implemented. Also read the documentation for the functions `kf_predict` and `kf_update` (type, e.g., "`help kf_predict`" in MATLAB®).

(b) Consider the following state space model:

$$\begin{aligned} \mathbf{x}_k &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ y_k &= \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{x}_k + r_k, \end{aligned} \tag{3.37}$$

where $\mathbf{x}_k = (x_k \ \dot{x}_k)^\mathsf{T}$ is the state, y_k is the measurement, and $\mathbf{q}_k \sim \mathrm{N}(\mathbf{0}, \mathrm{diag}(1/10^2, 1^2))$ and $r_k \sim \mathrm{N}(0, 10^2)$ are white Gaussian noise processes.

Simulate a 100 step state sequence from the model and plot the signal x_k, signal derivative $\dot{x}_k$, and the simulated measurements y_k. Start from an initial state drawn from the zero-mean 2d-Gaussian distribution with identity covariance.

(c) Use the Kalman filter for computing the state estimates $\mathbf{m}_k$ using MATLAB®-code along the following lines:

```
m = [0;0];  % Initial mean
P = eye(2); % Initial covariance
for k = 1:100
```

[2] The EKF/UKF toolbox is available for download on the web page `becs.aalto.fi/en/research/bayes/ekfukf/`

```
    [m,P] = kf_predict(m,P,A,Q);
    [m,P] = kf_update(m,P,y(k),H,R);
    % Store the estimate m of state x_k here
end
```

(d) Plot the state estimates $\mathbf{m}_k$, the true states $\mathbf{x}_k$ and measurements y_k. Compute the root mean squared error (RMSE) of using the first components of vectors $\mathbf{m}_k$ as the estimates of the first components of states $\mathbf{x}_k$. Also compute the RMSE error that we would have if we used the measurements as the estimates.

3.2 Note that the model in Exercise 2.4 can be rewritten as a linear state space model

$$\begin{aligned} \mathbf{w}_k &= \mathbf{w}_{k-1}, \\ y_k &= \mathbf{H}_k \, \mathbf{w}_k + \varepsilon_k, \end{aligned}$$

where $\mathbf{H}_k = (x_k \;\; 1)$, $\mathbf{w}_0 \sim \mathrm{N}(0, \sigma^2 \mathbf{I})$ and $\varepsilon_k \sim \mathrm{N}(0, 1)$. The state in the model is now $\mathbf{w}_k = (\theta_1 \;\; \theta_2)^\mathsf{T}$ and the measurements are y_k for $k = 1, \ldots, T$. Assume that the Kalman filter is used for processing the measurements $y_1, \ldots, y_T$. Your task is to prove that at time step T, the mean and covariance of $\mathbf{w}_T$ computed by the Kalman filter are the same as the mean and covariance of the posterior distribution computed in Exercise 2.4.
The Kalman filter equations for the above model can be written as:

$$\begin{aligned} S_k &= \mathbf{H}_k \, \mathbf{P}_{k-1} \, \mathbf{H}_k^\mathsf{T} + 1, \\ \mathbf{K}_k &= \mathbf{P}_{k-1} \, \mathbf{H}_k^\mathsf{T} \, S_k^{-1}, \\ \mathbf{m}_k &= \mathbf{m}_{k-1} + \mathbf{K}_k \, (y_k - \mathbf{H}_k \, \mathbf{m}_{k-1}), \\ \mathbf{P}_k &= \mathbf{P}_{k-1} - \mathbf{K}_k \, S_k \, \mathbf{K}_k^\mathsf{T}. \end{aligned}$$

(a) Write formulas for the posterior mean $\mathbf{m}_{k-1}$ and covariance $\mathbf{P}_{k-1}$ assuming that they are the same as those which would be obtained if the pairs $\{(x_i, y_i) \;:\; i = 1, \ldots, k-1\}$ were (batch) processed as in Exercise 2.4. Write similar equations for the mean $\mathbf{m}_k$ and covariance $\mathbf{P}_k$. Show that the posterior means can be expressed in the form

$$\begin{aligned} \mathbf{m}_{k-1} &= \mathbf{P}_{k-1} \, \mathbf{X}_{k-1}^\mathsf{T} \, \mathbf{y}_{k-1}, \\ \mathbf{m}_k &= \mathbf{P}_k \, \mathbf{X}_k^\mathsf{T} \, \mathbf{y}_k, \end{aligned}$$

where $\mathbf{X}_{k-1}$ and $\mathbf{y}_{k-1}$ have been constructed as $\mathbf{X}$ and $\mathbf{y}$ in Exercise 2.4, except that only the pairs $\{(x_i, y_i) \;:\; i = 1, \ldots, k-1\}$ have been used, and $\mathbf{X}_k$ and $\mathbf{y}_k$ have been constructed similarly from pairs up to the step k.

(b) Rewrite the expressions $\mathbf{X}_k^\mathsf{T} \, \mathbf{X}_k$ and $\mathbf{X}_k^\mathsf{T} \, \mathbf{y}_k$ in terms of $\mathbf{X}_{k-1}$, $\mathbf{y}_{k-1}$, $\mathbf{H}_k$ and y_k. Substitute these into the expressions of $\mathbf{m}_k$ and $\mathbf{P}_k$ obtained in (a).

(c) Expand the expression of the covariance $\mathbf{P}_k = \mathbf{P}_{k-1} - \mathbf{K}_k\, S_k\, \mathbf{K}_k^\mathsf{T}$ by substituting the expressions for $\mathbf{K}_k$ and S_k. Convert it to a simpler form by applying the matrix inversion lemma

$$\mathbf{P}_{k-1} - \mathbf{P}_{k-1}\, \mathbf{H}_k^\mathsf{T}\, (\mathbf{H}_k\, \mathbf{P}_{k-1}\, \mathbf{H}_k^\mathsf{T} + 1)^{-1}\, \mathbf{H}_k\, \mathbf{P}_{k-1} = (\mathbf{P}_{k-1}^{-1} + \mathbf{H}_k^\mathsf{T}\, \mathbf{H}_k)^{-1}.$$

Show that this expression for $\mathbf{P}_k$ is equivalent to the expression in (a).

(d) Expand the expression of the mean $\mathbf{m}_k = \mathbf{m}_{k-1} + \mathbf{K}_k\, (y_k - \mathbf{H}_k\, \mathbf{m}_{k-1})$ and show that the result is equivalent to the expression obtained in (a). *Hint:* the Kalman gain can also be written as $\mathbf{K}_k = \mathbf{P}_k\, \mathbf{H}_k^\mathsf{T}$.

(e) Prove by an induction argument that the mean and covariance computed by the Kalman filter at step T is the same as the posterior mean and covariance obtained in Exercise 2.4.

3.3 Consider the regression problem

$$\begin{aligned} y_k &= a_1\, s_k + a_2\, \sin(s_k) + b + \varepsilon_k, \qquad k = 1, \ldots, T, \\ \varepsilon_k &\sim \mathrm{N}(0, R), \\ a_1 &\sim \mathrm{N}(0, \sigma_1^2), \\ a_2 &\sim \mathrm{N}(0, \sigma_2^2), \\ b &\sim \mathrm{N}(0, \sigma_b^2), \end{aligned} \tag{3.38}$$

where $s_k \in \mathbb{R}$ are the known regressor values, $R, \sigma_1^2, \sigma_2^2, \sigma_b^2$ are given positive constants, $y_k \in \mathbb{R}$ are the observed output variables, and ε_k are independent Gaussian measurement errors. The scalars a_1, a_2, and b are the unknown parameters to be estimated. Formulate the estimation problem as a linear Gaussian state space model.

3.4 Consider the model

$$\begin{aligned} x_k &= \sum_{i=1}^{d} a_i\, x_{k-i} + q_{k-1}, \\ y_k &= x_k + \varepsilon_k, \end{aligned} \tag{3.39}$$

where the values $\{y_k\}$ are observed, q_{k-1} and ε_k are independent Gaussian noises, and the weights $a_1, \ldots, a_d$ are known. The aim is to estimate the sequence $\{x_k\}$. Rewrite the problem as an estimation problem in a linear Gaussian state space model.

3.5 Recall that the Gaussian probability density is defined as

$$\mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P}) = \frac{1}{(2\,\pi)^{n/2}\, |\mathbf{P}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m})^\mathsf{T}\, \mathbf{P}^{-1}\, (\mathbf{x} - \mathbf{m})\right).$$

Derive the following Gaussian identities.

(a) Let $\mathbf{x}$ and $\mathbf{y}$ have the Gaussian densities

$$p(\mathbf{x}) = \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P}), \qquad p(\mathbf{y} \mid \mathbf{x}) = \mathrm{N}(\mathbf{y} \mid \mathbf{H}\, \mathbf{x}, \mathbf{R}),$$

then the joint distribution of $\mathbf{x}$ and $\mathbf{y}$ is

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix}\mathbf{m}\\ \mathbf{H}\,\mathbf{m}\end{pmatrix}, \begin{pmatrix}\mathbf{P} & \mathbf{P}\,\mathbf{H}^{\mathsf{T}}\\ \mathbf{H}\,\mathbf{P} & \mathbf{H}\,\mathbf{P}\,\mathbf{H}^{\mathsf{T}} + \mathbf{R}\end{pmatrix}\right)$$

and the marginal distribution of $\mathbf{y}$ is

$$\mathbf{y} \sim \mathrm{N}(\mathbf{H}\,\mathbf{m}, \mathbf{H}\,\mathbf{P}\,\mathbf{H}^{\mathsf{T}} + \mathbf{R}).$$

Hint: use the properties of expectation $\mathrm{E}[\mathbf{H}\,\mathbf{x} + \mathbf{r}] = \mathbf{H}\ \mathrm{E}[\mathbf{x}] + \mathrm{E}[\mathbf{r}]$ and $\mathrm{Cov}[\mathbf{H}\,\mathbf{x} + \mathbf{r}] = \mathbf{H}\ \mathrm{Cov}[\mathbf{x}]\,\mathbf{H}^{\mathsf{T}} + \mathrm{Cov}[\mathbf{r}]$ (if $\mathbf{x}$ and $\mathbf{r}$ are independent).

(b) Write down the explicit expression for the joint and marginal probability densities above:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x})\, p(\mathbf{x}) = ?$$
$$p(\mathbf{y}) = \int p(\mathbf{y} \mid \mathbf{x})\, p(\mathbf{x})\, d\mathbf{x} = ?$$

(c) If the random variables $\mathbf{x}$ and $\mathbf{y}$ have the joint Gaussian probability density

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix}\mathbf{a}\\ \mathbf{b}\end{pmatrix}, \begin{pmatrix}\mathbf{A} & \mathbf{C}\\ \mathbf{C}^{\mathsf{T}} & \mathbf{B}\end{pmatrix}\right),$$

then the conditional density of $\mathbf{x}$ given $\mathbf{y}$ is

$$\mathbf{x} \mid \mathbf{y} \sim \mathrm{N}(\mathbf{a} + \mathbf{C}\,\mathbf{B}^{-1}\,(\mathbf{y} - \mathbf{b}), \mathbf{A} - \mathbf{C}\,\mathbf{B}^{-1}\mathbf{C}^{\mathsf{T}}).$$

Hints:

- Denote inverse covariance as $\mathbf{D} = \begin{pmatrix}\mathbf{D}_{11} & \mathbf{D}_{12}\\ \mathbf{D}_{12}^{\mathsf{T}} & \mathbf{D}_{22}\end{pmatrix}$ and expand the quadratic form in the Gaussian exponent.
- Compute the derivative with respect to $\mathbf{x}$ and set it to zero. Conclude that due to symmetry the point where the derivative vanishes is the mean.
- Check from a linear algebra book that the inverse of $\mathbf{D}_{11}$ is given by the Schur complement:

$$\mathbf{D}_{11}^{-1} = \mathbf{A} - \mathbf{C}\,\mathbf{B}^{-1}\,\mathbf{C}^{\mathsf{T}}$$

  and that $\mathbf{D}_{12}$ can be then written as

$$\mathbf{D}_{12} = -\mathbf{D}_{11}\,\mathbf{C}\,\mathbf{B}^{-1}.$$

- Find the simplified expression for the mean by applying the identities above.
- Find the second derivative of the negative Gaussian exponent with respect to $\mathbf{x}$. Conclude that it must be the inverse conditional covariance of $\mathbf{x}$.

- Use the Schur complement expression above for computing the conditional covariance.

4

Bayesian filtering equations and exact solutions

In this chapter, we derive the Bayesian filtering equations, which are the general equations for computing Bayesian filtering solutions to both linear Gaussian and non-linear/non-Gaussian state space models. We also derive the Kalman filtering equations which give the closed form solution to the linear Gaussian Bayesian filtering problem.

4.1 Probabilistic state space models

Bayesian filtering is considered with state estimation in general probabilistic state space models which have the following form.

Definition 4.1 (Probabilistic state space model) A probabilistic state space model *or non-linear filtering model consists of a sequence of conditional probability distributions:*

$$\begin{aligned} \mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}), \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k), \end{aligned} \tag{4.1}$$

for $k = 1, 2, \ldots$*, where*

- $\mathbf{x}_k \in \mathbb{R}^n$ *is the* state *of the system at time step* k*,*
- $\mathbf{y}_k \in \mathbb{R}^m$ *is the measurement at time step* k*,*
- $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ *is the* dynamic model *which describes the stochastic dynamics of the system. The dynamic model can be a probability density, a counting measure or a combination of them depending on whether the state* $\mathbf{x}_k$ *is continuous, discrete, or hybrid.*
- $p(\mathbf{y}_k \mid \mathbf{x}_k)$ *is the* measurement model*, which is the distribution of measurements given the state.*

The model is assumed to be Markovian, which means that it has the following two properties.

Property 4.1 (Markov property of states)
The states $\{\mathbf{x}_k : k = 0, 1, 2, \ldots\}$ *form a Markov sequence (or Markov chain if the state is discrete). This* Markov property *means that* $\mathbf{x}_k$ *(and actually the whole future* $\mathbf{x}_{k+1}, \mathbf{x}_{k+2}, \ldots$*) given* $\mathbf{x}_{k-1}$ *is independent of anything that has happened before the time step* $k - 1$*:*

$$p(\mathbf{x}_k \mid \mathbf{x}_{1:k-1}, \mathbf{y}_{1:k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}). \tag{4.2}$$

Also the past is independent of the future given the present:

$$p(\mathbf{x}_{k-1} \mid \mathbf{x}_{k:T}, \mathbf{y}_{k:T}) = p(\mathbf{x}_{k-1} \mid \mathbf{x}_k). \tag{4.3}$$

Property 4.2 (Conditional independence of measurements)
The current measurement $\mathbf{y}_k$ *given the current state* $\mathbf{x}_k$ *is conditionally independent of the measurement and state histories:*

$$p(\mathbf{y}_k \mid \mathbf{x}_{1:k}, \mathbf{y}_{1:k-1}) = p(\mathbf{y}_k \mid \mathbf{x}_k). \tag{4.4}$$

A simple example of a Markovian sequence is the Gaussian random walk. When this is combined with noisy measurements, we obtain the following example of a probabilistic state space model.

Example 4.1 (Gaussian random walk) *A Gaussian random walk model can be written as*

$$\begin{aligned} x_k &= x_{k-1} + q_{k-1}, \quad & q_{k-1} &\sim \mathrm{N}(0, Q), \\ y_k &= x_k + r_k, & r_k &\sim \mathrm{N}(0, R), \end{aligned} \tag{4.5}$$

where x_k *is the hidden state (or signal) and* y_k *is the measurement. In terms of probability densities the model can be written as*

$$\begin{aligned} p(x_k \mid x_{k-1}) &= \mathrm{N}(x_k \mid x_{k-1}, Q) \\ &= \frac{1}{\sqrt{2\pi Q}} \exp\left(-\frac{1}{2Q}(x_k - x_{k-1})^2\right), \\ p(y_k \mid x_k) &= \mathrm{N}(y_k \mid x_k, R) \\ &= \frac{1}{\sqrt{2\pi R}} \exp\left(-\frac{1}{2R}(y_k - x_k)^2\right), \end{aligned} \tag{4.6}$$

which is a probabilistic state space model. Example realizations of the signal x_k *and measurements* y_k *are shown in Figure 4.1. The parameter values in the simulation were* $Q = R = 1$*.*

With the Markovian assumption and the filtering model (4.1), the *joint prior distribution of the states* $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \ldots, \mathbf{x}_T\}$, and the *joint likeli-*

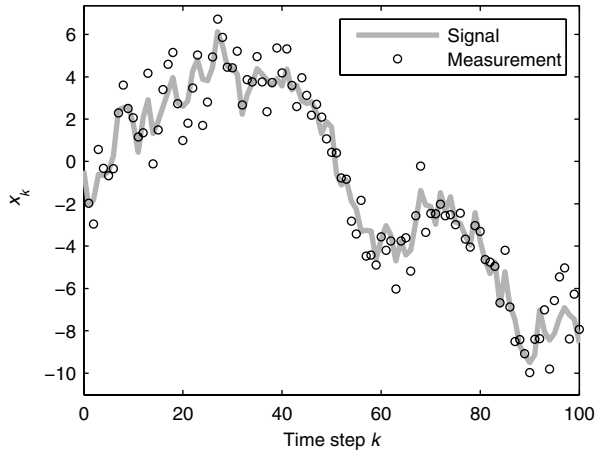


Figure 4.1 Simulated signal and measurements from the Gaussian random walk model in Example 4.1.

hood of the measurements $\mathbf{y}_{1:T} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ are, respectively,

$$p(\mathbf{x}_{0:T}) = p(\mathbf{x}_0) \prod_{k=1}^{T} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}), \tag{4.7}$$

$$p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T}) = \prod_{k=1}^{T} p(\mathbf{y}_k \mid \mathbf{x}_k). \tag{4.8}$$

In principle, for a given T we could simply compute the posterior distribution of the states by Bayes' rule:

$$\begin{aligned} p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}) &= \frac{p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T})\, p(\mathbf{x}_{0:T})}{p(\mathbf{y}_{1:T})} \\ &\propto p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T})\, p(\mathbf{x}_{0:T}). \end{aligned} \tag{4.9}$$

However, this kind of explicit usage of the full Bayes' rule is not feasible in real-time applications, because the number of computations per time step increases as new observations arrive. Thus, this way we could only work with small data sets, because if the amount of data is unbounded (as in real-time sensing applications), then at some point of time the computations will become intractable. To cope with real-time data we need to have an algorithm which does a constant number of computations per time step.

As discussed in Section 1.3, *filtering distributions*, *prediction distributions*, and *smoothing distributions* can be computed recursively such that only a constant number of computations is done on each time step. For this reason we shall not consider the full posterior computation at all, but concentrate on the above-mentioned distributions instead. In this and the few following chapters, we consider computation of the filtering and prediction distributions; algorithms for computing the smoothing distributions will be considered in later chapters.

4.2 Bayesian filtering equations

The purpose of *Bayesian filtering* is to compute the *marginal posterior distribution* or *filtering distribution* of the state $\mathbf{x}_k$ at each time step k given the history of the measurements up to the time step k:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}). \tag{4.10}$$

The fundamental equations of the Bayesian filtering theory are given by the following theorem.

Theorem 4.1 (Bayesian filtering equations) *The recursive equations (the* Bayesian filter*) for computing the* predicted distribution $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$ *and the* filtering distribution $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ *at the time step k are given by the following* Bayesian filtering equations.

- Initialization. *The recursion starts from the prior distribution* $p(\mathbf{x}_0)$.
- Prediction step. *The predictive distribution of the state* $\mathbf{x}_k$ *at the time step k, given the dynamic model, can be computed by the Chapman–Kolmogorov equation*

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, \mathrm{d}\mathbf{x}_{k-1}. \tag{4.11}$$

- Update step. *Given the measurement* $\mathbf{y}_k$ *at time step k the posterior distribution of the state* $\mathbf{x}_k$ *can be computed by Bayes' rule*

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{1}{Z_k}\, p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}), \tag{4.12}$$

where the normalization constant Z_k *is given as*

$$Z_k = \int p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})\, \mathrm{d}\mathbf{x}_k. \tag{4.13}$$

If some of the components of the state are discrete, the corresponding integrals are replaced with summations.

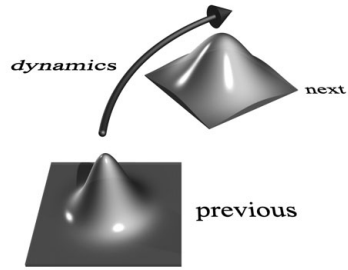


Figure 4.2 Visualization of the prediction step: prediction propagates the state distribution of the previous measurement step through the dynamic model such that the uncertainties (stochastics) in the dynamic model are taken into account.

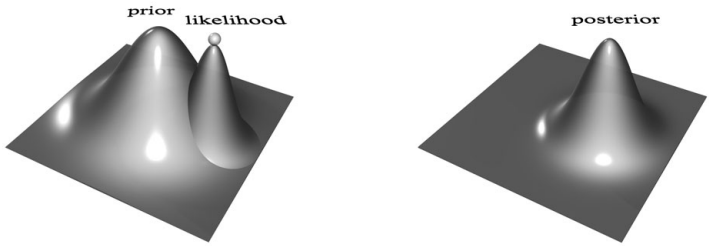


Figure 4.3 Visualization of the update step: (a) prior distribution from prediction and the likelihood of measurement just before the update step; (b) the posterior distribution after combining the prior and likelihood by Bayes' rule.

Proof The joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ given $\mathbf{y}_{1:k-1}$ can be computed as

$$\begin{aligned} p(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_{1:k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) \\ &= p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}), \end{aligned} \tag{4.14}$$

where the disappearance of the measurement history $\mathbf{y}_{1:k-1}$ is due to the Markov property of the sequence $\{\mathbf{x}_k : k = 1, 2, \ldots\}$. The marginal distribution of $\mathbf{x}_k$ given $\mathbf{y}_{1:k-1}$ can be obtained by integrating the distribution (4.14) over $\mathbf{x}_{k-1}$, which gives the *Chapman–Kolmogorov equation*

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, \mathrm{d}\mathbf{x}_{k-1}. \tag{4.15}$$

If $\mathbf{x}_{k-1}$ is discrete, then the above integral is replaced with summation over $\mathbf{x}_{k-1}$. The distribution of $\mathbf{x}_k$ given $\mathbf{y}_k$ and $\mathbf{y}_{1:k-1}$, that is, given $\mathbf{y}_{1:k}$, can be computed by *Bayes' rule*

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) &= \frac{1}{Z_k}\, p(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1})\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) \\ &= \frac{1}{Z_k}\, p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}), \end{aligned} \tag{4.16}$$

where the normalization constant is given by Equation (4.13). The disappearance of the measurement history $\mathbf{y}_{1:k-1}$ in Equation (4.16) is due to the conditional independence of $\mathbf{y}_k$ of the measurement history, given $\mathbf{x}_k$. □

4.3 Kalman filter

The Kalman filter (Kalman, 1960b) is the closed form solution to the Bayesian filtering equations for the filtering model, where the dynamic and measurement models are linear Gaussian:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}_{k-1}\, \mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{H}_k\, \mathbf{x}_k + \mathbf{r}_k, \end{aligned} \tag{4.17}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state, $\mathbf{y}_k \in \mathbb{R}^m$ is the measurement, $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ is the process noise, $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}_k)$ is the measurement noise, and the prior distribution is Gaussian $\mathbf{x}_0 \sim \mathrm{N}(\mathbf{m}_0, \mathbf{P}_0)$. The matrix $\mathbf{A}_{k-1}$ is the transition matrix of the dynamic model and $\mathbf{H}_k$ is the measurement model matrix. In probabilistic terms the model is

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{A}_{k-1}\, \mathbf{x}_{k-1}, \mathbf{Q}_{k-1}), \\ p(\mathbf{y}_k \mid \mathbf{x}_k) &= \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}_k\, \mathbf{x}_k, \mathbf{R}_k). \end{aligned} \tag{4.18}$$

Theorem 4.2 (Kalman filter) *The Bayesian filtering equations for the linear filtering model* (4.17) *can be evaluated in closed form and the resulting*

distributions are Gaussian:

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-), \\ p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k), \\ p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}) &= \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}_k \mathbf{m}_k^-, \mathbf{S}_k). \end{aligned} \tag{4.19}$$

The parameters of the distributions above can be computed with the following Kalman filter prediction *and* update steps*.*

- The prediction step *is*

$$\begin{aligned} \mathbf{m}_k^- &= \mathbf{A}_{k-1}\, \mathbf{m}_{k-1}, \\ \mathbf{P}_k^- &= \mathbf{A}_{k-1}\, \mathbf{P}_{k-1}\, \mathbf{A}_{k-1}^\mathsf{T} + \mathbf{Q}_{k-1}. \end{aligned} \tag{4.20}$$

- The update step *is*

$$\begin{aligned} \mathbf{v}_k &= \mathbf{y}_k - \mathbf{H}_k\, \mathbf{m}_k^-, \\ \mathbf{S}_k &= \mathbf{H}_k\, \mathbf{P}_k^-\, \mathbf{H}_k^\mathsf{T} + \mathbf{R}_k, \\ \mathbf{K}_k &= \mathbf{P}_k^-\, \mathbf{H}_k^\mathsf{T}\, \mathbf{S}_k^{-1}, \\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\, \mathbf{v}_k, \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\, \mathbf{S}_k\, \mathbf{K}_k^\mathsf{T}. \end{aligned} \tag{4.21}$$

The recursion is started from the prior mean $\mathbf{m}_0$ *and covariance* $\mathbf{P}_0$*.*

Proof The Kalman filter equations can be derived as follows.

1 By Lemma A.1 on page 209, the joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ given $\mathbf{y}_{1:k-1}$ is

$$\begin{aligned} &p(\mathbf{x}_{k-1}, \mathbf{x}_k \mid \mathbf{y}_{1:k-1}) \\ &= p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) \\ &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{A}_{k-1}\, \mathbf{x}_{k-1}, \mathbf{Q}_{k-1})\, \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1}) \\ &= \mathrm{N}\left(\begin{pmatrix} \mathbf{x}_{k-1} \\ \mathbf{x}_k \end{pmatrix} \,\middle|\, \mathbf{m}', \mathbf{P}' \right), \end{aligned} \tag{4.22}$$

where

$$\begin{aligned} \mathbf{m}' &= \begin{pmatrix} \mathbf{m}_{k-1} \\ \mathbf{A}_{k-1}\, \mathbf{m}_{k-1} \end{pmatrix}, \\ \mathbf{P}' &= \begin{pmatrix} \mathbf{P}_{k-1} & \mathbf{P}_{k-1}\, \mathbf{A}_{k-1}^\mathsf{T} \\ \mathbf{A}_{k-1}\, \mathbf{P}_{k-1} & \mathbf{A}_{k-1}\, \mathbf{P}_{k-1}\, \mathbf{A}_{k-1}^\mathsf{T} + \mathbf{Q}_{k-1} \end{pmatrix}, \end{aligned} \tag{4.23}$$

and the marginal distribution of $\mathbf{x}_k$ is by Lemma A.2

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-), \tag{4.24}$$

where

$$\mathbf{m}_k^- = \mathbf{A}_{k-1}\,\mathbf{m}_{k-1}, \quad \mathbf{P}_k^- = \mathbf{A}_{k-1}\,\mathbf{P}_{k-1}\,\mathbf{A}_{k-1}^\mathsf{T} + \mathbf{Q}_{k-1}. \tag{4.25}$$

2 By Lemma A.1, the joint distribution of $\mathbf{y}_k$ and $\mathbf{x}_k$ is

$$\begin{aligned} p(\mathbf{x}_k, \mathbf{y}_k \mid \mathbf{y}_{1:k-1}) &= p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) \\ &= \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}_k\,\mathbf{x}_k, \mathbf{R}_k)\ \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-) \\ &= \mathrm{N}\left(\begin{pmatrix}\mathbf{x}_k \\ \mathbf{y}_k\end{pmatrix} \,\middle|\, \mathbf{m}'', \mathbf{P}''\right), \end{aligned} \tag{4.26}$$

where

$$\mathbf{m}'' = \begin{pmatrix}\mathbf{m}_k^- \\ \mathbf{H}_k\,\mathbf{m}_k^-\end{pmatrix}, \qquad \mathbf{P}'' = \begin{pmatrix}\mathbf{P}_k^- & \mathbf{P}_k^-\,\mathbf{H}_k^\mathsf{T} \\ \mathbf{H}_k\,\mathbf{P}_k^- & \mathbf{H}_k\,\mathbf{P}_k^-\,\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k\end{pmatrix}. \tag{4.27}$$

3 By Lemma A.2 the conditional distribution of $\mathbf{x}_k$ is

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{y}_k, \mathbf{y}_{1:k-1}) &= p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \\ &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k), \end{aligned} \tag{4.28}$$

where

$$\begin{aligned} \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{P}_k^-\,\mathbf{H}_k^\mathsf{T}(\mathbf{H}_k\,\mathbf{P}_k^-\,\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k)^{-1}[\mathbf{y}_k - \mathbf{H}_k\,\mathbf{m}_k^-], \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{P}_k^-\,\mathbf{H}_k^\mathsf{T}\,(\mathbf{H}_k\,\mathbf{P}_k^-\,\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k)^{-1}\,\mathbf{H}_k\,\mathbf{P}_k^-, \end{aligned} \tag{4.29}$$

which can be also written in the form (4.21). □

The functional form of the Kalman filter equations given here is not the only possible one. From a numerical stability point of view it would be better to work with matrix square roots of covariances instead of plain covariance matrices. The theory and details of implementation of this kind of method is well covered, for example, in the book of Grewal and Andrews (2001).

Example 4.2 (Kalman filter for a Gaussian random walk) *Assume that we are observing measurements y_k of the Gaussian random walk model given in Example 4.1 and we want to estimate the state x_k at each time step. The information obtained up to time step k is summarized by the Gaussian filtering density*

$$p(x_k \mid y_{1:k}) = \mathrm{N}(x_k \mid m_k, P_k). \tag{4.30}$$

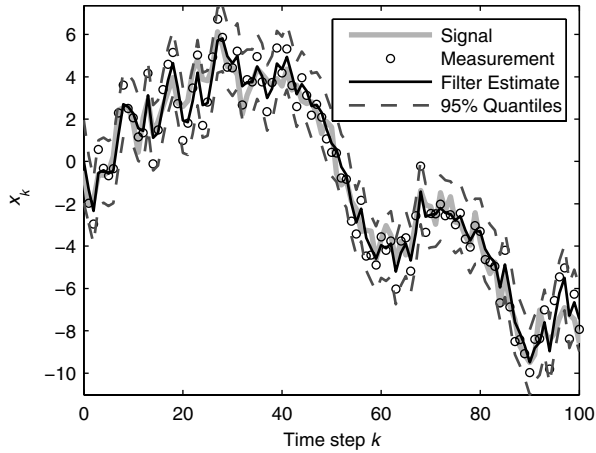


Figure 4.4 Signal, measurements and the result of Kalman filtering of the Gaussian random walk in Example 4.2.

The Kalman filter prediction and update equations are now given as

$$\begin{aligned} m_k^- &= m_{k-1}, \\ P_k^- &= P_{k-1} + Q, \\ m_k &= m_k^- + \frac{P_k^-}{P_k^- + R}(y_k - m_k^-), \\ P_k &= P_k^- - \frac{(P_k^-)^2}{P_k^- + R}. \end{aligned} \tag{4.31}$$

The result of applying this Kalman filter to the data in Figure 4.1 is shown in Figure 4.4.

Example 4.3 (Kalman filter for car tracking) *By discretizing the state space model for the car in Example 3.6 we get the following linear state space model:*

$$\mathbf{x}_k = \mathbf{A}\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \qquad \mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}), \tag{4.32}$$

$$\mathbf{y}_k = \mathbf{H}\,\mathbf{x}_k + \mathbf{r}_k, \qquad \mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}), \tag{4.33}$$

where the state is four dimensional, $\mathbf{x} = (x_1, x_2, x_3, x_4)$*, such that the position of the car is* (x_1, x_2) *and the corresponding velocities are* (x_3, x_4)*.*

The matrices in the dynamic model are

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{Q} = \begin{pmatrix} \frac{q_1^{\mathrm{c}} \Delta t^3}{3} & 0 & \frac{q_1^{\mathrm{c}} \Delta t^2}{2} & 0 \\ 0 & \frac{q_2^{\mathrm{c}} \Delta t^3}{3} & 0 & \frac{q_2^{\mathrm{c}} \Delta t^2}{2} \\ \frac{q_1^{\mathrm{c}} \Delta t^2}{2} & 0 & q_1^{\mathrm{c}} \Delta t & 0 \\ 0 & \frac{q_2^{\mathrm{c}} \Delta t^2}{2} & 0 & q_2^{\mathrm{c}} \Delta t \end{pmatrix},$$

where q_1^{c} and q_2^{c} are the spectral densities (continuous time variances) of the process noises in each direction. The matrices in the measurement model are

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \qquad \mathbf{R} = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix},$$

where σ_1^2 and σ_2^2 are the measurement noise variances in each position coordinate.

- *The Kalman filter prediction step now becomes the following:*

$$\mathbf{m}_k^- = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{m}_{k-1},$$

$$\mathbf{P}_k^- = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{P}_{k-1} \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{\mathsf{T}} + \begin{pmatrix} \frac{q_1^{\mathrm{c}} \Delta t^3}{3} & 0 & \frac{q_1^{\mathrm{c}} \Delta t^2}{2} & 0 \\ 0 & \frac{q_2^{\mathrm{c}} \Delta t^3}{3} & 0 & \frac{q_2^{\mathrm{c}} \Delta t^2}{2} \\ \frac{q_1^{\mathrm{c}} \Delta t^2}{2} & 0 & q_1^{\mathrm{c}} \Delta t & 0 \\ 0 & \frac{q_2^{\mathrm{c}} \Delta t^2}{2} & 0 & q_2^{\mathrm{c}} \Delta t \end{pmatrix}.$$

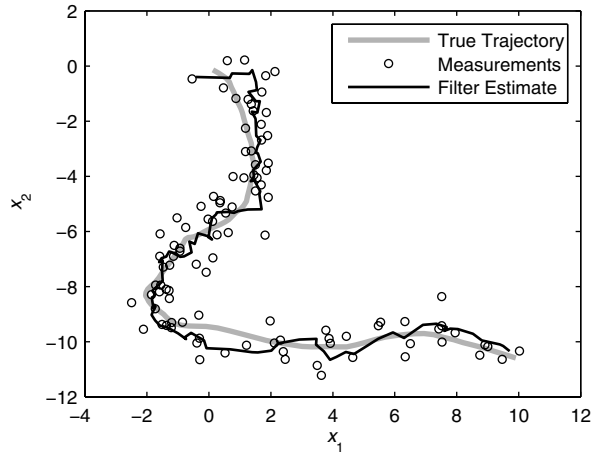


Figure 4.5 Simulated trajectory, measurements, and the result of the Kalman filter based car tracking in Example 4.3. The starting point is at the top of the trajectory. The RMSE position error based on the measurements only is 0.77 whereas the position RMSE of the Kalman filter estimate is 0.43.

- *The corresponding Kalman filter update step is*

$$\mathbf{S}_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{P}_k^- \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}^\mathsf{T} + \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix},$$

$$\mathbf{K}_k = \mathbf{P}_k^- \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}^\mathsf{T} \mathbf{S}_k^{-1},$$

$$\mathbf{m}_k = \mathbf{m}_k^- + \mathbf{K}_k \left(\mathbf{y}_k - \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{m}_k^- \right),$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \, \mathbf{S}_k \, \mathbf{K}_k^\mathsf{T}.$$

The result of applying this filter to simulated data is shown in Figure 4.5. The parameter values used in the simulation were $\sigma_1 = \sigma_2 = 1/2$, $q_1^\mathrm{c} = q_2^\mathrm{c} = 1$, *and* $\Delta t = 1/10$.

Although the Kalman filter can be seen as the closed form Bayesian solution to the linear Gaussian filtering problem, the original derivation of Kalman (1960b) is based on a different principle. In his seminal article, Kalman (1960b) derived the filter by considering orthogonal projections on the linear manifold spanned by the measurements. A similar approach

is also employed in many other works concentrating on estimation in linear systems such as in the book of Kailath et al. (2000). The advantage of the approach is that it avoids explicit Gaussian assumptions in the noise processes, as the Kalman filter can be seen as a more general best linear unbiased estimator of the state. The disadvantage is that the route to non-linear models is much less straightforward than in the present Bayesian approach.

4.4 Exercises

4.1 Derive the Kalman filter equations for the following linear-Gaussian filtering model with non-zero-mean noises:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{H}\,\mathbf{x}_k + \mathbf{r}_k, \end{aligned} \tag{4.34}$$

where $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{m}_q, \mathbf{Q})$ and $\mathbf{r}_k \sim \mathrm{N}(\mathbf{m}_r, \mathbf{R})$.

4.2 Write down the Bayesian filtering equations for finite-state hidden Markov models (HMM), that is, for models where the state only takes values from a finite set $x_k \in \{1, \ldots, N_x\}$.

4.3 Implement the Kalman filter in Example 4.2 for the Gaussian random walk model with MATLAB®. Draw realizations of state and measurement sequences and apply the filter to it. Plot the evolution of the filtering distribution.

4.4 Select a finite interval in the state space, say, $x \in [-10, 10]$ and discretize it evenly to N subintervals (e.g. $N = 1000$). Using a suitable numerical approximation to the integrals in the Bayesian filtering equations, implement a finite grid approximation to the Bayesian filter for the Gaussian random walk in Example 4.1. Verify that the result is practically the same as that of the Kalman filter above.

4.5 Derive the stationary Kalman filter for the Gaussian random walk model. That is, compute the limiting Kalman filter gain when $k \to \infty$ and write down the mean equation of the resulting constant-gain Kalman filter. Plot the frequency response of the resulting time-invariant filter. Which type of digital filter is it?

4.6 Consider the following dynamic model:

$$\begin{aligned} \mathbf{x}_k &= \begin{pmatrix} \cos\omega & \frac{\sin(\omega)}{\omega} \\ -\omega\,\sin(\omega) & \cos(\omega) \end{pmatrix} \mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ y_k &= \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{x}_k + r_k, \end{aligned}$$

where $\mathbf{x}_k \in \mathbb{R}^2$ is the state, y_k is the measurement, $r_k \sim \mathrm{N}(0, 0.1)$ is a white Gaussian measurement noise, and $\mathbf{q}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$, where

$$\mathbf{Q} = \begin{pmatrix} \frac{q^c\,\omega - q^c\,\cos(\omega)\,\sin(\omega)}{2\omega^3} & \frac{q^c\,\sin^2(\omega)}{2\omega^2} \\ \frac{q^c\,\sin^2(\omega)}{2\omega^2} & \frac{q^c\,\omega + q^c\,\cos(\omega)\,\sin(\omega)}{2\omega} \end{pmatrix}. \tag{4.35}$$

The angular velocity is $\omega = 1/2$ and the spectral density is $q^c = 0.01$. The model is a discretized version of the noisy resonator model with a given angular velocity ω.

In the file[1] `kf_ex.m` there is simulation of the dynamic model together with a base line solution, where the measurement is directly used as the estimate of the state component x_1 and the second component x_2 is computed as a weighted average of the measurement differences.

(a) Implement the Kalman filter for the model and compare its performance (in the RMSE sense) to the base line solution. Plot figures of the solutions.

(b) Compute (numerically) the stationary Kalman filter corresponding to the model. Test this stationary filter against the base line and Kalman filter solutions. Plot the results and report the RMSE values for the solutions. What is the practical difference in the stationary and non-stationary Kalman filter solutions?

[1] The MATLAB® files are available on this book's web page `www.cambridge.org/sarkka/`

5

Extended and unscented Kalman filtering

It often happens in practical applications that the dynamic and measurement models are not linear and the Kalman filter is not appropriate. However, often the filtering distributions of this kind of model can be approximated by Gaussian distributions. In this chapter, three types of method for forming the Gaussian approximations are considered, the Taylor series based extended Kalman filters (EKF), the statistical linearization based statistically linearized filters (SLF), and the unscented transform based unscented Kalman filters (UKF). Among these, the UKF differs from the other filters in this section in the sense that it is not a series expansion based method per se – even though it was originally justified by considering a series expansion of the non-linear function. Actually, the relationship is even closer than that, because the UKF can be considered as an approximation of the SLF and it converges to the EKF in a suitable parameter limit.

5.1 Taylor series expansions

Consider the following transformation of a Gaussian random variable $\mathbf{x}$ into another random variable $\mathbf{y}$:

$$\begin{aligned} \mathbf{x} &\sim \mathrm{N}(\mathbf{m}, \mathbf{P}), \\ \mathbf{y} &= \mathbf{g}(\mathbf{x}), \end{aligned} \tag{5.1}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$, and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$ is a general non-linear function. Formally, the probability density of the random variable $\mathbf{y}$ is[1] (see, e.g., Gelman et al., 2004)

$$p(\mathbf{y}) = |\mathbf{J}(\mathbf{y})| \; \mathrm{N}(\mathbf{g}^{-1}(\mathbf{y}) \,|\, \mathbf{m}, \mathbf{P}), \tag{5.2}$$

[1] This actually only applies to invertible $\mathbf{g}(\cdot)$, but it can easily be generalized to the non-invertible case.

where $|\mathbf{J}(\mathbf{y})|$ is the determinant of the Jacobian matrix of the inverse transform $\mathbf{g}^{-1}(\mathbf{y})$. However, it is not generally possible to handle this distribution directly, because it is non-Gaussian for all but linear $\mathbf{g}$.

A first order Taylor series based Gaussian approximation to the distribution of $\mathbf{y}$ can now be formed as follows. If we let $\mathbf{x} = \mathbf{m} + \delta\mathbf{x}$, where $\delta\mathbf{x} \sim \mathrm{N}(\mathbf{0}, \mathbf{P})$, we can form the Taylor series expansion of the function $\mathbf{g}(\cdot)$ as follows (provided that the function is sufficiently differentiable):

$$\mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{m}+\delta\mathbf{x}) \approx \mathbf{g}(\mathbf{m}) + \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\delta\mathbf{x} + \sum_i \frac{1}{2}\delta\mathbf{x}^{\mathsf{T}}\,\mathbf{G}^{(i)}_{\mathbf{xx}}(\mathbf{m})\,\delta\mathbf{x}\,\mathbf{e}_i + \cdots, \tag{5.3}$$

where $\mathbf{G}_{\mathbf{x}}(\mathbf{m})$ is the Jacobian matrix of $\mathbf{g}$ with elements

$$[\mathbf{G}_{\mathbf{x}}(\mathbf{m})]_{j,j'} = \left.\frac{\partial g_j(\mathbf{x})}{\partial x_{j'}}\right|_{\mathbf{x}=\mathbf{m}}, \tag{5.4}$$

and $\mathbf{G}^{(i)}_{\mathbf{xx}}(\mathbf{m})$ is the Hessian matrix of $g_i(\cdot)$ evaluated at $\mathbf{m}$:

$$\left[\mathbf{G}^{(i)}_{\mathbf{xx}}(\mathbf{m})\right]_{j,j'} = \left.\frac{\partial^2 g_i(\mathbf{x})}{\partial x_j\,\partial x_{j'}}\right|_{\mathbf{x}=\mathbf{m}}. \tag{5.5}$$

Furthermore, $\mathbf{e}_i = (0\ \cdots\ 0\ 1\ 0\ \cdots\ 0)^{\mathsf{T}}$ is a vector with 1 at position i and all other elements zero, that is, it is the unit vector in the direction of the coordinate axis i.

The linear approximation can be obtained by approximating the function by the first two terms in the Taylor series:

$$\mathbf{g}(\mathbf{x}) \simeq \mathbf{g}(\mathbf{m}) + \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\delta\mathbf{x}. \tag{5.6}$$

Computing the expected value with respect to $\mathbf{x}$ gives:

$$\begin{aligned}\mathrm{E}[\mathbf{g}(\mathbf{x})] &\simeq \mathrm{E}[\mathbf{g}(\mathbf{m}) + \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\delta\mathbf{x}] \\ &= \mathbf{g}(\mathbf{m}) + \mathbf{G}_{\mathbf{x}}(\mathbf{m})\ \mathrm{E}[\delta\mathbf{x}] \\ &= \mathbf{g}(\mathbf{m}).\end{aligned} \tag{5.7}$$

The covariance can then be approximated as

$$\begin{aligned}
&\mathrm{E}\Big[(\mathbf{g}(\mathbf{x}) - \mathrm{E}[\mathbf{g}(\mathbf{x})])\ (\mathbf{g}(\mathbf{x}) - \mathrm{E}[\mathbf{g}(\mathbf{x})])^{\mathsf{T}}\Big] \\
&\quad \simeq \mathrm{E}\Big[(\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{m}))\ (\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{m}))^{\mathsf{T}}\Big] \\
&\quad \simeq \mathrm{E}\Big[(\mathbf{g}(\mathbf{m}) + \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\delta\mathbf{x} - \mathbf{g}(\mathbf{m}))\ (\mathbf{g}(\mathbf{m}) + \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\delta\mathbf{x} - \mathbf{g}(\mathbf{m}))^{\mathsf{T}}\Big] \\
&\quad = \mathrm{E}\Big[(\mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\delta\mathbf{x})\ (\mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\delta\mathbf{x})^{\mathsf{T}}\Big] \\
&\quad = \mathbf{G}_{\mathbf{x}}(\mathbf{m})\ \mathrm{E}\left[\delta\mathbf{x}\,\delta\mathbf{x}^{\mathsf{T}}\right]\mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}) \\
&\quad = \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\mathbf{P}\,\mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}).
\end{aligned} \tag{5.8}$$

We often are also interested in the the joint covariance between the variables $\mathbf{x}$ and $\mathbf{y}$. Approximation of the joint covariance can be achieved by considering the augmented transformation

$$\tilde{\mathbf{g}}(\mathbf{x}) = \begin{pmatrix} \mathbf{x} \\ \mathbf{g}(\mathbf{x}) \end{pmatrix}. \tag{5.9}$$

The resulting mean and covariance are

$$\begin{aligned}
\mathrm{E}[\tilde{\mathbf{g}}(\mathbf{x})] &\simeq \begin{pmatrix} \mathbf{m} \\ \mathbf{g}(\mathbf{m}) \end{pmatrix}, \\
\mathrm{Cov}[\tilde{\mathbf{g}}(\mathbf{x})] &\simeq \begin{pmatrix} \mathbf{I} \\ \mathbf{G}_{\mathbf{x}}(\mathbf{m}) \end{pmatrix} \mathbf{P} \begin{pmatrix} \mathbf{I} \\ \mathbf{G}_{\mathbf{x}}(\mathbf{m}) \end{pmatrix}^{\mathsf{T}} \\
&= \begin{pmatrix} \mathbf{P} & \mathbf{P}\,\mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}) \\ \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\mathbf{P} & \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,\mathbf{P}\,\mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}) \end{pmatrix}.
\end{aligned} \tag{5.10}$$

In the derivation of the extended Kalman filter equations, we need a slightly more general transformation of the form

$$\begin{aligned}
\mathbf{x} &\sim \mathrm{N}(\mathbf{m}, \mathbf{P}), \\
\mathbf{q} &\sim \mathrm{N}(\mathbf{0}, \mathbf{Q}), \\
\mathbf{y} &= \mathbf{g}(\mathbf{x}) + \mathbf{q},
\end{aligned} \tag{5.11}$$

where $\mathbf{q}$ is independent of $\mathbf{x}$. The joint distribution of $\mathbf{x}$ and $\mathbf{y}$, as defined above, is now the same as in Equations (5.10) except that the covariance $\mathbf{Q}$ is added to the lower right block of the covariance matrix of $\tilde{\mathbf{g}}(\cdot)$. Thus we get the following algorithm.

Algorithm 5.1 (Linear approximation of an additive transform) *The linear approximation based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}) + \mathbf{q}$*, where* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$*, is given as*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{L}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{L}} \\ \mathbf{C}_{\mathrm{L}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{L}} \end{pmatrix} \right), \tag{5.12}$$

where

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{L}} &= \mathbf{g}(\mathbf{m}), \\ \mathbf{S}_{\mathrm{L}} &= \mathbf{G}_{\mathbf{x}}(\mathbf{m})\, \mathbf{P}\, \mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}) + \mathbf{Q}, \\ \mathbf{C}_{\mathrm{L}} &= \mathbf{P}\, \mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}), \end{aligned} \tag{5.13}$$

and $\mathbf{G}_{\mathbf{x}}(\mathbf{m})$ *is the Jacobian matrix of* $\mathbf{g}$ *with respect to* $\mathbf{x}$*, evaluated at* $\mathbf{x} = \mathbf{m}$*, with elements*

$$[\mathbf{G}_{\mathbf{x}}(\mathbf{m})]_{j,j'} = \left. \frac{\partial g_j(\mathbf{x})}{\partial x_{j'}} \right|_{\mathbf{x}=\mathbf{m}}. \tag{5.14}$$

In filtering models where the process noise is not additive, we often need to approximate transformations of the form

$$\begin{aligned} \mathbf{x} &\sim \mathrm{N}(\mathbf{m}, \mathbf{P}), \\ \mathbf{q} &\sim \mathrm{N}(\mathbf{0}, \mathbf{Q}), \\ \mathbf{y} &= \mathbf{g}(\mathbf{x}, \mathbf{q}), \end{aligned} \tag{5.15}$$

where $\mathbf{x}$ and $\mathbf{q}$ are independent random variables. The mean and covariance can now be computed by substituting the augmented vector $(\mathbf{x}, \mathbf{q})$ for the vector $\mathbf{x}$ in Equation (5.10). The joint Jacobian matrix can then be written as $\mathbf{G}_{\mathbf{x},\mathbf{q}} = (\mathbf{G}_{\mathbf{x}}\ \mathbf{G}_{\mathbf{q}})$. Here $\mathbf{G}_{\mathbf{q}}$ is the Jacobian matrix of $\mathbf{g}(\cdot)$ with respect to $\mathbf{q}$ and both Jacobian matrices are evaluated at $\mathbf{x} = \mathbf{m}, \mathbf{q} = \mathbf{0}$. The approximations to the mean and covariance of the augmented transform as in Equation (5.10) are then given as

$$\begin{aligned} \mathrm{E}[\tilde{\mathbf{g}}(\mathbf{x}, \mathbf{q})] &\simeq \mathbf{g}(\mathbf{m}, \mathbf{0}), \\ \mathrm{Cov}[\tilde{\mathbf{g}}(\mathbf{x}, \mathbf{q})] &\simeq \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{G}_{\mathbf{x}}(\mathbf{m}) & \mathbf{G}_{\mathbf{q}}(\mathbf{m}) \end{pmatrix} \begin{pmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{G}_{\mathbf{x}}(\mathbf{m}) & \mathbf{G}_{\mathbf{q}}(\mathbf{m}) \end{pmatrix}^{\mathsf{T}} \\ &= \begin{pmatrix} \mathbf{P} & \mathbf{P}\, \mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}) \\ \mathbf{G}_{\mathbf{x}}(\mathbf{m})\, \mathbf{P} & \mathbf{G}_{\mathbf{x}}(\mathbf{m})\, \mathbf{P}\, \mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}) + \mathbf{G}_{\mathbf{q}}(\mathbf{m})\, \mathbf{Q}\, \mathbf{G}_{\mathbf{q}}^{\mathsf{T}}(\mathbf{m}) \end{pmatrix}. \end{aligned} \tag{5.16}$$

The approximation above can be formulated as the following algorithm.

Algorithm 5.2 (Linear approximation of a non-additive transform) *The linear approximation based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{q})$ *when* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$ *is given as*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{L}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{L}} \\ \mathbf{C}_{\mathrm{L}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{L}} \end{pmatrix} \right), \tag{5.17}$$

where

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{L}} &= \mathbf{g}(\mathbf{m}), \\ \mathbf{S}_{\mathrm{L}} &= \mathbf{G}_{\mathbf{x}}(\mathbf{m})\, \mathbf{P}\, \mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}) + \mathbf{G}_{\mathbf{q}}(\mathbf{m})\, \mathbf{Q}\, \mathbf{G}_{\mathbf{q}}^{\mathsf{T}}(\mathbf{m}), \\ \mathbf{C}_{\mathrm{L}} &= \mathbf{P}\, \mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}), \end{aligned} \tag{5.18}$$

$\mathbf{G}_{\mathbf{x}}(\mathbf{m})$ *is the Jacobian matrix of* $\mathbf{g}$ *with respect to* $\mathbf{x}$*, evaluated at* $\mathbf{x} = \mathbf{m}, \mathbf{q} = \mathbf{0}$*, with elements*

$$[\mathbf{G}_{\mathbf{x}}(\mathbf{m})]_{j,j'} = \left. \frac{\partial g_j(\mathbf{x}, \mathbf{q})}{\partial x_{j'}} \right|_{\mathbf{x}=\mathbf{m},\, \mathbf{q}=\mathbf{0}}, \tag{5.19}$$

and $\mathbf{G}_{\mathbf{q}}(\mathbf{m})$ *is the corresponding Jacobian matrix with respect to* $\mathbf{q}$*:*

$$\left[\mathbf{G}_{\mathbf{q}}(\mathbf{m})\right]_{j,j'} = \left. \frac{\partial g_j(\mathbf{x}, \mathbf{q})}{\partial q_{j'}} \right|_{\mathbf{x}=\mathbf{m},\, \mathbf{q}=\mathbf{0}}. \tag{5.20}$$

In quadratic approximations, in addition to the first order terms, the second order terms in the Taylor series expansion of the non-linear function are also retained.

Algorithm 5.3 (Quadratic approximation of an additive non-linear transform) *The second order approximation is of the form*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{Q}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{Q}} \\ \mathbf{C}_{\mathrm{Q}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{Q}} \end{pmatrix} \right), \tag{5.21}$$

where the parameters are

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{Q}} &= \mathbf{g}(\mathbf{m}) + \frac{1}{2} \sum_i \mathbf{e}_i\ \mathrm{tr}\left\{ \mathbf{G}_{\mathbf{xx}}^{(i)}(\mathbf{m})\, \mathbf{P} \right\}, \\ \mathbf{S}_{\mathrm{Q}} &= \mathbf{G}_{\mathbf{x}}(\mathbf{m})\, \mathbf{P}\, \mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}) + \frac{1}{2} \sum_{i,i'} \mathbf{e}_i\, \mathbf{e}_{i'}^{\mathsf{T}}\ \mathrm{tr}\left\{ \mathbf{G}_{\mathbf{xx}}^{(i)}(\mathbf{m})\, \mathbf{P}\, \mathbf{G}_{\mathbf{xx}}^{(i')}(\mathbf{m})\, \mathbf{P} \right\}, \\ \mathbf{C}_{\mathrm{Q}} &= \mathbf{P}\, \mathbf{G}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}), \end{aligned} \tag{5.22}$$

$\mathbf{G}_{\mathbf{x}}(\mathbf{m})$ *is the Jacobian matrix* (5.14)*, and* $\mathbf{G}_{\mathbf{xx}}^{(i)}(\mathbf{m})$ *is the Hessian matrix of* $g_i(\cdot)$ *evaluated at* $\mathbf{m}$*:*

$$\left[\mathbf{G}_{\mathbf{xx}}^{(i)}(\mathbf{m})\right]_{j,j'} = \left.\frac{\partial^2 g_i(\mathbf{x})}{\partial x_j \, \partial x_{j'}}\right|_{\mathbf{x}=\mathbf{m}}, \tag{5.23}$$

where $\mathbf{e}_i = (0 \cdots 0\ 1\ 0 \cdots 0)^\mathsf{T}$ *is a vector with 1 at position* i *and other elements zero, that is, it is the unit vector in the direction of the coordinate axis* i*.*

5.2 Extended Kalman filter

The extended Kalman filter (EKF) (see, e.g., Jazwinski, 1970; Maybeck, 1982b; Bar-Shalom et al., 2001; Grewal and Andrews, 2001) is an extension of the Kalman filter to non-linear filtering problems. If the process and measurement noises can be assumed to be additive, the EKF model can be written as

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k, \end{aligned} \tag{5.24}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state, $\mathbf{y}_k \in \mathbb{R}^m$ is the measurement, $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ is the Gaussian process noise, $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}_k)$ is the Gaussian measurement noise, $\mathbf{f}(\cdot)$ is the dynamic model function, and $\mathbf{h}(\cdot)$ is the measurement model function. The functions $\mathbf{f}$ and $\mathbf{h}$ can also depend on the step number k, but for notational convenience, this dependence has not been explicitly denoted.

The idea of the extended Kalman filter is to use (or assume) Gaussian approximations

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \simeq \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k) \tag{5.25}$$

to the filtering densities. In the EKF, these approximations are formed by utilizing Taylor series approximations to the non-linearities and the result is the following algorithm.

Algorithm 5.4 (Extended Kalman filter I) *The prediction and update steps of the first order additive noise extended Kalman filter (EKF) are:*

- *Prediction:*

$$\begin{aligned} \mathbf{m}_k^- &= \mathbf{f}(\mathbf{m}_{k-1}), \\ \mathbf{P}_k^- &= \mathbf{F}_{\mathbf{x}}(\mathbf{m}_{k-1})\, \mathbf{P}_{k-1}\, \mathbf{F}_{\mathbf{x}}^\mathsf{T}(\mathbf{m}_{k-1}) + \mathbf{Q}_{k-1}, \end{aligned} \tag{5.26}$$

- *Update:*

$$\begin{aligned}
\mathbf{v}_k &= \mathbf{y}_k - \mathbf{h}(\mathbf{m}_k^-), \\
\mathbf{S}_k &= \mathbf{H}_\mathbf{x}(\mathbf{m}_k^-)\,\mathbf{P}_k^-\,\mathbf{H}_\mathbf{x}^\mathsf{T}(\mathbf{m}_k^-) + \mathbf{R}_k, \\
\mathbf{K}_k &= \mathbf{P}_k^-\,\mathbf{H}_\mathbf{x}^\mathsf{T}(\mathbf{m}_k^-)\,\mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\,\mathbf{v}_k, \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\,\mathbf{S}_k\,\mathbf{K}_k^\mathsf{T}.
\end{aligned} \tag{5.27}$$

Derivation These filtering equations can be derived by repeating the same steps as in the derivation of the Kalman filter in Section 4.3 and by applying Taylor series approximations on the appropriate steps.

1 The joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ is non-Gaussian, but we can form a Gaussian approximation to it by applying the approximation Algorithm 5.1 to the function

$$\mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, \tag{5.28}$$

which results in the Gaussian approximation

$$p(\mathbf{x}_{k-1}, \mathbf{x}_k, \mid \mathbf{y}_{1:k-1}) \simeq \mathrm{N}\left(\begin{pmatrix}\mathbf{x}_{k-1}\\ \mathbf{x}_k\end{pmatrix} \,\middle|\, \mathbf{m}', \mathbf{P}'\right), \tag{5.29}$$

where

$$\begin{aligned}
\mathbf{m}' &= \begin{pmatrix}\mathbf{m}_{k-1}\\ \mathbf{f}(\mathbf{m}_{k-1})\end{pmatrix}, \\
\mathbf{P}' &= \begin{pmatrix}\mathbf{P}_{k-1} & \mathbf{P}_{k-1}\,\mathbf{F}_\mathbf{x}^\mathsf{T} \\ \mathbf{F}_\mathbf{x}\,\mathbf{P}_{k-1} & \mathbf{F}_\mathbf{x}\,\mathbf{P}_{k-1}\,\mathbf{F}_\mathbf{x}^\mathsf{T} + \mathbf{Q}_{k-1}\end{pmatrix},
\end{aligned} \tag{5.30}$$

and the Jacobian matrix $\mathbf{F}_\mathbf{x}$ of $\mathbf{f}(\mathbf{x})$ is evaluated at $\mathbf{x} = \mathbf{m}_{k-1}$. The approximations of the marginal mean and covariance of $\mathbf{x}_k$ are thus

$$\begin{aligned}
\mathbf{m}_k^- &= \mathbf{f}(\mathbf{m}_{k-1}), \\
\mathbf{P}_k^- &= \mathbf{F}_\mathbf{x}\,\mathbf{P}_{k-1}\,\mathbf{F}_\mathbf{x}^\mathsf{T} + \mathbf{Q}_{k-1}.
\end{aligned} \tag{5.31}$$

2 The joint distribution of $\mathbf{y}_k$ and $\mathbf{x}_k$ is also non-Gaussian, but we can again approximate it by applying Algorithm 5.1 to the function

$$\mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k. \tag{5.32}$$

We get the approximation

$$p(\mathbf{x}_k, \mathbf{y}_k \mid \mathbf{y}_{1:k-1}) \simeq \mathrm{N}\left(\begin{pmatrix}\mathbf{x}_k\\ \mathbf{y}_k\end{pmatrix} \,\middle|\, \mathbf{m}'', \mathbf{P}''\right), \tag{5.33}$$

where

$$\mathbf{m}'' = \begin{pmatrix} \mathbf{m}_k^- \\ \mathbf{h}(\mathbf{m}_k^-) \end{pmatrix}, \qquad \mathbf{P}'' = \begin{pmatrix} \mathbf{P}_k^- & \mathbf{P}_k^- \, \mathbf{H}_\mathbf{x}^\mathsf{T} \\ \mathbf{H}_\mathbf{x} \, \mathbf{P}_k^- & \mathbf{H}_\mathbf{x} \, \mathbf{P}_k^- \, \mathbf{H}_\mathbf{x}^\mathsf{T} + \mathbf{R}_k \end{pmatrix}, \quad (5.34)$$

and the Jacobian matrix $\mathbf{H}_\mathbf{x}$ of $\mathbf{h}(\mathbf{x})$ is evaluated at $\mathbf{x} = \mathbf{m}_k^-$.

3 By Lemma A.2 the conditional distribution of $\mathbf{x}_k$ is approximately

$$p(\mathbf{x}_k \mid \mathbf{y}_k, \mathbf{y}_{1:k-1}) \simeq \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k), \quad (5.35)$$

where

$$\begin{aligned} \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{P}_k^- \, \mathbf{H}_\mathbf{x}^\mathsf{T} \, (\mathbf{H}_\mathbf{x} \, \mathbf{P}_k^- \, \mathbf{H}_\mathbf{x}^\mathsf{T} + \mathbf{R}_k)^{-1} \, [\mathbf{y}_k - \mathbf{h}(\mathbf{m}_k^-)], \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{P}_k^- \, \mathbf{H}_\mathbf{x}^\mathsf{T} \, (\mathbf{H}_\mathbf{x} \, \mathbf{P}_k^- \, \mathbf{H}_\mathbf{x}^\mathsf{T} + \mathbf{R}_k)^{-1} \, \mathbf{H}_\mathbf{x} \, \mathbf{P}_k^-. \end{aligned} \quad (5.36)$$

□

A more general non-additive noise EKF filtering model can be written as

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}), \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{r}_k), \end{aligned} \quad (5.37)$$

where $\mathbf{q}_{k-1} \sim \mathrm{N}(0, \mathbf{Q}_{k-1})$ and $\mathbf{r}_k \sim \mathrm{N}(0, \mathbf{R}_k)$ are the Gaussian process and measurement noises, respectively. Again, the functions $\mathbf{f}$ and $\mathbf{h}$ can also depend on the step number k. The EKF algorithm for the above model is the following.

Algorithm 5.5 (Extended Kalman filter II) *The prediction and update steps of the (first order) extended Kalman filter (EKF) in the non-additive noise case are:*

- *Prediction:*

$$\begin{aligned} \mathbf{m}_k^- &= \mathbf{f}(\mathbf{m}_{k-1}, \mathbf{0}), \\ \mathbf{P}_k^- &= \mathbf{F}_\mathbf{x}(\mathbf{m}_{k-1}) \, \mathbf{P}_{k-1} \, \mathbf{F}_\mathbf{x}^\mathsf{T}(\mathbf{m}_{k-1}) + \mathbf{F}_\mathbf{q}(\mathbf{m}_{k-1}) \, \mathbf{Q}_{k-1} \, \mathbf{F}_\mathbf{q}^\mathsf{T}(\mathbf{m}_{k-1}), \end{aligned} \quad (5.38)$$

- *Update:*

$$\begin{aligned} \mathbf{v}_k &= \mathbf{y}_k - \mathbf{h}(\mathbf{m}_k^-, \mathbf{0}), \\ \mathbf{S}_k &= \mathbf{H}_\mathbf{x}(\mathbf{m}_k^-) \, \mathbf{P}_k^- \, \mathbf{H}_\mathbf{x}^\mathsf{T}(\mathbf{m}_k^-) + \mathbf{H}_\mathbf{r}(\mathbf{m}_k^-) \, \mathbf{R}_k \, \mathbf{H}_\mathbf{r}^\mathsf{T}(\mathbf{m}_k^-), \\ \mathbf{K}_k &= \mathbf{P}_k^- \, \mathbf{H}_\mathbf{x}^\mathsf{T}(\mathbf{m}_k^-) \, \mathbf{S}_k^{-1}, \\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k \, \mathbf{v}_k, \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k \, \mathbf{S}_k \, \mathbf{K}_k^\mathsf{T}, \end{aligned} \quad (5.39)$$

where the matrices $\mathbf{F_x}(\mathbf{m})$, $\mathbf{F_q}(\mathbf{m})$, $\mathbf{H_x}(\mathbf{m})$, *and* $\mathbf{H_r}(\mathbf{m})$ *are the Jacobian matrices of* $\mathbf{f}$ *and* $\mathbf{h}$ *with respect to state and noise, with elements*

$$[\mathbf{F_x}(\mathbf{m})]_{j,j'} = \left.\frac{\partial f_j(\mathbf{x},\mathbf{q})}{\partial x_{j'}}\right|_{\mathbf{x}=\mathbf{m},\,\mathbf{q}=\mathbf{0}}, \tag{5.40}$$

$$\left[\mathbf{F_q}(\mathbf{m})\right]_{j,j'} = \left.\frac{\partial f_j(\mathbf{x},\mathbf{q})}{\partial q_{j'}}\right|_{\mathbf{x}=\mathbf{m},\,\mathbf{q}=\mathbf{0}}, \tag{5.41}$$

$$[\mathbf{H_x}(\mathbf{m})]_{j,j'} = \left.\frac{\partial h_j(\mathbf{x},\mathbf{r})}{\partial x_{j'}}\right|_{\mathbf{x}=\mathbf{m},\,\mathbf{r}=\mathbf{0}}, \tag{5.42}$$

$$[\mathbf{H_r}(\mathbf{m})]_{j,j'} = \left.\frac{\partial h_j(\mathbf{x},\mathbf{r})}{\partial r_{j'}}\right|_{\mathbf{x}=\mathbf{m},\,\mathbf{r}=\mathbf{0}}. \tag{5.43}$$

Derivation These filtering equations can be derived by repeating the same steps as in the derivation of the extended Kalman filter above, but instead of using Algorithm 5.1, we use Algorithm 5.2 for computing the approximations. □

The advantage of the EKF over other non-linear filtering methods is its relative simplicity compared to its performance. Linearization is a very common engineering way of constructing approximations to non-linear systems and thus it is very easy to understand and apply. A disadvantage is that because it is based on a local linear approximation, it will not work in problems with considerable non-linearities. The filtering model is also restricted in the sense that only Gaussian noise processes are allowed and thus the model cannot contain, for example, discrete valued random variables. The Gaussian restriction also prevents the handling of hierarchical models or other models where significantly non-Gaussian distribution models would be needed.

The EKF also requires the measurement model and the dynamic model functions to be differentiable, which is a restriction. In some cases it might also be simply impossible to compute the required Jacobian matrices, which renders the use of the EKF impossible. Even when the Jacobian matrices exist and could be computed, the actual computation and programming of Jacobian matrices can be quite error prone and hard to debug.

In the so-called second order EKF the non-linearity is approximated by retaining the second order terms in the Taylor series expansion as in Algorithm 5.3. The resulting algorithm is the following.

Algorithm 5.6 (Extended Kalman filter III) *The prediction and update steps of the second order extended Kalman filter (in the additive noise case) are:*

- *Prediction:*

$$\begin{aligned}
\mathbf{m}_k^- &= \mathbf{f}(\mathbf{m}_{k-1}) + \frac{1}{2}\sum_i \mathbf{e}_i \ \mathrm{tr}\left\{\mathbf{F}_{\mathbf{xx}}^{(i)}(\mathbf{m}_{k-1})\,\mathbf{P}_{k-1}\right\},\\
\mathbf{P}_k^- &= \mathbf{F}_{\mathbf{x}}(\mathbf{m}_{k-1})\,\mathbf{P}_{k-1}\,\mathbf{F}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}_{k-1})\\
&\quad + \frac{1}{2}\sum_{i,i'} \mathbf{e}_i\,\mathbf{e}_{i'}^{\mathsf{T}}\,\mathrm{tr}\left\{\mathbf{F}_{\mathbf{xx}}^{(i)}(\mathbf{m}_{k-1})\mathbf{P}_{k-1}\mathbf{F}_{\mathbf{xx}}^{(i')}(\mathbf{m}_{k-1})\mathbf{P}_{k-1}\right\} + \mathbf{Q}_{k-1},
\end{aligned} \tag{5.44}$$

- *Update:*

$$\begin{aligned}
\mathbf{v}_k &= \mathbf{y}_k - \mathbf{h}(\mathbf{m}_k^-) - \frac{1}{2}\sum_i \mathbf{e}_i \ \mathrm{tr}\left\{\mathbf{H}_{\mathbf{xx}}^{(i)}(\mathbf{m}_k^-)\,\mathbf{P}_k^-\right\},\\
\mathbf{S}_k &= \mathbf{H}_{\mathbf{x}}(\mathbf{m}_k^-)\,\mathbf{P}_k^-\,\mathbf{H}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}_k^-)\\
&\quad + \frac{1}{2}\sum_{i,i'} \mathbf{e}_i\,\mathbf{e}_{i'}^{\mathsf{T}}\ \mathrm{tr}\left\{\mathbf{H}_{\mathbf{xx}}^{(i)}(\mathbf{m}_k^-)\,\mathbf{P}_k^-\,\mathbf{H}_{\mathbf{xx}}^{(i')}(\mathbf{m}_k^-)\,\mathbf{P}_k^-\right\} + \mathbf{R}_k,\\
\mathbf{K}_k &= \mathbf{P}_k^-\,\mathbf{H}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}_k^-)\,\mathbf{S}_k^{-1},\\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\,\mathbf{v}_k,\\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\,\mathbf{S}_k\,\mathbf{K}_k^{\mathsf{T}},
\end{aligned} \tag{5.45}$$

where the matrices $\mathbf{F}_{\mathbf{x}}(\mathbf{m})$ *and* $\mathbf{H}_{\mathbf{x}}(\mathbf{m})$ *are given by Equations* (5.40) *and* (5.42)*. The matrices* $\mathbf{F}_{\mathbf{xx}}^{(i)}(\mathbf{m})$ *and* $\mathbf{H}_{\mathbf{xx}}^{(i)}(\mathbf{m})$ *are the Hessian matrices of* f_i *and* h_i *respectively:*

$$\left[\mathbf{F}_{\mathbf{xx}}^{(i)}(\mathbf{m})\right]_{j,j'} = \left.\frac{\partial^2 f_i(\mathbf{x})}{\partial x_j\,\partial x_{j'}}\right|_{\mathbf{x}=\mathbf{m}}, \tag{5.46}$$

$$\left[\mathbf{H}_{\mathbf{xx}}^{(i)}(\mathbf{m})\right]_{j,j'} = \left.\frac{\partial^2 h_i(\mathbf{x})}{\partial x_j\,\partial x_{j'}}\right|_{\mathbf{x}=\mathbf{m}}. \tag{5.47}$$

The non-additive version can be derived in an analogous manner, but due to its complicated appearance, it is not presented here.

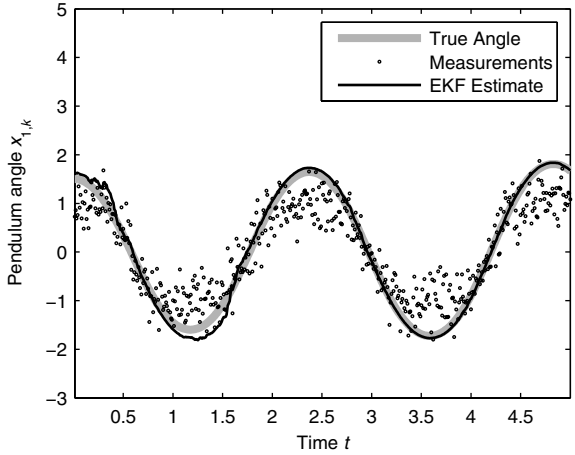


Figure 5.1 Simulated pendulum data and the result of tracking the pendulum angle and angular rate with the EKF in Example 5.1. The resulting RMSE was 0.12.

Example 5.1 (Pendulum tracking with EKF) *A simple discretization of the pendulum model in Example 3.7 leads to the following model:*

$$\underbrace{\begin{pmatrix} x_{1,k} \\ x_{2,k} \end{pmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{pmatrix} x_{1,k-1} + x_{2,k-1}\,\Delta t \\ x_{2,k-1} - g\,\sin(x_{1,k-1})\,\Delta t \end{pmatrix}}_{\mathbf{f}(\mathbf{x}_{k-1})} + \mathbf{q}_{k-1},$$

$$y_k = \underbrace{\sin(x_{1,k})}_{h(\mathbf{x}_k)} + r_k, \tag{5.48}$$

where $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$ *and* $r_k \sim \mathrm{N}(0, R)$ *with*

$$\mathbf{Q} = \begin{pmatrix} \frac{q^{\mathrm{c}}\,\Delta t^3}{3} & \frac{q^{\mathrm{c}}\,\Delta t^2}{2} \\ \frac{q^{\mathrm{c}}\,\Delta t^2}{2} & q^{\mathrm{c}}\,\Delta t \end{pmatrix}, \tag{5.49}$$

and q^{c} *is the spectral density of the continuous-time process noise. The required Jacobian matrices of* $\mathbf{f}(\mathbf{x})$ *and* $h(\mathbf{x})$ *for the first order EKF are:*

$$\mathbf{F}_x(\mathbf{x}) = \begin{pmatrix} 1 & \Delta t \\ -g\,\cos(x_1)\,\Delta t & 1 \end{pmatrix}, \qquad \mathbf{H}_x(\mathbf{x}) = \begin{pmatrix} \cos(x_1) & 0 \end{pmatrix}. \tag{5.50}$$

An example result of applying the EKF to simulated data from the pendulum model is shown in Figure 5.1. The EKF is indeed able to track the pendulum angle quite well despite the non-linearity of the model. The RMSE

in the angle is 0.12 *which is much lower than the standard deviation of the measurement noise which was* 0.32.

5.3 Statistical linearization

In the statistically linearized filter (Gelb, 1974) the first order Taylor series approximation used in the first order EKF is replaced by statistical linearization. Recall the transformation problem considered in Section 5.1, which was stated as

$$\begin{aligned}\mathbf{x} &\sim \mathrm{N}(\mathbf{m}, \mathbf{P}),\\ \mathbf{y} &= \mathbf{g}(\mathbf{x}).\end{aligned}$$

In statistical linearization we form a linear approximation to the transformation as follows:

$$\mathbf{g}(\mathbf{x}) \simeq \mathbf{b} + \mathbf{A}\,\delta\mathbf{x}, \tag{5.51}$$

where $\delta\mathbf{x} = \mathbf{x} - \mathbf{m}$, such that the mean squared error is minimized:

$$\mathrm{MSE}(\mathbf{b}, \mathbf{A}) = \mathrm{E}[(\mathbf{g}(\mathbf{x}) - \mathbf{b} - \mathbf{A}\,\delta\mathbf{x})^{\mathsf{T}}(\mathbf{g}(\mathbf{x}) - \mathbf{b} - \mathbf{A}\,\delta\mathbf{x})]. \tag{5.52}$$

Setting the derivatives with respect to $\mathbf{b}$ and $\mathbf{A}$ to zero gives

$$\begin{aligned}\mathbf{b} &= \mathrm{E}[\mathbf{g}(\mathbf{x})],\\ \mathbf{A} &= \mathrm{E}[\mathbf{g}(\mathbf{x})\,\delta\mathbf{x}^{\mathsf{T}}]\,\mathbf{P}^{-1}.\end{aligned} \tag{5.53}$$

In this approximation to the transform $\mathbf{g}(\mathbf{x})$, $\mathbf{b}$ is now exactly the mean and the approximate covariance is given as

$$\begin{aligned}&\mathrm{E}[(\mathbf{g}(\mathbf{x}) - \mathrm{E}[\mathbf{g}(\mathbf{x})])\ (\mathbf{g}(\mathbf{x}) - \mathrm{E}[\mathbf{g}(\mathbf{x})])^{\mathsf{T}}]\\ &\quad\simeq \mathbf{A}\,\mathbf{P}\,\mathbf{A}^{\mathsf{T}}\\ &\quad= \mathrm{E}[\mathbf{g}(\mathbf{x})\,\delta\mathbf{x}^{\mathsf{T}}]\,\mathbf{P}^{-1}\ \mathrm{E}[\mathbf{g}(\mathbf{x})\,\delta\mathbf{x}^{\mathsf{T}}]^{\mathsf{T}}.\end{aligned} \tag{5.54}$$

We may now apply this approximation to the augmented function $\tilde{\mathbf{g}}(\mathbf{x}) = (\mathbf{x}, \mathbf{g}(\mathbf{x}))$ in Equation (5.9) of Section 5.1, where we get the approximations

$$\begin{aligned}\mathrm{E}[\tilde{\mathbf{g}}(\mathbf{x})] &\simeq \begin{pmatrix}\mathbf{m}\\ \mathrm{E}[\mathbf{g}(\mathbf{x})]\end{pmatrix},\\ \mathrm{Cov}[\tilde{\mathbf{g}}(\mathbf{x})] &\simeq \begin{pmatrix}\mathbf{P} & \mathrm{E}[\mathbf{g}(\mathbf{x})\,\delta\mathbf{x}^{\mathsf{T}}]^{\mathsf{T}}\\ \mathrm{E}[\mathbf{g}(\mathbf{x})\,\delta\mathbf{x}^{\mathsf{T}}] & \mathrm{E}[\mathbf{g}(\mathbf{x})\,\delta\mathbf{x}^{\mathsf{T}}]\,\mathbf{P}^{-1}\ \mathrm{E}[\mathbf{g}(\mathbf{x})\,\delta\mathbf{x}^{\mathsf{T}}]^{\mathsf{T}}\end{pmatrix}.\end{aligned} \tag{5.55}$$

Thus we get the following algorithm.

Algorithm 5.7 (Statistically linearized approximation of an additive transform) *The statistical linearization based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}) + \mathbf{q}$, *where* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$, *is given as*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{S}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{S}} \\ \mathbf{C}_{\mathrm{S}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{S}} \end{pmatrix} \right), \tag{5.56}$$

where

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{S}} &= \mathrm{E}[\mathbf{g}(\mathbf{x})], \\ \mathbf{S}_{\mathrm{S}} &= \mathrm{E}[\mathbf{g}(\mathbf{x})\, \delta\mathbf{x}^{\mathsf{T}}]\, \mathbf{P}^{-1}\, \mathrm{E}[\mathbf{g}(\mathbf{x})\, \delta\mathbf{x}^{\mathsf{T}}]^{\mathsf{T}} + \mathbf{Q}, \\ \mathbf{C}_{\mathrm{S}} &= \mathrm{E}[\mathbf{g}(\mathbf{x})\, \delta\mathbf{x}^{\mathsf{T}}]^{\mathsf{T}}. \end{aligned} \tag{5.57}$$

The expectations are taken with respect to the distribution of $\mathbf{x}$.

Applying the same approximation with $(\mathbf{x}, \mathbf{q})$ in place of $\mathbf{x}$ we obtain the following mean and covariance for the non-additive transform:

$$\begin{aligned} \mathrm{E}[\tilde{\mathbf{g}}(\mathbf{x}, \mathbf{q})] &\simeq \begin{pmatrix} \mathbf{m} \\ \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})] \end{pmatrix}, \\ \mathrm{Cov}[\tilde{\mathbf{g}}(\mathbf{x}, \mathbf{q})] &\simeq \begin{pmatrix} \mathbf{P} & \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \delta\mathbf{x}^{\mathsf{T}}]^{\mathsf{T}} \\ \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \delta\mathbf{x}^{\mathsf{T}}] & \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \delta\mathbf{x}^{\mathsf{T}}]\, \mathbf{P}^{-1}\, \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \delta\mathbf{x}^{\mathsf{T}}]^{\mathsf{T}} \\ & \quad + \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \mathbf{q}^{\mathsf{T}}]\, \mathbf{Q}^{-1}\, \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \mathbf{q}^{\mathsf{T}}]^{\mathsf{T}} \end{pmatrix}. \end{aligned} \tag{5.58}$$

Thus we get the following algorithm for the non-additive transform.

Algorithm 5.8 (Statistically linearized approximation of a non-additive transform) *The statistical linearization based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{q})$ *when* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$ *is given as*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{S}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{S}} \\ \mathbf{C}_{\mathrm{S}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{S}} \end{pmatrix} \right), \tag{5.59}$$

where

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{S}} &= \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})], \\ \mathbf{S}_{\mathrm{S}} &= \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \delta\mathbf{x}^{\mathsf{T}}]\, \mathbf{P}^{-1}\, \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \delta\mathbf{x}^{\mathsf{T}}]^{\mathsf{T}} \\ &\quad + \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \mathbf{q}^{\mathsf{T}}]\, \mathbf{Q}^{-1}\, \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \mathbf{q}^{\mathsf{T}}]^{\mathsf{T}}, \\ \mathbf{C}_{\mathrm{S}} &= \mathrm{E}[\mathbf{g}(\mathbf{x}, \mathbf{q})\, \delta\mathbf{x}^{\mathsf{T}}]^{\mathsf{T}}. \end{aligned} \tag{5.60}$$

The expectations are taken with respect to the variables $\mathbf{x}$ *and* $\mathbf{q}$.

If the function $\mathbf{g}(\mathbf{x})$ is differentiable, it is possible to use the following well-known property of Gaussian random variables for simplifying the expressions:

$$\mathrm{E}[\mathbf{g}(\mathbf{x})\,(\mathbf{x}-\mathbf{m})^{\mathsf{T}}] = \mathrm{E}[\mathbf{G}_{\mathbf{x}}(\mathbf{x})]\,\mathbf{P}, \tag{5.61}$$

where $\mathrm{E}[\cdot]$ denotes the expected value with respect to $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$, and $\mathbf{G}_{\mathbf{x}}(\mathbf{x})$ is the Jacobian matrix of $\mathbf{g}(\mathbf{x})$. The statistical linearization equations then reduce to the same form as Taylor series based linearization, except that instead of the Jacobians we have the expected values of the Jacobians (see the exercises). Algorithm 5.7 can be then written in the following form.

Algorithm 5.9 (Statistically linearized approximation of an additive transform II) *The statistical linearization based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}) + \mathbf{q}$, *where* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$, *can be written as*

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix}\mathbf{m}\\ \boldsymbol{\mu}_{\mathrm{S}}\end{pmatrix}, \begin{pmatrix}\mathbf{P} & \mathbf{C}_{\mathrm{S}}\\ \mathbf{C}_{\mathrm{S}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{S}}\end{pmatrix}\right), \tag{5.62}$$

where

$$\begin{aligned}\boldsymbol{\mu}_{\mathrm{S}} &= \mathrm{E}[\mathbf{g}(\mathbf{x})],\\ \mathbf{S}_{\mathrm{S}} &= \mathrm{E}[\mathbf{G}_{\mathbf{x}}(\mathbf{x})]\,\mathbf{P}\;\mathrm{E}[\mathbf{G}_{\mathbf{x}}(\mathbf{x})]^{\mathsf{T}} + \mathbf{Q},\\ \mathbf{C}_{\mathrm{S}} &= \mathbf{P}\;\mathrm{E}[\mathbf{G}_{\mathbf{x}}(\mathbf{x})]^{\mathsf{T}},\end{aligned} \tag{5.63}$$

and $\mathbf{G}_{\mathbf{x}}(\mathbf{x})$ *is the Jacobian matrix of* $\mathbf{g}$. *The expectations are taken with respect to the distribution of* $\mathbf{x}$.

Note that we actually only need to compute the expectation $\mathrm{E}[\mathbf{g}(\mathbf{x})]$, because if we know the function

$$\boldsymbol{\mu}_{\mathrm{S}}(\mathbf{m}) = \mathrm{E}[\mathbf{g}(\mathbf{x})], \tag{5.64}$$

where $\mathrm{E}[\cdot]$ denotes the expected value with respect to $\mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})$, then

$$\frac{\partial \boldsymbol{\mu}_{\mathrm{S}}(\mathbf{m})}{\partial \mathbf{m}} = \mathrm{E}[\mathbf{G}_{\mathbf{x}}(\mathbf{x})]. \tag{5.65}$$

5.4 Statistically linearized filter

The statistically linearized filter (SLF) (Gelb, 1974) or quasi-linear filter (Stengel, 1994) is a Gaussian approximation based filter which can be applied to the same kind of models as the EKF, that is, to models of the form

(5.24) or (5.37). The filter is similar to the EKF, except that statistical linearizations in Algorithms 5.7, 5.8, and 5.9 are used instead of the Taylor series approximations.

Algorithm 5.10 (Statistically linearized filter I) *The prediction and update steps of the additive noise statistically linearized (Kalman) filter are:*

- *Prediction:*

$$\begin{aligned}\mathbf{m}_k^- &= \mathrm{E}[\mathbf{f}(\mathbf{x}_{k-1})],\\ \mathbf{P}_k^- &= \mathrm{E}[\mathbf{f}(\mathbf{x}_{k-1})\,\delta\mathbf{x}_{k-1}^\mathsf{T}]\,\mathbf{P}_{k-1}^{-1}\,\mathrm{E}[\mathbf{f}(\mathbf{x}_{k-1})\,\delta\mathbf{x}_{k-1}^\mathsf{T}]^\mathsf{T} + \mathbf{Q}_{k-1},\end{aligned} \tag{5.66}$$

  where $\delta\mathbf{x}_{k-1} = \mathbf{x}_{k-1} - \mathbf{m}_{k-1}$ *and the expectations are taken with respect to the variable* $\mathbf{x}_{k-1} \sim \mathrm{N}(\mathbf{m}_{k-1}, \mathbf{P}_{k-1})$*,*

- *Update:*

$$\begin{aligned}\mathbf{v}_k &= \mathbf{y}_k - \mathrm{E}[\mathbf{h}(\mathbf{x}_k)],\\ \mathbf{S}_k &= \mathrm{E}[\mathbf{h}(\mathbf{x}_k)\,\delta\tilde{\mathbf{x}}_k^\mathsf{T}]\,(\mathbf{P}_k^-)^{-1}\,\mathrm{E}[\mathbf{h}(\mathbf{x}_k)\,\delta\tilde{\mathbf{x}}_k^\mathsf{T}]^\mathsf{T} + \mathbf{R}_k,\\ \mathbf{K}_k &= \mathrm{E}[\mathbf{h}(\mathbf{x}_k)\,\delta\tilde{\mathbf{x}}_k^\mathsf{T}]^\mathsf{T}\,\mathbf{S}_k^{-1},\\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\,\mathbf{v}_k,\\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\,\mathbf{S}_k\,\mathbf{K}_k^\mathsf{T},\end{aligned} \tag{5.67}$$

  where $\delta\tilde{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{m}_k^-$ *and the expectations are taken with respect to the variable* $\mathbf{x}_k \sim \mathrm{N}(\mathbf{m}_k^-, \mathbf{P}_k^-)$*.*

The above filter can also be rewritten using the expectations of Jacobians by using Algorithm 5.9 instead of 5.7 (see the exercises).

Algorithm 5.11 (Statistically linearized filter II) *The prediction and update steps of the non-additive statistically linearized (Kalman) filter are:*

- *Prediction:*

$$\begin{aligned}\mathbf{m}_k^- &= \mathrm{E}[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})],\\ \mathbf{P}_k^- &= \mathrm{E}[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})\,\delta\mathbf{x}_{k-1}^\mathsf{T}]\,\mathbf{P}_{k-1}^{-1}\,\mathrm{E}[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})\,\delta\mathbf{x}_{k-1}^\mathsf{T}]^\mathsf{T}\\ &\quad + \mathrm{E}[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})\,\mathbf{q}_{k-1}^\mathsf{T}]\,\mathbf{Q}_{k-1}^{-1}\,\mathrm{E}[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})\,\mathbf{q}_{k-1}^\mathsf{T}]^\mathsf{T},\end{aligned} \tag{5.68}$$

  where $\delta\mathbf{x}_{k-1} = \mathbf{x}_{k-1} - \mathbf{m}_{k-1}$ *and the expectations are taken with respect to the variables* $\mathbf{x}_{k-1} \sim \mathrm{N}(\mathbf{m}_{k-1}, \mathbf{P}_{k-1})$ *and* $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_{k-1})$*,*

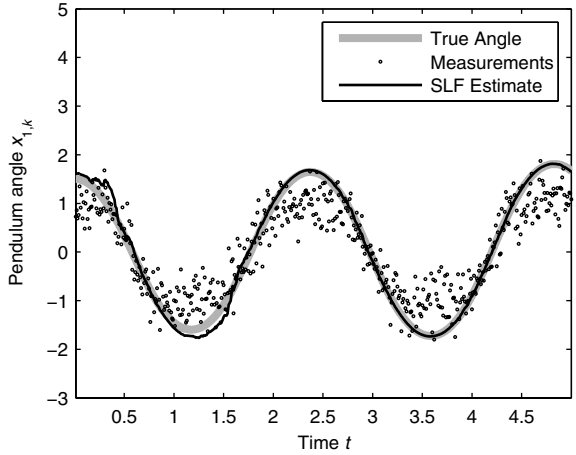


Figure 5.2 Simulated pendulum data and the result of tracking the pendulum angle and angular rate with the SLF in Example 5.2. The resulting RMSE was 0.11 which is slightly lower than the error of EKF (Figure 5.1), but still practically the same.

- *Update:*

$$
\begin{aligned}
\mathbf{v}_k &= \mathbf{y}_k - \mathrm{E}[\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k)], \\
\mathbf{S}_k &= \mathrm{E}[\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k)\, \delta\tilde{\mathbf{x}}_k^\mathsf{T}]\, (\mathbf{P}_k^-)^{-1}\, \mathrm{E}[\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k)\, \delta\tilde{\mathbf{x}}_k^\mathsf{T}]^\mathsf{T} \\
&\quad + \mathrm{E}[\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k)\, \mathbf{r}_k^\mathsf{T}]\, \mathbf{R}_k^{-1}\, \mathrm{E}[\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k)\, \mathbf{r}_k^\mathsf{T}]^\mathsf{T}, \\
\mathbf{K}_k &= \mathrm{E}[\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k)\, \delta\tilde{\mathbf{x}}_k^\mathsf{T}]^\mathsf{T}\, \mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\, \mathbf{v}_k, \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\, \mathbf{S}_k\, \mathbf{K}_k^\mathsf{T},
\end{aligned}
\tag{5.69}
$$

where $\delta\tilde{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{m}_k^-$ *and the expectations are taken with respect to the variables* $\mathbf{x}_k \sim \mathrm{N}(\mathbf{m}_k^-, \mathbf{P}_k^-)$ *and* $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}_k)$.

Both of the filters above can be derived by following the derivation of the EKF in Section 5.2 and by utilizing the statistical linearization approximations instead of the linear approximations on the appropriate steps.

Example 5.2 (Pendulum tracking with SLF) *The expectations of* $\mathbf{f}$ *and* h *which are required for implementation of the additive noise SLF for the*

pendulum model in Example 5.2 can be computed as follows:

$$\mathrm{E}[\mathbf{f}(\mathbf{x})] = \begin{pmatrix} m_1 + m_2\,\Delta t \\ m_2 - g\,\sin(m_1)\,\exp(-P_{11}/2)\,\Delta t \end{pmatrix},$$
$$\mathrm{E}[h(\mathbf{x})] = \sin(m_1)\,\exp(-P_{11}/2). \tag{5.70}$$

The required cross-correlation terms are

$$\mathrm{E}[\mathbf{f}(\mathbf{x})\,(\mathbf{x}-\mathbf{m})^{\mathsf{T}}] = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}, \tag{5.71}$$

where

$$\begin{aligned} c_{11} &= P_{11} + \Delta t\; P_{12}, \\ c_{12} &= P_{12} + \Delta t\; P_{22}, \\ c_{21} &= P_{12} - g\,\Delta t\;\cos(m_1)\,P_{11}\,\exp(-P_{11}/2), \\ c_{22} &= P_{22} - g\,\Delta t\;\cos(m_1)\,P_{12}\,\exp(-P_{11}/2), \end{aligned} \tag{5.72}$$

and

$$\mathrm{E}[h(\mathbf{x})\,(\mathbf{x}-\mathbf{m})^{\mathsf{T}}] = \begin{pmatrix} \cos(m_1)\,P_{11}\,\exp(-P_{11}/2) \\ \cos(m_1)\,P_{12}\,\exp(-P_{11}/2) \end{pmatrix}. \tag{5.73}$$

The above computations reveal the main weakness of statistical linearization based filtering – there is no hope of computing the above expectations when the functions are complicated. In this case we were lucky, because the only non-linearities in the dynamic and measurement models were sinusoidal, for which the closed form expectations can be computed.

The result of applying the SLF to the same simulated pendulum data as was used with the EKF in Figure 5.1 is shown in Figure 5.2. The result of the SLF is practically the same as that of the EKF. However, the RMSE of the SLF is slightly lower than that of the EKF.

The advantage of the SLF over the EKF is that it is a more global approximation than the EKF, because the linearization is not only based on the local region around the mean but on a whole range of function values. The non-linearities also do not have to be differentiable. However, if the non-linearities are differentiable, then we can use the Gaussian random variable property (5.61) for rewriting the equations in an EKF-like form. The clear disadvantage of the SLF over the EKF is that the expected values of the non-linear functions have to be computed in closed form. Naturally, it is not possible for all functions. Fortunately, the expected values involved are of such a type that one is likely to find many of them tabulated in older

physics and control engineering books (see, e.g., Gelb and Vander Velde, 1968).

The statistically linearized filter (SLF) is a special case of the Fourier–Hermite Kalman filter (FHKF), when the first order truncation of the series is used (Sarmavuori and Särkkä, 2012a). Many of the sigma-point methods can also be interpreted as approximations to the Fourier–Hermite Kalman filters and statistically linearized filters (see Van der Merwe and Wan, 2003; Särkkä and Hartikainen, 2010b; Sarmavuori and Särkkä, 2012a).

5.5 Unscented transform

The *unscented transform* (UT) (Julier and Uhlmann, 1995; Julier et al., 2000) is a relatively recent numerical method that can also be used for approximating the joint distribution of random variables $\mathbf{x}$ and $\mathbf{y}$ defined as

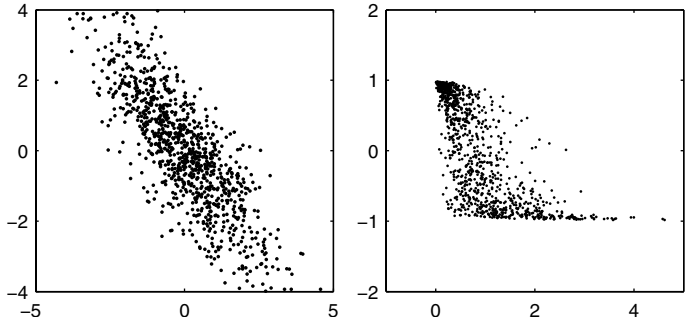
$$\begin{aligned}\mathbf{x} &\sim \mathrm{N}(\mathbf{m}, \mathbf{P}),\\ \mathbf{y} &= \mathbf{g}(\mathbf{x}).\end{aligned}$$

However, the philosophy of the UT differs from linearization and statistical linearization in the sense that it tries to directly approximate the mean and covariance of the target distribution instead of trying to approximate the non-linear function (Julier and Uhlmann, 1995).

The idea of the UT is to deterministically choose a fixed number of sigma points that capture the mean and covariance of the original distribution of $\mathbf{x}$ exactly. These sigma points are then propagated through the non-linearity, and the mean and covariance of the transformed variable are estimated from them. Note that although the unscented transform resembles Monte Carlo estimation, the approaches are significantly different, because in the UT the sigma points are selected deterministically (Julier and Uhlmann, 2004). The difference between linear approximation and the UT is illustrated in Figures 5.3, 5.4, and 5.5.

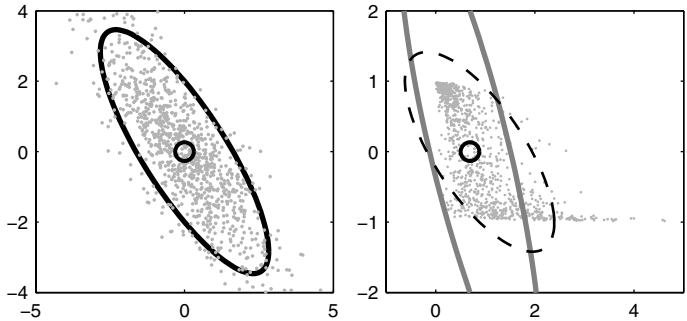
The *unscented transform* forms the Gaussian approximation[2] with the following procedure.

[2] Note that this Gaussianity assumption is one interpretation, but the unscented transform can also be applied without the Gaussian assumption. However, because the assumption makes Bayesian interpretation of the UT much easier, we shall use it here.



(a) Original (b) Transformed

Figure 5.3 Example of applying a non-linear transformation to a random variable on the left, which results in the random variable on the right.

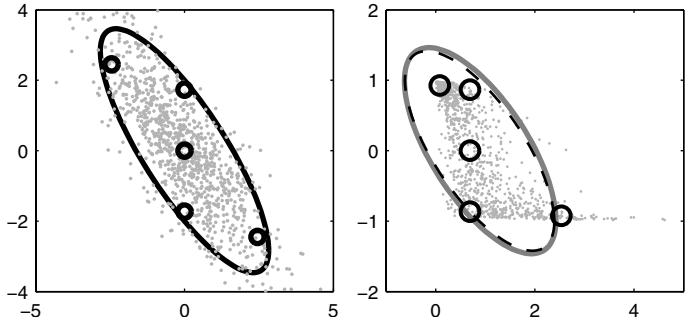


(a) Original (b) Transformed

Figure 5.4 Illustration of the linearization based (EKF) approximation to the transformation in Figure 5.3. The Gaussian approximation is formed by calculating the curvature at the mean, which results in a bad approximation further away from the mean. The covariance of the true distribution is presented by the dashed line and the solid line is the approximation.

1 Form a set of $2n + 1$ sigma points as follows:

$$\begin{aligned} \mathcal{X}^{(0)} &= \mathbf{m}, \\ \mathcal{X}^{(i)} &= \mathbf{m} + \sqrt{n+\lambda}\left[\sqrt{\mathbf{P}}\right]_i, \\ \mathcal{X}^{(i+n)} &= \mathbf{m} - \sqrt{n+\lambda}\left[\sqrt{\mathbf{P}}\right]_i, \quad i = 1, \ldots, n, \end{aligned} \tag{5.74}$$



(a) Original (b) Transformed

Figure 5.5 Illustration of the unscented transform based (UKF) approximation to the transformation in Figure 5.3. The Gaussian approximation is formed by propagating the sigma points through the non-linearity and the mean and covariance are estimated from the transformed sigma points. The covariance of the true distribution is presented by the dashed line and the solid line is the approximation.

where $[\cdot]_i$ denotes the ith column of the matrix, and λ is a scaling parameter, which is defined in terms of algorithm parameters α and κ as follows:

$$\lambda = \alpha^2 (n + \kappa) - n. \tag{5.75}$$

The parameters α and κ determine the spread of the sigma points around the mean (Wan and Van der Merwe, 2001). The matrix square root denotes a matrix such that $\sqrt{\mathbf{P}}\sqrt{\mathbf{P}}^{\mathsf{T}} = \mathbf{P}$.

2 Propagate the sigma points through the non-linear function $\mathbf{g}(\cdot)$:

$$\mathcal{Y}^{(i)} = \mathbf{g}(\mathcal{X}^{(i)}), \quad i = 0, \ldots, 2n,$$

which results in the transformed sigma points $\mathcal{Y}^{(i)}$.

3 Estimates of the mean and covariance of the transformed variable can be computed from the sigma points as follows:

$$\begin{aligned} \mathrm{E}[\mathbf{g}(\mathbf{x})] &\simeq \boldsymbol{\mu}_{\mathrm{U}} = \sum_{i=0}^{2n} W_i^{(\mathrm{m})} \mathcal{Y}^{(i)}, \\ \mathrm{Cov}[\mathbf{g}(\mathbf{x})] &\simeq \mathbf{S}_{\mathrm{U}} = \sum_{i=0}^{2n} W_i^{(\mathrm{c})} (\mathcal{Y}^{(i)} - \boldsymbol{\mu}_{\mathrm{U}}) (\mathcal{Y}^{(i)} - \boldsymbol{\mu}_{\mathrm{U}})^{\mathsf{T}}, \end{aligned} \tag{5.76}$$

where the constant weights $W_i^{(\mathrm{m})}$ and $W_i^{(\mathrm{c})}$ are given as follows (Wan and Van der Merwe, 2001):

$$\begin{aligned} W_0^{(\mathrm{m})} &= \frac{\lambda}{n+\lambda}, \\ W_0^{(\mathrm{c})} &= \frac{\lambda}{n+\lambda} + (1-\alpha^2+\beta), \\ W_i^{(\mathrm{m})} &= \frac{1}{2(n+\lambda)}, \quad i = 1,\ldots,2n, \\ W_i^{(\mathrm{c})} &= \frac{1}{2(n+\lambda)}, \quad i = 1,\ldots,2n, \end{aligned} \tag{5.77}$$

and β is an additional algorithm parameter that can be used for incorporating prior information on the (non-Gaussian) distribution of $\mathbf{x}$ (Wan and Van der Merwe, 2001).

If we apply the unscented transform to the augmented function $\tilde{\mathbf{g}}(\mathbf{x}) = (\mathbf{x}, \mathbf{g}(\mathbf{x}))$, we simply get a set of sigma points, where the sigma points $\mathcal{X}^{(i)}$ and $\mathcal{Y}^{(i)}$ have been concatenated to the same vector. Thus, also forming the approximation to the joint distribution $\mathbf{x}$ and $\mathbf{g}(\mathbf{x}) + \mathbf{q}$ is straightforward and the result is the following algorithm.

Algorithm 5.12 (Unscented approximation of an additive transform) *The unscented transform based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}) + \mathbf{q}$, *where* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$, *is given as*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{U}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{U}} \\ \mathbf{C}_{\mathrm{U}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{U}} \end{pmatrix} \right), \tag{5.78}$$

where the submatrices can be computed as follows.

1 *Form the set of* $2n+1$ *sigma points as follows:*

$$\begin{aligned} \mathcal{X}^{(0)} &= \mathbf{m}, \\ \mathcal{X}^{(i)} &= \mathbf{m} + \sqrt{n+\lambda}\left[\sqrt{\mathbf{P}}\right]_i, \\ \mathcal{X}^{(i+n)} &= \mathbf{m} - \sqrt{n+\lambda}\left[\sqrt{\mathbf{P}}\right]_i, \quad i = 1,\ldots,n, \end{aligned} \tag{5.79}$$

where the parameter λ *is defined in Equation* (5.75).

2 *Propagate the sigma points through the non-linear function* $\mathbf{g}(\cdot)$*:*

$$\mathcal{Y}^{(i)} = \mathbf{g}(\mathcal{X}^{(i)}), \quad i = 0,\ldots,2n.$$

3 The submatrices are then given as:

$$\begin{aligned}
\boldsymbol{\mu}_{\mathrm{U}} &= \sum_{i=0}^{2n} W_i^{(\mathrm{m})}\, \mathcal{Y}^{(i)}, \\
\mathbf{S}_{\mathrm{U}} &= \sum_{i=0}^{2n} W_i^{(\mathrm{c})}\, (\mathcal{Y}^{(i)} - \boldsymbol{\mu}_{\mathrm{U}})\, (\mathcal{Y}^{(i)} - \boldsymbol{\mu}_{\mathrm{U}})^{\mathsf{T}} + \mathbf{Q}, \\
\mathbf{C}_{\mathrm{U}} &= \sum_{i=0}^{2n} W_i^{(\mathrm{c})}\, (\mathcal{X}^{(i)} - \mathbf{m})\, (\mathcal{Y}^{(i)} - \boldsymbol{\mu}_{\mathrm{U}})^{\mathsf{T}},
\end{aligned} \tag{5.80}$$

where the constant weights $W_i^{(\mathrm{m})}$ and $W_i^{(\mathrm{c})}$ were defined in Equation (5.77)*.*

The unscented transform approximation to a transformation of the form $\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{q})$ can be derived by considering the augmented random variable $\tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{q})$ as the random variable in the transform. The resulting algorithm is the following.

Algorithm 5.13 (Unscented approximation of a non-additive transform) *The (augmented) unscented transform based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{q})$ *when* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$ *is given as*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{U}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{U}} \\ \mathbf{C}_{\mathrm{U}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{U}} \end{pmatrix} \right), \tag{5.81}$$

where the sub-matrices can be computed as follows. Let the dimensionalities of $\mathbf{x}$ *and* $\mathbf{q}$ *be* n *and* n_q*, respectively, and let* $n' = n + n_q$*.*

1 Form the sigma points for the augmented random variable $\tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{q})$*:*

$$\begin{aligned}
\tilde{\mathcal{X}}^{(0)} &= \tilde{\mathbf{m}}, \\
\tilde{\mathcal{X}}^{(i)} &= \tilde{\mathbf{m}} + \sqrt{n' + \lambda'}\, \left[\sqrt{\tilde{\mathbf{P}}} \right]_i, \\
\tilde{\mathcal{X}}^{(i+n')} &= \tilde{\mathbf{m}} - \sqrt{n' + \lambda'}\, \left[\sqrt{\tilde{\mathbf{P}}} \right]_i, \quad i = 1, \ldots, n',
\end{aligned} \tag{5.82}$$

where the parameter λ' *is defined as in Equation* (5.75)*, but with* n *replaced by* n'*, and the augmented mean and covariance are defined by*

$$\tilde{\mathbf{m}} = \begin{pmatrix} \mathbf{m} \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\mathbf{P}} = \begin{pmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{pmatrix}.$$

2 *Propagate the sigma points through the function:*

$$\tilde{\mathcal{Y}}^{(i)} = \mathbf{g}(\tilde{\mathcal{X}}^{(i),x}, \tilde{\mathcal{X}}^{(i),q}), \quad i = 0, \ldots, 2n',$$

where $\tilde{\mathcal{X}}^{(i),x}$ *and* $\tilde{\mathcal{X}}^{(i),q}$ *denote the parts of the augmented sigma point* i *which correspond to* $\mathbf{x}$ *and* $\mathbf{q}$*, respectively.*

3 *Compute the predicted mean* $\boldsymbol{\mu}_{\mathrm{U}}$*, the predicted covariance* $\mathbf{S}_{\mathrm{U}}$*, and the cross-covariance* $\mathbf{C}_{\mathrm{U}}$*:*

$$\boldsymbol{\mu}_{\mathrm{U}} = \sum_{i=0}^{2n'} W_i^{(\mathrm{m})'} \tilde{\mathcal{Y}}^{(i)},$$

$$\mathbf{S}_{\mathrm{U}} = \sum_{i=0}^{2n'} W_i^{(\mathrm{c})'} (\tilde{\mathcal{Y}}^{(i)} - \boldsymbol{\mu}_{\mathrm{U}}) (\tilde{\mathcal{Y}}^{(i)} - \boldsymbol{\mu}_{\mathrm{U}})^{\mathsf{T}},$$

$$\mathbf{C}_{\mathrm{U}} = \sum_{i=0}^{2n'} W_i^{(\mathrm{c})'} (\tilde{\mathcal{X}}^{(i),x} - \mathbf{m}) (\tilde{\mathcal{Y}}^{(i)} - \boldsymbol{\mu}_{\mathrm{U}})^{\mathsf{T}},$$

where the definitions of the weights $W_i^{(\mathrm{m})'}$ *and* $W_i^{(\mathrm{c})'}$ *are the same as in Equation* (5.77)*, but with* n *replaced by* n' *and* λ *replaced by* λ'*.*

The unscented transform is a third order method in the sense that the estimate of the mean of $\mathbf{g}(\cdot)$ is exact for polynomials up to order three. That is, if $\mathbf{g}(\cdot)$ is indeed a multi-variate polynomial of order three, the mean is exact. However, the covariance approximation is exact only for first order polynomials, because the square of a second order polynomial is already a polynomial of order four, and the unscented transform (UT) does not compute the exact result for fourth order polynomials. In this sense the UT is only a first order method. With suitable selection of parameters ($\kappa = 3-n$) it is possible to get some of the fourth order terms appearing in the covariance computation right also for quadratic functions, but not all.

5.6 Unscented Kalman filter

The *unscented Kalman filter* (UKF) (Julier et al., 1995; Julier and Uhlmann, 2004; Wan and Van der Merwe, 2001) is an approximate filtering algorithm that utilizes the unscented transform and can be used for approximating the filtering distributions of models having the same form as with the EKF and the SLF, that is, models of the form (5.24) or (5.37). As the EKF and the SLF, the UKF forms a Gaussian approximation to the filtering distribution:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \simeq \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k), \tag{5.83}$$

where $\mathbf{m}_k$ and $\mathbf{P}_k$ are the mean and covariance computed by the algorithm.

Algorithm 5.14 (Unscented Kalman filter I) *In the additive form of the unscented Kalman filter (UKF) algorithm, which can be applied to additive models of the form* (5.24)*, the following operations are performed at each measurement step* $k = 1, 2, 3, \ldots$

- *Prediction:*

  1 *Form the sigma points:*

$$\begin{aligned}
\mathcal{X}_{k-1}^{(0)} &= \mathbf{m}_{k-1}, \\
\mathcal{X}_{k-1}^{(i)} &= \mathbf{m}_{k-1} + \sqrt{n+\lambda}\left[\sqrt{\mathbf{P}_{k-1}}\right]_i, \\
\mathcal{X}_{k-1}^{(i+n)} &= \mathbf{m}_{k-1} - \sqrt{n+\lambda}\left[\sqrt{\mathbf{P}_{k-1}}\right]_i, \quad i = 1, \ldots, n,
\end{aligned} \tag{5.84}$$

  where the parameter λ *is defined in Equation* (5.75)*.*

  2 *Propagate the sigma points through the dynamic model:*

$$\hat{\mathcal{X}}_k^{(i)} = \mathbf{f}(\mathcal{X}_{k-1}^{(i)}), \quad i = 0, \ldots, 2n. \tag{5.85}$$

  3 *Compute the predicted mean* $\mathbf{m}_k^-$ *and the predicted covariance* $\mathbf{P}_k^-$*:*

$$\begin{aligned}
\mathbf{m}_k^- &= \sum_{i=0}^{2n} W_i^{(\mathrm{m})}\, \hat{\mathcal{X}}_k^{(i)}, \\
\mathbf{P}_k^- &= \sum_{i=0}^{2n} W_i^{(\mathrm{c})}\, (\hat{\mathcal{X}}_k^{(i)} - \mathbf{m}_k^-)\, (\hat{\mathcal{X}}_k^{(i)} - \mathbf{m}_k^-)^{\mathsf{T}} + \mathbf{Q}_{k-1},
\end{aligned} \tag{5.86}$$

  where the weights $W_i^{(\mathrm{m})}$ *and* $W_i^{(\mathrm{c})}$ *were defined in Equation* (5.77)*.*

- *Update:*

  1 *Form the sigma points:*

$$\begin{aligned}
\mathcal{X}_k^{-(0)} &= \mathbf{m}_k^-, \\
\mathcal{X}_k^{-(i)} &= \mathbf{m}_k^- + \sqrt{n+\lambda}\left[\sqrt{\mathbf{P}_k^-}\right]_i, \\
\mathcal{X}_k^{-(i+n)} &= \mathbf{m}_k^- - \sqrt{n+\lambda}\left[\sqrt{\mathbf{P}_k^-}\right]_i, \quad i = 1, \ldots, n.
\end{aligned} \tag{5.87}$$

  2 *Propagate sigma points through the measurement model:*

$$\hat{\mathcal{Y}}_k^{(i)} = \mathbf{h}(\mathcal{X}_k^{-(i)}), \quad i = 0, \ldots, 2n. \tag{5.88}$$

3 *Compute the predicted mean* $\boldsymbol{\mu}_k$*, the predicted covariance of the measurement* $\mathbf{S}_k$*, and the cross-covariance of the state and the measurement* $\mathbf{C}_k$*:*

$$\begin{aligned}
\boldsymbol{\mu}_k &= \sum_{i=0}^{2n} W_i^{(\mathrm{m})}\, \hat{\mathcal{Y}}_k^{(i)}, \\
\mathbf{S}_k &= \sum_{i=0}^{2n} W_i^{(\mathrm{c})}\, (\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)\, (\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)^{\mathsf{T}} + \mathbf{R}_k, \\
\mathbf{C}_k &= \sum_{i=0}^{2n} W_i^{(\mathrm{c})}\, (\mathcal{X}_k^{-(i)} - \mathbf{m}_k^-)\, (\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)^{\mathsf{T}}.
\end{aligned} \tag{5.89}$$

4 *Compute the filter gain* $\mathbf{K}_k$*, the filtered state mean* $\mathbf{m}_k$ *and the covariance* $\mathbf{P}_k$*, conditional on the measurement* $\mathbf{y}_k$*:*

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{C}_k\, \mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\, [\mathbf{y}_k - \boldsymbol{\mu}_k], \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\, \mathbf{S}_k\, \mathbf{K}_k^{\mathsf{T}}.
\end{aligned} \tag{5.90}$$

The filtering equations above can be derived in an analogous manner to the EKF equations, but the unscented transform based approximations are used instead of the linear approximations.

The non-additive form of the UKF (Julier and Uhlmann, 2004) can be derived by augmenting the process and measurement noises to the state vector and then using the UT approximation for performing prediction and update steps simultaneously. Alternatively, we can first augment the state vector with process noise, then approximate the prediction step and after that do the same with measurement noise on the update step. The different algorithms and ways of doing this in practice are analyzed in the article by Wu et al. (2005). However, if we directly apply the non-additive UT in Algorithm 5.13 separately to prediction and update steps, we get the following algorithm.

Algorithm 5.15 (Unscented Kalman filter II) *In the augmented form of the unscented Kalman filter (UKF) algorithm, which can be applied to non-additive models of the form* (5.37)*, the following operations are performed at each measurement step* $k = 1, 2, 3, \ldots$

- *Prediction:*

1 *Form the sigma points for the augmented random variable* $(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})$*:*

$$\begin{aligned}
\tilde{\mathcal{X}}_{k-1}^{(0)} &= \tilde{\mathbf{m}}_{k-1}, \\
\tilde{\mathcal{X}}_{k-1}^{(i)} &= \tilde{\mathbf{m}}_{k-1} + \sqrt{n' + \lambda'} \left[\sqrt{\tilde{\mathbf{P}}_{k-1}} \right]_i, \\
\tilde{\mathcal{X}}_{k-1}^{(i+n')} &= \tilde{\mathbf{m}}_{k-1} - \sqrt{n' + \lambda'} \left[\sqrt{\tilde{\mathbf{P}}_{k-1}} \right]_i, \quad i = 1, \ldots, n', \quad (5.91)
\end{aligned}$$

where

$$\tilde{\mathbf{m}}_{k-1} = \begin{pmatrix} \mathbf{m}_{k-1} \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\mathbf{P}}_{k-1} = \begin{pmatrix} \mathbf{P}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{k-1} \end{pmatrix}.$$

Here $n' = n + n_q$*, where* n *is the dimensionality of the state* $\mathbf{x}_{k-1}$ *and* n_q *is the dimensionality of the noise* $\mathbf{q}_{k-1}$*. The parameter* λ' *is defined as in Equation* (5.75)*, but with* n *replaced by* n'*.*

2 *Propagate the sigma points through the dynamic model:*

$$\hat{\mathcal{X}}_k^{(i)} = \mathbf{f}(\tilde{\mathcal{X}}_{k-1}^{(i),x}, \tilde{\mathcal{X}}_{k-1}^{(i),q}), \quad i = 0, \ldots, 2n', \qquad (5.92)$$

where $\tilde{\mathcal{X}}_{k-1}^{(i),x}$ *denotes the first* n *components in* $\tilde{\mathcal{X}}_{k-1}^{(i)}$ *and* $\tilde{\mathcal{X}}_{k-1}^{(i),q}$ *denotes the last* n_q *components.*

3 *Compute the predicted mean* $\mathbf{m}_k^-$ *and the predicted covariance* $\mathbf{P}_k^-$*:*

$$\begin{aligned}
\mathbf{m}_k^- &= \sum_{i=0}^{2n} W_i^{(\mathrm{m})'} \hat{\mathcal{X}}_k^{(i)}, \\
\mathbf{P}_k^- &= \sum_{i=0}^{2n} W_i^{(\mathrm{c})'} (\hat{\mathcal{X}}_k^{(i)} - \mathbf{m}_k^-)(\hat{\mathcal{X}}_k^{(i)} - \mathbf{m}_k^-)^{\mathsf{T}}, \qquad (5.93)
\end{aligned}$$

where the weights $W_i^{(\mathrm{m})'}$ *and* $W_i^{(\mathrm{c})'}$ *are the same as in Equation* (5.77)*, but with* n *replaced by* n' *and* λ *by* λ'*.*

- *Update:*

1 *Form the sigma points for the augmented random variable* $(\mathbf{x}_k, \mathbf{r}_k)$*:*

$$\begin{aligned}
\tilde{\mathcal{X}}_k^{-(0)} &= \tilde{\mathbf{m}}_k^-, \\
\tilde{\mathcal{X}}_k^{-(i)} &= \tilde{\mathbf{m}}_k^- + \sqrt{n'' + \lambda''} \left[\sqrt{\tilde{\mathbf{P}}_k^-} \right]_i, \\
\tilde{\mathcal{X}}_k^{-(i+n'')} &= \mathbf{m}_k^- - \sqrt{n'' + \lambda''} \left[\sqrt{\tilde{\mathbf{P}}_k^-} \right]_i, \quad i = 1, \ldots, n'', \quad (5.94)
\end{aligned}$$

where

$$\tilde{\mathbf{m}}_k^- = \begin{pmatrix} \mathbf{m}_k^- \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\mathbf{P}}_k^- = \begin{pmatrix} \mathbf{P}_k^- & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k \end{pmatrix}.$$

Here we have defined $n'' = n + n_r$, where n is the dimensionality of the state $\mathbf{x}_k$ and n_r is the dimensionality of the noise $\mathbf{r}_k$. The parameter λ'' is defined as in Equation (5.75)*, but with n replaced by n''.*

2 *Propagate sigma points through the measurement model:*

$$\hat{\mathcal{Y}}_k^{(i)} = \mathbf{h}(\tilde{\mathcal{X}}_k^{-(i),x}, \tilde{\mathcal{X}}_k^{-(i),r}), \quad i = 0, \ldots, 2n'', \tag{5.95}$$

where $\tilde{\mathcal{X}}_k^{-(i),x}$ denotes the first n components in $\tilde{\mathcal{X}}_k^{-(i)}$ and $\tilde{\mathcal{X}}_k^{-(i),r}$ denotes the last n_r components.

3 *Compute the predicted mean $\boldsymbol{\mu}_k$, the predicted covariance of the measurement $\mathbf{S}_k$, and the cross-covariance of the state and the measurement $\mathbf{C}_k$:*

$$\begin{aligned}
\boldsymbol{\mu}_k &= \sum_{i=0}^{2n''} W_i^{(\mathrm{m})''} \hat{\mathcal{Y}}_k^{(i)}, \\
\mathbf{S}_k &= \sum_{i=0}^{2n''} W_{i-1}^{(\mathrm{c})''} (\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)(\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)^\mathsf{T}, \\
\mathbf{C}_k &= \sum_{i=0}^{2n''} W_i^{(\mathrm{c})''} (\mathcal{X}_k^{-(i),x} - \mathbf{m}_k^-)(\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)^\mathsf{T},
\end{aligned} \tag{5.96}$$

where the weights $W_i^{(\mathrm{m})''}$ and $W_i^{(\mathrm{c})''}$ are the same as in Equation (5.77)*, but with n replaced by n'' and λ by λ''.*

4 *Compute the filter gain $\mathbf{K}_k$ and the filtered state mean $\mathbf{m}_k$ and covariance $\mathbf{P}_k$, conditional to the measurement $\mathbf{y}_k$:*

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{C}_k \mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k [\mathbf{y}_k - \boldsymbol{\mu}_k], \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\mathsf{T}.
\end{aligned} \tag{5.97}$$

The advantage of the UKF over the EKF is that the UKF is not based on a linear approximation at a single point, but uses further points in approximating the non-linearity. As discussed in Julier and Uhlmann (2004), the unscented transform is able to capture the higher order moments caused by the non-linear transform better than Taylor series based approximations. However, as already pointed out in the previous section, although the mean

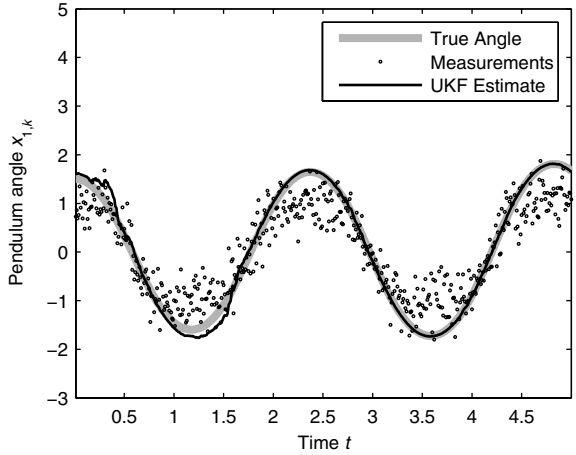


Figure 5.6 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with the UKF. The resulting RMSE was 0.11 which is the same as with the SLF (Figure 5.2).

estimate of the UKF is exact for polynomials up to order 3, the covariance computation is only exact for polynomials up to the first order (as, e.g., in the SLF). In the UKF, the dynamic and model functions are also not required to be formally differentiable nor do their Jacobian matrices need to be computed. The advantage of the UKF over the SLF is that in the UKF there is no need to compute any expected values in closed form, only evaluations of the dynamic and measurement models are needed. However, the accuracy of the UKF cannot be expected to be as good as that of the SLF, because the SLF uses a larger area in the approximation, whereas the UKF only selects a fixed number of points in the area. The disadvantage over the EKF is that the UKF often requires slightly more computational operations than the EKF.

Example 5.3 (Pendulum tracking with UKF) *The result of applying the UKF to the pendulum model and simulated data in Example 5.1 is shown in Figure 5.6. Unlike in the EKF and the SLF, we do not need to derive analytical derivatives or expectations for implementing the UKF. Still the RMSE is the same as with the SLF.*

The UKF can be interpreted as belonging to a wider class of filters called sigma-point filters (Van der Merwe and Wan, 2003), which also includes other types of filter such as the central differences Kalman filter (CDKF), the Gauss–Hermite Kalman filter (GHKF) and a few others (Ito and Xiong, 2000; Wu et al., 2006; Nørgaard et al., 2000; Arasaratnam and Haykin, 2009). The classification of sigma-point methods by Van der Merwe and Wan (2003) is based on interpreting the methods as special cases of statistical linear regression (Lefebvre et al., 2002). Statistical linearization is closely related to sigma-point approximations, because they both are related to statistical linear regression (Van der Merwe and Wan, 2003). However, it is important to note that the statistical linear regression (Lefebvre et al., 2002) which is the basis of the sigma-point framework (Van der Merwe and Wan, 2003) is not exactly equivalent to statistical linearization (Gelb, 1974). Statistical linear regression can be considered as a discrete approximation to statistical linearization.

Although at first glance the UKF seems to be quite different from the EKF, they are indeed very closely related. In fact, Gustafsson and Hendeby (2012) have shown that the mean of the UKF converges to the second order EKF in a certain limit of parameter values. In the limit the UT approximation becomes equivalent to using central difference approximations to the first and second derivatives, hence the result.

Sandblom and Svensson (2012) have also recently proposed a marginalized transform which forms (hierarchical Bayesian) Hermite series expansion based approximations to transformations of Gaussian random variables by fitting the series coefficients via point-wise evaluations. The resulting algorithm contains the UKF (or the UT) as a special case and hence it also reveals the close connection between the Fourier–Hermite series based filters (Sarmavuori and Särkkä, 2012a) and the UKF.

5.7 Exercises

5.1 Consider the following non-linear state space model:

$$\begin{aligned} x_k &= x_{k-1} - 0.01\,\sin(x_{k-1}) + q_{k-1}, \\ y_k &= 0.5\,\sin(2\,x_k) + r_k, \end{aligned} \tag{5.98}$$

where q_{k-1} has a variance of 0.01^2 and r_k has a variance of 0.02. Derive the required derivatives for an EKF and implement the EKF for the model. Simulate trajectories from the model, compute the RMSE values, and plot the result.

5.2 For the above model, derive the required expected values for an SLF and implement the SLF for the model. *Hint:* use the imaginary part of the inverse Fourier transform of the Gaussian distribution. Compute the RMSE values, plot the results, and compare the performance to the EKF above.

5.3 In this exercise your task is to derive the derivative form of the statistically linearized filter (SLF).

(a) Prove using integration by parts the following identity for a Gaussian random variable $\mathbf{x}$, differentiable non-linear function $\mathbf{g}(\mathbf{x})$, and its Jacobian matrix $\mathbf{G}_x(\mathbf{x}) = \partial \mathbf{g}(\mathbf{x})/\partial \mathbf{x}$:

$$\mathrm{E}[\mathbf{g}(\mathbf{x})\,(\mathbf{x} - \mathbf{m})^\mathsf{T}] = \mathrm{E}[\mathbf{G}_x(\mathbf{x})]\,\mathbf{P}, \tag{5.99}$$

where $\mathrm{E}[\cdot]$ denotes the expected value with respect to $\mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})$. *Hint*: $\frac{\partial}{\partial \mathbf{x}} \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P}) = -\mathbf{P}^{-1}(\mathbf{x} - \mathbf{m})\,\mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})$.

(b) Prove the following. Let

$$\boldsymbol{\mu}(\mathbf{m}) = \mathrm{E}[\mathbf{g}(\mathbf{x})], \tag{5.100}$$

where $\mathrm{E}[\cdot]$ denotes the expected value with respect to $\mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})$. Then

$$\frac{\partial \boldsymbol{\mu}(\mathbf{m})}{\partial \mathbf{m}} = \mathrm{E}[\mathbf{G}_x(\mathbf{x})]. \tag{5.101}$$

(c) Write down the additive form of the SLF equations in an alternative form, where you have eliminated all the cross terms of the form $\mathrm{E}[\mathbf{f}(\mathbf{x}_{k-1})\,\delta\mathbf{x}_{k-1}^\mathsf{T}]$ and $\mathrm{E}[\mathbf{h}(\mathbf{x}_k)\,\delta\mathbf{x}_k^\mathsf{T}]^\mathsf{T}$, using the result in (a).

(d) How can you utilize the result (b) when using the alternative form of the SLF? Check that you get the same equations for the SLF in the previous exercise using this alternative form of the SLF.

5.4 Implement a UKF for the model in Exercise 5.1. Plot the results and compare the RMSE values to the EKF and the SLF.

5.5 In this exercise we consider a classical bearings only target tracking problem which frequently arises in the context of passive sensor tracking. In this problem there is single target in the scene and two angular sensors are used for tracking it. The scenario is illustrated in Figure 5.7.

The state of the target at time step k consist of the position (x_k, y_k) and the velocity $(\dot{x}_k, \dot{y}_k)$. The dynamics of the state vector $\mathbf{x}_k = (x_k\ y_k\ \dot{x}_k\ \dot{y}_k)^\mathsf{T}$ are modeled with the discretized Wiener velocity model:

$$\begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{pmatrix} + \mathbf{q}_{k-1},$$

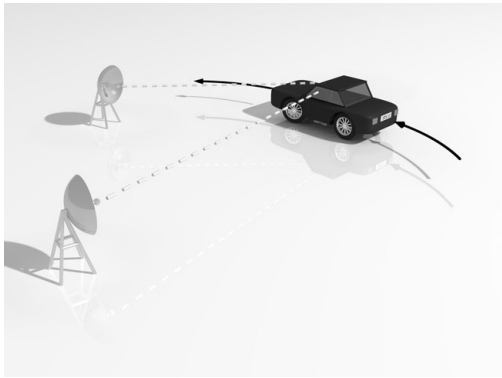


Figure 5.7 In the bearings only target tracking problem the sensors generate angle measurements of the target, and the purpose is to determine the target trajectory.

where $\mathbf{q}_k$ is a zero mean Gaussian process noise with covariance

$$\mathbf{Q} = \begin{pmatrix} q_1^{\mathrm{c}}\,\Delta t^3/3 & 0 & q_1^{\mathrm{c}}\,\Delta t^2/2 & 0 \\ 0 & q_2^{\mathrm{c}}\,\Delta t^3/3 & 0 & q_2^{\mathrm{c}}\,\Delta t^2/2 \\ q_1^{\mathrm{c}}\,\Delta t^2/2 & 0 & q_1^{\mathrm{c}}\,\Delta t & 0 \\ 0 & q_2^{\mathrm{c}}\,\Delta t^2/2 & 0 & q_2^{\mathrm{c}}\,\Delta t \end{pmatrix}.$$

In this scenario the diffusion coefficients are $q_1^{\mathrm{c}} = q_2^{\mathrm{c}} = 0.1$ and the sampling period is $\Delta t = 0.1$. The measurement model for sensor $i \in \{1, 2\}$ is the following:

$$\theta_k^i = \tan^{-1}\left(\frac{y_k - s_y^i}{x_k - s_x^i}\right) + r_k, \tag{5.102}$$

where (s_x^i, s_y^i) is the position of the sensor i and $r_k \sim \mathrm{N}(0, \sigma^2)$ is a Gaussian measurement noise with a standard deviation of $\sigma = 0.05$ radians. At each sampling time, which occurs ten times per second (i.e., $\Delta t = 0.1$), both of the two sensors produce a measurement.

In the file `angle_ex.m` there is a baseline solution, which computes estimates of the position from the crossing of the measurements and estimates the velocity to be always zero. Your task is to implement an EKF for the problem and compare the results graphically and in the RMSE sense.

(a) Implement an EKF for the bearings only target tracking problem, which uses the non-linear measurement model (5.102) as its measurement model function (not the crossings). *Hints:*

- Use the MATLAB® function `atan2` in the measurement model instead of `atan` to directly get an answer in the range $[-\pi, \pi]$.

- The two measurements at each measurement time can be processed one at a time, that is, you can simply perform two scalar updates instead of a single two-dimensional measurement update.
- Start by computing the Jacobian matrix of the measurement model function with respect to the state components. Before implementing the filter, check by finite differences that the Jacobian matrix is correct.

(b) Compute the RMSE values and plot figures of the estimates.

5.6 Implement a UKF for the bearings only target tracking problem in Exercise 5.5. Compare the performance to the EKF.

6

General Gaussian filtering

Quite soon after the unscented Kalman filter (UKF) was published, Ito and Xiong (2000) pointed out that the UKF can be considered as a special case of so-called Gaussian filters, where the non-linear filtering problem is solved using Gaussian assumed density approximations. The generalized framework also enables the usage of various powerful Gaussian quadrature and cubature integration methods (Wu et al., 2006; Arasaratnam and Haykin, 2009). The series expansion based filters presented in the previous sections can be seen as approximations to the general Gaussian filter. In this section we present the Gaussian filtering framework and show how the Gauss–Hermite Kalman filter (GHKF) and the cubature Kalman filter (CKF) can be derived as its approximations. We also show how the UKF can be seen as a generalization of the CKF.

6.1 Gaussian moment matching

One way to unify various Gaussian approximations to non-linear transforms is to reduce them to approximate computation of Gaussian integrals of the form

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x}.$$

If we can compute these, a straightforward way to form the Gaussian approximation for $(\mathbf{x}, \mathbf{y})$ is to simply match the moments of the distributions, which gives the following algorithm.

Algorithm 6.1 (Gaussian moment matching of an additive transform) *The moment matching based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}) + \mathbf{q}$*, where* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$*, is given by*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{M}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{M}} \\ \mathbf{C}_{\mathrm{M}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{M}} \end{pmatrix} \right), \tag{6.1}$$

where

$$\begin{aligned}
\boldsymbol{\mu}_{\mathrm{M}} &= \int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x}, \\
\mathbf{S}_{\mathrm{M}} &= \int (\mathbf{g}(\mathbf{x}) - \boldsymbol{\mu}_{\mathrm{M}})\,(\mathbf{g}(\mathbf{x}) - \boldsymbol{\mu}_{\mathrm{M}})^{\mathsf{T}}\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} + \mathbf{Q}, \\
\mathbf{C}_{\mathrm{M}} &= \int (\mathbf{x} - \mathbf{m})\,(\mathbf{g}(\mathbf{x}) - \boldsymbol{\mu}_{\mathrm{M}})^{\mathsf{T}}\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x}.
\end{aligned} \tag{6.2}$$

It is now easy to check by substituting the linear approximation $\mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{m}) + \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,(\mathbf{x} - \mathbf{m})$ into the above expression that the integrals reduce to the linear approximations in Algorithm 5.1. The analogous thing happens with quadratic approximations and statistical linearization. The unscented transform can also be seen as a special case of the above algorithm, as we will see in the following sections.

The non-additive version of the transform is the following.

Algorithm 6.2 (Gaussian moment matching of a non-additive transform) *The moment matching based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{q})$*, where* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$*, is given by*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{M}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{M}} \\ \mathbf{C}_{\mathrm{M}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{M}} \end{pmatrix} \right), \tag{6.3}$$

where

$$\begin{aligned}
\boldsymbol{\mu}_{\mathrm{M}} &= \int \mathbf{g}(\mathbf{x}, \mathbf{q})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{N}(\mathbf{q} \mid \mathbf{0}, \mathbf{Q})\ \mathrm{d}\mathbf{x}\ \mathrm{d}\mathbf{q}, \\
\mathbf{S}_{\mathrm{M}} &= \int (\mathbf{g}(\mathbf{x}, \mathbf{q}) - \boldsymbol{\mu}_{\mathrm{M}})\,(\mathbf{g}(\mathbf{x}, \mathbf{q}) - \boldsymbol{\mu}_{\mathrm{M}})^{\mathsf{T}} \\
&\qquad \times \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{N}(\mathbf{q} \mid \mathbf{0}, \mathbf{Q})\ \mathrm{d}\mathbf{x}\ \mathrm{d}\mathbf{q}, \\
\mathbf{C}_{\mathrm{M}} &= \int (\mathbf{x} - \mathbf{m})\,(\mathbf{g}(\mathbf{x}, \mathbf{q}) - \boldsymbol{\mu}_{\mathrm{M}})^{\mathsf{T}}\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{N}(\mathbf{q} \mid \mathbf{0}, \mathbf{Q})\ \mathrm{d}\mathbf{x}\ \mathrm{d}\mathbf{q}.
\end{aligned} \tag{6.4}$$

6.2 Gaussian filter

If we use the moment matching approximations for constructing a filtering algorithm, we get the following *Gaussian assumed density filter* (ADF) which is also called the *Gaussian filter* (Maybeck, 1982b; Ito and Xiong,

2000; Wu et al., 2006). The key idea is to *assume* that the filtering distribution is indeed Gaussian,

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \simeq \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k), \tag{6.5}$$

and approximate its mean $\mathbf{m}_k$ and covariance $\mathbf{P}_k$ via moment matching.

Algorithm 6.3 (Gaussian filter I) *The prediction and update steps of the additive noise Gaussian (Kalman) filter are:*

- *Prediction:*

$$\begin{aligned}
\mathbf{m}_k^- &= \int \mathbf{f}(\mathbf{x}_{k-1})\ \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1})\ \mathrm{d}\mathbf{x}_{k-1}, \\
\mathbf{P}_k^- &= \int (\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{m}_k^-)\,(\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{m}_k^-)^\mathsf{T} \\
&\qquad \times \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1})\ \mathrm{d}\mathbf{x}_{k-1} + \mathbf{Q}_{k-1}.
\end{aligned} \tag{6.6}$$

- *Update:*

$$\begin{aligned}
\boldsymbol{\mu}_k &= \int \mathbf{h}(\mathbf{x}_k)\ \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\ \mathrm{d}\mathbf{x}_k, \\
\mathbf{S}_k &= \int (\mathbf{h}(\mathbf{x}_k) - \boldsymbol{\mu}_k)\,(\mathbf{h}(\mathbf{x}_k) - \boldsymbol{\mu}_k)^\mathsf{T}\ \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\ \mathrm{d}\mathbf{x}_k + \mathbf{R}_k, \\
\mathbf{C}_k &= \int (\mathbf{x}_k - \mathbf{m}_k^-)\,(\mathbf{h}(\mathbf{x}_k) - \boldsymbol{\mu}_k)^\mathsf{T}\ \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\ \mathrm{d}\mathbf{x}_k, \\
\mathbf{K}_k &= \mathbf{C}_k\,\mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\,(\mathbf{y}_k - \boldsymbol{\mu}_k), \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\,\mathbf{S}_k\,\mathbf{K}_k^\mathsf{T}.
\end{aligned} \tag{6.7}$$

The advantage of the moment matching formulation is that it enables the use of many well-known numerical integration methods such as Gauss–Hermite quadratures and cubature rules (Ito and Xiong, 2000; Wu et al., 2006; Arasaratnam and Haykin, 2009). It is also possible to use other methods such as the Bayes–Hermite quadrature (O'Hagan, 1991) or Monte Carlo integration for approximating the integrals.

The Gaussian filter can be extended to non-additive noise models as follows.

Algorithm 6.4 (Gaussian filter II) *The prediction and update steps of the non-additive noise Gaussian (Kalman) filter are:*

- *Prediction:*

$$\begin{aligned}
\mathbf{m}_k^- &= \int \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) \\
&\qquad \times \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1})\ \mathrm{N}(\mathbf{q}_{k-1} \mid \mathbf{0}, \mathbf{Q}_{k-1})\ \mathrm{d}\mathbf{x}_{k-1}\ \mathrm{d}\mathbf{q}_{k-1}, \\
\mathbf{P}_k^- &= \int (\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) - \mathbf{m}_k^-)\,(\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) - \mathbf{m}_k^-)^\mathsf{T} \\
&\qquad \times \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1})\ \mathrm{N}(\mathbf{q}_{k-1} \mid \mathbf{0}, \mathbf{Q}_{k-1})\ \mathrm{d}\mathbf{x}_{k-1}\ \mathrm{d}\mathbf{q}_{k-1}.
\end{aligned} \tag{6.8}$$

- *Update:*

$$\begin{aligned}
\boldsymbol{\mu}_k &= \int \mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) \\
&\qquad \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\ \mathrm{N}(\mathbf{r}_k \mid \mathbf{0}, \mathbf{R}_k)\ \mathrm{d}\mathbf{x}_k\ \mathrm{d}\mathbf{r}_k, \\
\mathbf{S}_k &= \int (\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) - \boldsymbol{\mu}_k)\,(\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) - \boldsymbol{\mu}_k)^\mathsf{T} \\
&\qquad \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\ \mathrm{N}(\mathbf{r}_k \mid \mathbf{0}, \mathbf{R}_k)\ \mathrm{d}\mathbf{x}_k\ \mathrm{d}\mathbf{r}_k, \\
\mathbf{C}_k &= \int (\mathbf{x}_k - \mathbf{m}_k^-)\,(\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) - \boldsymbol{\mu}_k)^\mathsf{T} \\
&\qquad \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\ \mathrm{N}(\mathbf{r}_k \mid \mathbf{0}, \mathbf{R}_k)\ \mathrm{d}\mathbf{x}_k\ \mathrm{d}\mathbf{r}_k, \\
\mathbf{K}_k &= \mathbf{C}_k\,\mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\,(\mathbf{y}_k - \boldsymbol{\mu}_k), \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\,\mathbf{S}_k\,\mathbf{K}_k^\mathsf{T}.
\end{aligned} \tag{6.9}$$

The above general Gaussian filters are theoretical constructions rather than practical filtering algorithms. However, there exist many models for which the required integrals can indeed be computed in closed form. But a more practical approach is to compute them numerically. This kind of method will be discussed in the next sections.

6.3 Gauss–Hermite integration

In the Gaussian filter (and later in the smoother) we are interested in approximating Gaussian integrals of the form

$$\begin{aligned}
&\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} \\
&\quad = \frac{1}{(2\,\pi)^{n/2}\,|\mathbf{P}|^{1/2}} \int \mathbf{g}(\mathbf{x})\ \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m})^\mathsf{T}\,\mathbf{P}^{-1}\,(\mathbf{x} - \mathbf{m})\right) \mathrm{d}\mathbf{x},
\end{aligned} \tag{6.10}$$

where $\mathbf{g}(\mathbf{x})$ is an arbitrary function. In this section, we shall derive a Gauss–Hermite based numerical cubature[1] algorithm for computing such integrals. The algorithm is based on direct generalization of the one-dimensional Gauss–Hermite rule into multiple dimensions by taking the Cartesian product of one-dimensional quadratures. The disadvantage of the method is that the required number of evaluation points is exponential with respect to the number of dimensions.

In its basic form, one-dimensional Gauss–Hermite quadrature integration refers to the special case of Gaussian quadratures with unit Gaussian weight function $w(x) = \mathrm{N}(x \mid 0, 1)$, that is, to approximations of the form

$$\int_{-\infty}^{\infty} g(x)\ \mathrm{N}(x \mid 0, 1)\ \mathrm{d}x \approx \sum_i W_i\ g(x^{(i)}), \tag{6.11}$$

where $W_i, i = 1, \ldots, p$ are the weights and $x^{(i)}$ are the evaluation points or abscissas – also sometimes called sigma points. Note that the quadrature is sometimes defined in terms of the weight function $\exp(-x^2)$, but here we shall use the "probabilists' definition" above. The two versions of the quadrature are related by simple scaling of variables.

Obviously, there are an infinite number of possible ways to select the weights and evaluation points. In Gauss–Hermite integration, as in all Gaussian quadratures, the weights and sigma points are chosen such that with a polynomial integrand the approximation becomes exact. It turns out that the polynomial order with a given number of points is maximized if we choose the sigma points to be roots of Hermite polynomials. When using the pth order Hermite polynomial $H_p(x)$, the rule will be exact for polynomials up to order $2p - 1$. The required weights can be computed in closed form (see below).

The Hermite polynomial of order p is defined as (these are the so-called "probabilists' Hermite polynomials"):

$$H_p(x) = (-1)^p\ \exp(x^2/2)\ \frac{\mathrm{d}^p}{\mathrm{d}x^p} \exp(-x^2/2). \tag{6.12}$$

[1] As one-dimensional integrals are *quadratures*, multi-dimensional integrals have been traditionally called *cubatures*.

The first few Hermite polynomials are:

$$\begin{aligned} H_0(x) &= 1, \\ H_1(x) &= x, \\ H_2(x) &= x^2 - 1, \\ H_3(x) &= x^3 - 3x, \\ H_4(x) &= x^4 - 6\,x^2 + 3, \end{aligned} \tag{6.13}$$

and further polynomials can be found from the recursion

$$H_{p+1}(x) = x\,H_p(x) - p\,H_{p-1}(x). \tag{6.14}$$

Using the same weights and sigma points, integrals over non-unit Gaussian weights functions $\mathrm{N}(x \mid m, P)$ can be evaluated using a simple change of integration variable:

$$\int_{-\infty}^{\infty} g(x)\ \mathrm{N}(x \mid m, P)\ \mathrm{d}x = \int_{-\infty}^{\infty} g(P^{1/2}\,\xi + m)\ \mathrm{N}(\xi \mid 0, 1)\ \mathrm{d}\xi. \tag{6.15}$$

Gauss–Hermite integration can be written as the following algorithm.

Algorithm 6.5 (Gauss–Hermite quadrature) *The pth order Gauss–Hermite approximation to the one-dimensional integral*

$$\int_{-\infty}^{\infty} g(x)\ \mathrm{N}(x \mid m, P)\ \mathrm{d}x \tag{6.16}$$

can be computed as follows:

1 *Compute the unit sigma points as the roots* $\xi^{(i)}, i = 1, \ldots, p$ *of the Hermite polynomial* $H_p(x)$. *Note that we do not need to form the polynomial and then compute its roots, but instead it is numerically more stable to compute the roots as eigenvalues of a suitable tridiagonal matrix (Golub and Welsch, 1969).*
2 *Compute the weights as*

$$W_i = \frac{p!}{p^2\,[H_{p-1}(\xi^{(i)})]^2}. \tag{6.17}$$

3 *Approximate the integral as*

$$\int_{-\infty}^{\infty} g(x)\ \mathrm{N}(x \mid m, P)\ \mathrm{d}x \approx \sum_{i=1}^{p} W_i\ g(P^{1/2}\,\xi^{(i)} + m). \tag{6.18}$$

By generalizing the change of variables idea, we can form approximations to multi-dimensional integrals of the form (6.10). First let $\mathbf{P} = \sqrt{\mathbf{P}}\,\sqrt{\mathbf{P}}^{\mathsf{T}}$, where $\sqrt{\mathbf{P}}$ is the Cholesky factor of the covariance matrix $\mathbf{P}$ or some other similar square root of the covariance matrix. If we define new integration variables $\boldsymbol{\xi}$ by

$$\mathbf{x} = \mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi}, \tag{6.19}$$

we get

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} = \int \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi})\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi}. \tag{6.20}$$

The integration over the multi-dimensional unit Gaussian distribution can be written as an iterated integral over one-dimensional Gaussian distributions, and each of the one-dimensional integrals can be approximated with Gauss–Hermite quadrature:

$$\begin{aligned}
&\int \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi})\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi} \\
&\quad = \int \cdots \int \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi})\ \mathrm{N}(\xi_1 \mid 0, 1)\ \mathrm{d}\xi_1 \times \cdots \times \mathrm{N}(\xi_n \mid 0, 1)\ \mathrm{d}\xi_n \\
&\quad \approx \sum_{i_1,\ldots,i_n} W_{i_1} \times \cdots \times W_{i_n} \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi}^{(i_1,\ldots,i_n)}).
\end{aligned} \tag{6.21}$$

The weights $W_{i_k}, k = 1, \ldots, n$ are simply the corresponding one-dimensional Gauss–Hermite weights and $\boldsymbol{\xi}^{(i_1,\ldots,i_n)}$ is an n-dimensional vector with one-dimensional unit sigma point $\xi^{(i_k)}$ at element k. The algorithm can now be written as follows.

Algorithm 6.6 (Gauss–Hermite cubature) *The pth order Gauss–Hermite approximation to the multi-dimensional integral*

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} \tag{6.22}$$

can be computed as follows.

1 *Compute the one-dimensional weights $W_i, i = 1, \ldots, p$ and unit sigma points $\xi^{(i)}$ as in the one-dimensional Gauss–Hermite quadrature Algorithm 6.5.*

2 *Form multi-dimensional weights as the products of one-dimensional weights:*

$$W_{i_1,\ldots,i_n} = W_{i_1} \times \cdots \times W_{i_n} = \frac{p!}{p^2\,[H_{p-1}(\xi^{(i_1)})]^2} \times \cdots \times \frac{p!}{p^2\,[H_{p-1}(\xi^{(i_n)})]^2}, \tag{6.23}$$

where each i_k *takes values* $1, \ldots, p$.

3 *Form multi-dimensional unit sigma points as Cartesian product of the one-dimensional unit sigma points:*

$$\boldsymbol{\xi}^{(i_1,\ldots,i_n)} = \begin{pmatrix} \xi^{(i_1)} \\ \vdots \\ \xi^{(i_n)} \end{pmatrix}. \tag{6.24}$$

4 *Approximate the integral as*

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} \approx \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi}^{(i_1,\ldots,i_n)}), \tag{6.25}$$

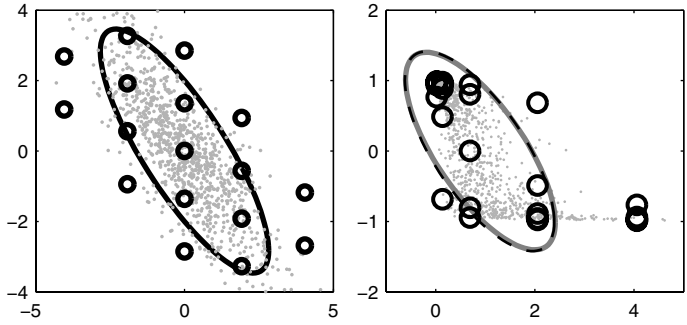
where $\sqrt{\mathbf{P}}$ *is a matrix square root defined by* $\mathbf{P} = \sqrt{\mathbf{P}}\,\sqrt{\mathbf{P}}^{\mathsf{T}}$.

The pth order multi-dimensional Gauss–Hermite integration is exact for monomials of the form $x_1^{d_1}\, x_2^{d_2} \cdots x_n^{d_n}$, and their arbitrary linear combinations, where each of the orders $d_i \leq 2p - 1$. The number of sigma points required for an n-dimensional integral with pth order rule is p^n, which quickly becomes unfeasible when the number of dimensions grows.

The Gaussian–Hermite cubature can also be seen as a sigma-point method where the sigma points are $\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi}^{(i_1,\ldots,i_n)}$. The difference from the unscented transform in Section 5.5 is that the sigma points and the weights are selected differently. Figure 6.1 illustrates how sigma points are selected in Gaussian–Hermite cubatures.

6.4 Gauss–Hermite Kalman filter

The additive form multi-dimensional Gauss–Hermite cubature based *Gauss–Hermite (Kalman) filter (GHKF)* (Ito and Xiong, 2000) which is also called the *quadrature Kalman filter (QKF)* (Arasaratnam et al., 2007) can be derived by replacing the Gaussian integrals in the Gaussian filter Algorithm 6.3 with the Gauss–Hermite approximations in Algorithm 6.6.



(a) Original (b) Transformed

Figure 6.1 Illustration of a fifth order Gauss–Hermite cubature based approximation to a non-linear transformation. The difference from the unscented transform (UT) illustrated in Figure 5.5 is that we are using more sigma points and they are placed differently. Due to the higher number of sigma points the covariance of the transformed random variable is captured more accurately. The covariance of the true distribution is presented by the dashed line and the solid line is the approximation.

Algorithm 6.7 (Gauss–Hermite Kalman filter) *The additive form Gauss–Hermite Kalman filter (GHKF) algorithm is the following.*

- *Prediction:*

  1 *Form the sigma points as:*

$$\mathcal{X}_{k-1}^{(i_1,\ldots,i_n)} = \mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\,\boldsymbol{\xi}^{(i_1,\ldots,i_n)}, \quad i_1,\ldots,i_n = 1,\ldots,p, \tag{6.26}$$

  where the unit sigma points $\boldsymbol{\xi}^{(i_1,\ldots,i_n)}$ *were defined in Equation* (6.24).

  2 *Propagate the sigma points through the dynamic model:*

$$\hat{\mathcal{X}}_k^{(i_1,\ldots,i_n)} = \mathbf{f}(\mathcal{X}_{k-1}^{(i_1,\ldots,i_n)}), \quad i_1,\ldots,i_n = 1,\ldots,p. \tag{6.27}$$

  3 *Compute the predicted mean* $\mathbf{m}_k^-$ *and the predicted covariance* $\mathbf{P}_k^-$*:*

$$\begin{aligned}
\mathbf{m}_k^- &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n}\,\hat{\mathcal{X}}_k^{(i_1,\ldots,i_n)},\\
\mathbf{P}_k^- &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n}\,(\hat{\mathcal{X}}_k^{(i_1,\ldots,i_n)} - \mathbf{m}_k^-)\,(\hat{\mathcal{X}}_k^{(i_1,\ldots,i_n)} - \mathbf{m}_k^-)^\mathsf{T} + \mathbf{Q}_{k-1},
\end{aligned} \tag{6.28}$$

where the weights $W_{i_1,\ldots,i_n}$ were defined in Equation (6.23).

- *Update:*

  1 *Form the sigma points:*

$$\mathcal{X}_k^{-(i_1,\ldots,i_n)} = \mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}^{(i_1,\ldots,i_n)}, \quad i_1,\ldots,i_n = 1,\ldots,p, \tag{6.29}$$

  where the unit sigma points $\boldsymbol{\xi}^{(i_1,\ldots,i_n)}$ were defined in Equation (6.24).

  2 *Propagate sigma points through the measurement model:*

$$\hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)} = \mathbf{h}(\mathcal{X}_k^{-(i_1,\ldots,i_n)}), \quad i_1,\ldots,i_n = 1,\ldots,p. \tag{6.30}$$

  3 *Compute the predicted mean $\boldsymbol{\mu}_k$, the predicted covariance of the measurement $\mathbf{S}_k$, and the cross-covariance of the state and the measurement $\mathbf{C}_k$:*

$$\begin{aligned}
\boldsymbol{\mu}_k &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} \hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)}, \\
\mathbf{S}_k &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} (\hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)} - \boldsymbol{\mu}_k)\,(\hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)} - \boldsymbol{\mu}_k)^\mathsf{T} + \mathbf{R}_k, \\
\mathbf{C}_k &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} (\mathcal{X}_k^{-(i_1,\ldots,i_n)} - \mathbf{m}_k^-)\,(\hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)} - \boldsymbol{\mu}_k)^\mathsf{T},
\end{aligned} \tag{6.31}$$

  where the weights $W_{i_1,\ldots,i_n}$ were defined in Equation (6.23).

  4 *Compute the filter gain $\mathbf{K}_k$, the filtered state mean $\mathbf{m}_k$ and the covariance $\mathbf{P}_k$, conditional on the measurement $\mathbf{y}_k$:*

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{C}_k\,\mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\,[\mathbf{y}_k - \boldsymbol{\mu}_k], \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\,\mathbf{S}_k\,\mathbf{K}_k^\mathsf{T}.
\end{aligned} \tag{6.32}$$

The non-additive version can be obtained by applying the Gauss–Hermite quadrature to the non-additive Gaussian filter Algorithm 6.4 in a similar manner. However, due to the rapid growth of computational requirements in state dimension the augmented form is computationally quite heavy, because it requires roughly doubling the dimensionality of the integration variable.

Example 6.1 (Pendulum tracking with GHKF) *The result of the GHKF in the pendulum model in Example 5.1 is shown in Figure 6.2. The result is the same as with the SLF and the UKF (RMSE = 0.11) which implies*

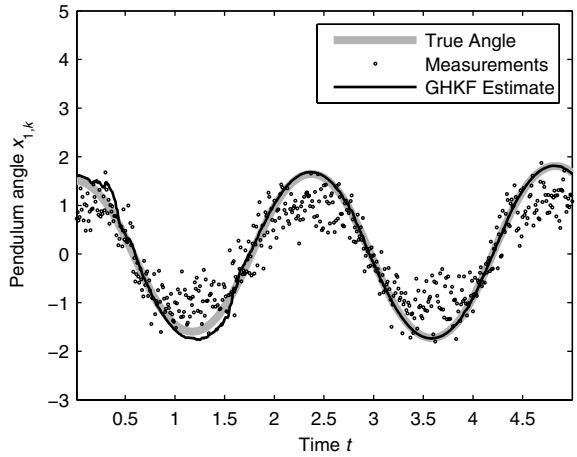


Figure 6.2 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with the GHKF. The resulting RMSE was 0.11 which is the same as with the SLF and the UKF.

that in this particular case the higher order approximation to the Gaussian integrals does not really help. It is possible that the consistency and stability properties of the filters are indeed different, but it is impossible to know based on this single test.

6.5 Spherical cubature integration

In this section, we derive the third order spherical cubature rule, which was popularized by Arasaratnam and Haykin (2009). However, instead of using the derivation of Arasaratnam and Haykin (2009), we shall use the derivation presented by Wu et al. (2006), due to its simplicity. Although the derivation that we present here is far simpler than the alternative, it is completely equivalent. Furthermore, the derivation presented here can be more easily extended to more complicated spherical cubatures.

Recall from Section 6.3 that the expectation of a non-linear function over an arbitrary Gaussian distribution $\mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})$ can always be transformed into an expectation over the unit Gaussian distribution $\mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})$. Thus,

we can start by considering the multi-dimensional unit Gaussian integral

$$\int \mathbf{g}(\boldsymbol{\xi})\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi}. \tag{6.33}$$

We now wish to form a $2n$-point approximation of the form

$$\int \mathbf{g}(\boldsymbol{\xi})\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi} \approx W \sum_i \mathbf{g}(c\,\mathbf{u}^{(i)}), \tag{6.34}$$

where the points $\mathbf{u}^{(i)}$ belong to the symmetric set $[\mathbf{1}]$ with generator $(1, 0, \ldots, 0)$ (see, e.g., Wu et al., 2006; Arasaratnam and Haykin, 2009):

$$[\mathbf{1}] = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \cdots \begin{pmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \cdots \right\}, \tag{6.35}$$

and W is a weight and c is a parameter yet to be determined.

Because the point set is symmetric, the rule is exact for all monomials of the form $x_1^{d_1}\, x_2^{d_2} \cdots x_n^{d_n}$, if at least one of the exponents d_i is odd. Thus we can construct a rule which is exact up to third degree by determining the coefficients W and c such that it is exact for selections $g_j(\boldsymbol{\xi}) = 1$ and $g_j(\boldsymbol{\xi}) = \xi_j^2$. Because the true values of the integrals are

$$\begin{aligned} \int \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi} &= 1, \\ \int \xi_j^2\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi} &= 1, \end{aligned} \tag{6.36}$$

we get the equations

$$\begin{aligned} W \sum_i 1 &= W\, 2n = 1, \\ W \sum_i [c\, u_j^{(i)}]^2 &= W\, 2c^2 = 1, \end{aligned} \tag{6.37}$$

which have the solutions

$$\begin{aligned} W &= \frac{1}{2n}, \\ c &= \sqrt{n}. \end{aligned} \tag{6.38}$$

That is, we get the following simple rule which is exact for monomials up to third degree:

$$\int \mathbf{g}(\boldsymbol{\xi})\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi} \approx \frac{1}{2n} \sum_i \mathbf{g}(\sqrt{n}\,\mathbf{u}^{(i)}). \tag{6.39}$$

We can now easily extend the method to arbitrary mean and covariance by using the change of variables in Equations (6.19) and (6.20) and the result is the following algorithm.

Algorithm 6.8 (Spherical cubature integration) *The third order spherical cubature approximation to the multi-dimensional integral*

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} \tag{6.40}$$

can be computed as follows.

1 *Compute the unit sigma points as*

$$\boldsymbol{\xi}^{(i)} = \begin{cases} \sqrt{n}\,\mathbf{e}_i, & i = 1, \ldots, n, \\ -\sqrt{n}\,\mathbf{e}_{i-n}, & i = n+1, \ldots, 2n, \end{cases} \tag{6.41}$$

where $\mathbf{e}_i$ *denotes a unit vector in the direction of the coordinate axis* i.

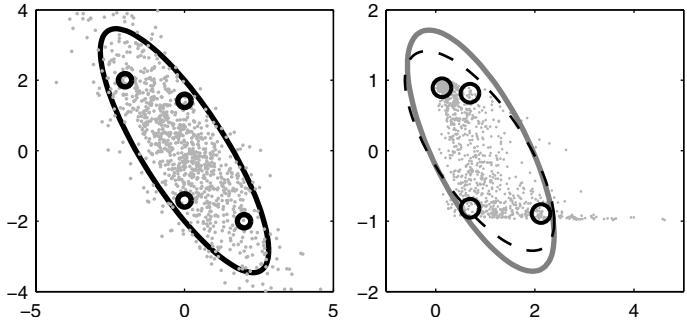
2 *Approximate the integral as*

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} \approx \frac{1}{2n} \sum_{i=1}^{2n} \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi}^{(i)}), \tag{6.42}$$

where $\sqrt{\mathbf{P}}$ *is a matrix square root defined by* $\mathbf{P} = \sqrt{\mathbf{P}}\,\sqrt{\mathbf{P}}^{\mathsf{T}}$.

It is easy to see that the approximation above is a special case of the unscented transform (see Section 5.5) with parameters $\alpha = 1$, $\beta = 0$, and $\kappa = 0$. With this parameter selection the mean weight is zero and the unscented transform is effectively a $2n$-point approximation as well. The selection of sigma points in spherical cubature integration is illustrated in Figure 6.3.

The derivation presented by Arasaratnam and Haykin (2009) is a bit more complicated than the derivation of Wu et al. (2006) presented above, as it is based on converting the Gaussian integral into spherical coordinates and then considering the even order monomials. However, Wu et al. (2006) actually did not present the most useful special case given in Algorithm 6.8, but instead presented the method for more general generators $[\mathbf{u}]$. The method in the above Algorithm 6.8 has the useful property that



(a) Original (b) Transformed

Figure 6.3 Illustration of the third order spherical cubature based approximation to a non-linear transformation. The difference from the unscented transform (UT) illustrated in Figure 5.5 is that we have one sigma point fewer and the spread of the sigma points is fixed (in UT it is parametrized). The covariance of the true distribution is presented by the dashed line and the solid line is the approximation.

its weights are always positive, which is not always true for more general methods (Wu et al., 2006).

We can generalize the above approach by using a $2n + 1$ point approximation, where the origin is also included:

$$\int \mathbf{g}(\boldsymbol{\xi})\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi} \approx W_0\, \mathbf{g}(\mathbf{0}) + W \sum_i \mathbf{g}(c\, \mathbf{u}^{(i)}). \tag{6.43}$$

We can now solve for the parameters W_0, W, and c such that we get the exact result with selections $g_j(\boldsymbol{\xi}) = 1$ and $g_j(\boldsymbol{\xi}) = \xi_j^2$. The solution can be written in the form

$$\begin{aligned} W_0 &= \frac{\kappa}{n + \kappa}, \\ W &= \frac{1}{2(n + \kappa)}, \\ c &= \sqrt{n + \kappa}, \end{aligned} \tag{6.44}$$

where κ is a free parameter. This gives an integration rule that can be written as

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} \approx \frac{\kappa}{n+\kappa}\mathbf{g}(\mathbf{m}) + \frac{1}{2(n+\kappa)}\sum_{i=1}^{2n}\mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi}^{(i)}), \tag{6.45}$$

where

$$\boldsymbol{\xi}^{(i)} = \begin{cases} \sqrt{n+\kappa}\,\mathbf{e}_i, & i = 1, \ldots, n, \\ -\sqrt{n+\kappa}\,\mathbf{e}_{i-n}, & i = n+1, \ldots, 2n. \end{cases} \tag{6.46}$$

The rule can be seen to coincide with the original UT (Julier and Uhlmann, 1995), which corresponds to the unscented transform presented in Section 5.5 with $\alpha = 1$, $\beta = 0$, and where κ is left as a free parameter. With the selection $\kappa = 3-n$, we can also match the fourth order moments of the distribution (Julier and Uhlmann, 1995), but with the price that when the dimensionality $n > 3$, we get negative weights and approximation rules that can sometimes be unstable. But nothing prevents us from using other values for the parameter.

Note that "third order" here means a different thing than in the Gauss–Hermite Kalman filter – the pth order Gauss–Hermite filter is exact for monomials up to order $2p-1$, which means that the third order GHKF is exact for monomials up to fifth order. The third order spherical cubature rule is exact only for monomials up to third order. It is also possible to derive symmetric rules that are exact for higher than third order. However, this is no longer possible with a number of sigma points which is linear $O(n)$ in state dimension (Wu et al., 2006; Arasaratnam and Haykin, 2009). For example, for a fifth order rule, the required number of sigma points is proportional to n^2, the state dimension squared.

As in the case of the unscented transform, being exact up to order three only ensures that the estimate of the mean of $\mathbf{g}(\cdot)$ is exact for polynomials of order three. The covariance will be exact only for polynomials up to order one (linear functions). In this sense the third order spherical cubature rule is actually a first order spherical cubature rule for the covariance.

6.6 Cubature Kalman filter

When we apply the third order spherical cubature integration rule in Algorithm 6.8 to the Gaussian filter equations in Algorithm 6.3, we get the cubature Kalman filter (CKF) of Arasaratnam and Haykin (2009).

Algorithm 6.9 (Cubature Kalman filter I) *The additive form of the cubature Kalman filter (CKF) algorithm is the following.*

- *Prediction:*
  1 *Form the sigma points as:*

$$\mathcal{X}_{k-1}^{(i)} = \mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\,\boldsymbol{\xi}^{(i)}, \qquad i = 1, \ldots, 2n, \tag{6.47}$$

  where the unit sigma points are defined as

$$\boldsymbol{\xi}^{(i)} = \begin{cases} \sqrt{n}\,\mathbf{e}_i, & i = 1, \ldots, n, \\ -\sqrt{n}\,\mathbf{e}_{i-n}, & i = n+1, \ldots, 2n. \end{cases} \tag{6.48}$$

  2 *Propagate the sigma points through the dynamic model:*

$$\hat{\mathcal{X}}_k^{(i)} = \mathbf{f}(\mathcal{X}_{k-1}^{(i)}), \quad i = 1, \ldots, 2n. \tag{6.49}$$

  3 *Compute the predicted mean* $\mathbf{m}_k^-$ *and the predicted covariance* $\mathbf{P}_k^-$*:*

$$\begin{aligned} \mathbf{m}_k^- &= \frac{1}{2n}\sum_{i=1}^{2n} \hat{\mathcal{X}}_k^{(i)}, \\ \mathbf{P}_k^- &= \frac{1}{2n}\sum_{i=1}^{2n} (\hat{\mathcal{X}}_k^{(i)} - \mathbf{m}_k^-)\,(\hat{\mathcal{X}}_k^{(i)} - \mathbf{m}_k^-)^{\mathsf{T}} + \mathbf{Q}_{k-1}. \end{aligned} \tag{6.50}$$

- *Update:*
  1 *Form the sigma points:*

$$\mathcal{X}_k^{-(i)} = \mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}^{(i)}, \qquad i = 1, \ldots, 2n, \tag{6.51}$$

  where the unit sigma points are defined as in Equation (6.48)*.*
  2 *Propagate sigma points through the measurement model:*

$$\hat{\mathcal{Y}}_k^{(i)} = \mathbf{h}(\mathcal{X}_k^{-(i)}), \quad i = 1 \ldots 2n. \tag{6.52}$$

  3 *Compute the predicted mean* $\boldsymbol{\mu}_k$*, the predicted covariance of the measurement* $\mathbf{S}_k$*, and the cross-covariance of the state and the*

measurement $\mathbf{C}_k$*:*

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{2n}\sum_{i=1}^{2n} \hat{\mathcal{Y}}_k^{(i)}, \\
\mathbf{S}_k &= \frac{1}{2n}\sum_{i=1}^{2n} (\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)\,(\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)^\mathsf{T} + \mathbf{R}_k, \\
\mathbf{C}_k &= \frac{1}{2n}\sum_{i=1}^{2n} (\mathcal{X}_k^{-(i)} - \mathbf{m}_k^-)\,(\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)^\mathsf{T}.
\end{aligned} \tag{6.53}$$

4 *Compute the filter gain* $\mathbf{K}_k$ *and the filtered state mean* $\mathbf{m}_k$ *and covariance* $\mathbf{P}_k$*, conditional on the measurement* $\mathbf{y}_k$*:*

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{C}_k\,\mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\,[\mathbf{y}_k - \boldsymbol{\mu}_k], \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\,\mathbf{S}_k\,\mathbf{K}_k^\mathsf{T}.
\end{aligned} \tag{6.54}$$

By applying the cubature rule to the non-additive Gaussian filter in Algorithm 6.4 we get the following augmented form of the cubature Kalman filter (CKF).

Algorithm 6.10 (Cubature Kalman filter II) *The augmented non-additive form of the cubature Kalman filter (CKF) algorithm is the following.*

- *Prediction:*

1 *Form the matrix of sigma points for the augmented random variable* $(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})$*:*

$$\tilde{\mathcal{X}}_{k-1}^{(i)} = \tilde{\mathbf{m}}_{k-1} + \sqrt{\tilde{\mathbf{P}}_{k-1}}\,\boldsymbol{\xi}^{(i)'}, \qquad i = 1, \ldots, 2n', \tag{6.55}$$

where

$$\tilde{\mathbf{m}}_{k-1} = \begin{pmatrix} \mathbf{m}_{k-1} \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\mathbf{P}}_{k-1} = \begin{pmatrix} \mathbf{P}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{k-1} \end{pmatrix}.$$

Here $n' = n + n_q$*, where* n *is the dimensionality of the state* $\mathbf{x}_{k-1}$ *and* n_q *is the dimensionality of the noise* $\mathbf{q}_{k-1}$*. The unit sigma points are defined as*

$$\boldsymbol{\xi}^{(i)'} = \begin{cases} \sqrt{n'}\,\mathbf{e}_i, & i = 1, \ldots, n', \\ -\sqrt{n'}\,\mathbf{e}_{i-n'}, & i = n'+1, \ldots, 2n'. \end{cases} \tag{6.56}$$

2 *Propagate the sigma points through the dynamic model:*

$$\hat{\mathcal{X}}_k^{(i)} = \mathbf{f}(\tilde{\mathcal{X}}_{k-1}^{(i),x}, \tilde{\mathcal{X}}_{k-1}^{(i),q}), \quad i = 1, \ldots, 2n', \tag{6.57}$$

where $\tilde{\mathcal{X}}_{k-1}^{(i),x}$ *denotes the first* n *components in* $\tilde{\mathcal{X}}_{k-1}^{(i)}$ *and* $\tilde{\mathcal{X}}_{k-1}^{(i),q}$ *denotes the last* n_q *components.*

3 *Compute the predicted mean* $\mathbf{m}_k^-$ *and the predicted covariance* $\mathbf{P}_k^-$*:*

$$\begin{aligned} \mathbf{m}_k^- &= \frac{1}{2n'} \sum_{i=1}^{2n'} \hat{\mathcal{X}}_k^{(i)}, \\ \mathbf{P}_k^- &= \frac{1}{2n'} \sum_{i=1}^{2n'} (\hat{\mathcal{X}}_k^{(i)} - \mathbf{m}_k^-)\,(\hat{\mathcal{X}}_k^{(i)} - \mathbf{m}_k^-)^\top. \end{aligned} \tag{6.58}$$

- *Update:*

1 *Let* $n'' = n + n_r$*, where* n *is the dimensionality of the state and* n_r *is the dimensionality of the measurement noise. Form the sigma points for the augmented vector* $(\mathbf{x}_k, \mathbf{r}_k)$ *as follows:*

$$\tilde{\mathcal{X}}_k^{-(i)} = \tilde{\mathbf{m}}_k^- + \sqrt{\tilde{\mathbf{P}}_k^-}\, \boldsymbol{\xi}^{(i)''}, \qquad i = 1, \ldots, 2n'', \tag{6.59}$$

where

$$\tilde{\mathbf{m}}_k^- = \begin{pmatrix} \mathbf{m}_k^- \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\mathbf{P}}_k^- = \begin{pmatrix} \mathbf{P}_k^- & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k \end{pmatrix}.$$

The unit sigma points $\boldsymbol{\xi}^{(i)''}$ *are defined as in Equation* (6.56)*, but with* n' *replaced by* n''*.*

2 *Propagate the sigma points through the measurement model:*

$$\hat{\mathcal{Y}}_k^{(i)} = \mathbf{h}(\tilde{\mathcal{X}}_k^{-(i),x}, \tilde{\mathcal{X}}_k^{-(i),r}), \quad i = 1, \ldots, 2n'', \tag{6.60}$$

where $\tilde{\mathcal{X}}_k^{-(i),x}$ *denotes the first* n *components in* $\tilde{\mathcal{X}}_k^{-(i)}$ *and* $\tilde{\mathcal{X}}_k^{-(i),r}$ *denotes the last* n_r *components.*

3 *Compute the predicted mean* $\boldsymbol{\mu}_k$*, the predicted covariance of the measurement* $\mathbf{S}_k$*, and the cross-covariance of the state and the*

measurement $\mathbf{C}_k$*:*

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{2n''} \sum_{i=1}^{2n''} \hat{\mathcal{Y}}_k^{(i)}, \\
\mathbf{S}_k &= \frac{1}{2n''} \sum_{i=1}^{2n''} (\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k) \, (\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)^{\mathsf{T}}, \\
\mathbf{C}_k &= \frac{1}{2n''} \sum_{i=1}^{2n''} (\mathcal{X}_k^{-(i),x} - \mathbf{m}_k^-) \, (\hat{\mathcal{Y}}_k^{(i)} - \boldsymbol{\mu}_k)^{\mathsf{T}}.
\end{aligned} \tag{6.61}$$

4 *Compute the filter gain* $\mathbf{K}_k$*, the filtered state mean* $\mathbf{m}_k$ *and the covariance* $\mathbf{P}_k$*, conditional on the measurement* $\mathbf{y}_k$*:*

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{C}_k \, \mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k \, [\mathbf{y}_k - \boldsymbol{\mu}_k], \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k \, \mathbf{S}_k \, \mathbf{K}_k^{\mathsf{T}}.
\end{aligned} \tag{6.62}$$

Although in the cubature Kalman filter (CKF) literature the "third order" characteristic of the cubature integration rule is often emphasized (see Arasaratnam and Haykin, 2009), it is important to remember that in the covariance computation, the rule is only exact for first order polynomials. Thus in that sense CKF is a first order method.

Example 6.2 (Pendulum tracking with CKF) *The result of the CKF in the pendulum model (Example 5.1) is shown in Figure 6.4. The result is practically the same as the result of the UKF, which was to be expected, because the CKF is just a UKF with a specific parametrization.*

6.7 Exercises

6.1 Show that the selection $\kappa = 3 - n$ in Equation (6.45) causes fourth order terms ξ_i^4 to be integrated exactly when $\mathbf{m} = \mathbf{0}$ and $\mathbf{P} = \mathbf{I}$. What happens to the weights if $n > 3$?

6.2 Show that when the function is linear, both the unscented transform and the spherical cubature rule give the exact result.

6.3 Show that one-dimensional Hermite polynomials are orthogonal with respect to the inner product

$$\langle f, g \rangle = \int f(x) \, g(x) \, \mathrm{N}(x \mid 0, 1) \, \mathrm{d}x. \tag{6.63}$$

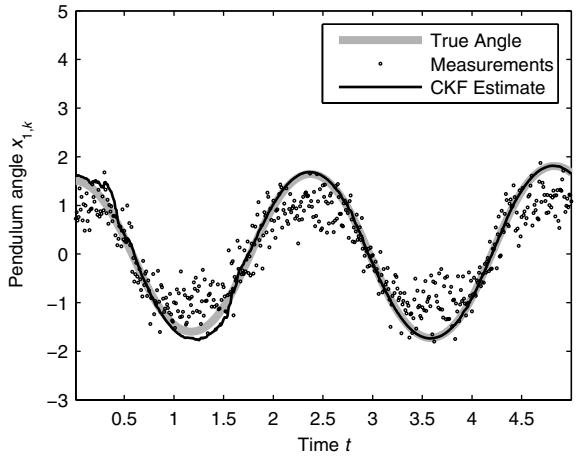


Figure 6.4 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with the CKF. The resulting RMSE was 0.11 which is practically the same as the error of the UKF. In fact, the result is practically the same as with all the other non-linear Kalman filters encountered in this book so far.

6.4 Show that multivariate Hermite polynomials defined as

$$H_{i_1,\ldots,i_n}(\mathbf{x}) = H_{i_1}(x_1) \times \cdots \times H_{i_n}(x_n) \tag{6.64}$$

are orthogonal with respect to the inner product

$$\langle f, g \rangle = \int f(\mathbf{x})\, g(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\mathbf{x}. \tag{6.65}$$

6.5 Implement a GHKF for the model in Exercise 5.1. Plot the results and compare the RMSE values with the EKF, SLF, and UKF.

6.6 Implement a CKF for the model in Exercise 5.1. Plot the results and compare the RMSE values with the other filters. Can you find such parameter values for the methods which cause the UKF, GHKF, and CKF methods to become identical?

6.7 Implement a CKF for the bearings only target tracking problem in Exercise 5.5. Compare the performance with the EKF and UKF.

7

Particle filtering

Although in many filtering problems Gaussian approximations work well, sometimes the filtering distributions can be, for example, multi-modal, or some of the state components might be discrete, in which case Gaussian approximations are not appropriate. In such cases sequential importance resampling based particle filters can be a better alternative. This chapter is concerned with particle filters, which are methods for forming Monte Carlo approximations to the solutions of the Bayesian filtering equations.

7.1 Monte Carlo approximations in Bayesian inference

In Bayesian inference, including Bayesian filtering, the main inference problem can often be reduced into computation of the following kind of expectations over the posterior distribution:[1]

$$\mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] = \int \mathbf{g}(\mathbf{x})\ p(\mathbf{x} \mid \mathbf{y}_{1:T})\ \mathrm{d}\mathbf{x}, \tag{7.1}$$

where $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$ is an arbitrary function and $p(\mathbf{x} \mid \mathbf{y}_{1:T})$ is the posterior probability density of $\mathbf{x}$ given the measurements $\mathbf{y}_1, \ldots, \mathbf{y}_T$. Now the problem is that such an integral can be evaluated in closed form only in a few special cases and generally, numerical methods have to be used.

Monte Carlo methods provide a numerical method for calculating integrals of the form (7.1). Monte Carlo refers to a general class of methods where closed form computation of statistical quantities is replaced by drawing samples from the distribution and estimating the quantities by sample averages.

In a (perfect) Monte Carlo approximation, we draw N independent random samples $\mathbf{x}^{(i)} \sim p(\mathbf{x} \mid \mathbf{y}_{1:T})$, $i = 1, \ldots, N$ and estimate the

[1] In this section we formally treat $\mathbf{x}$ as a continuous random variable with a density, but the analogous results apply to discrete random variables.

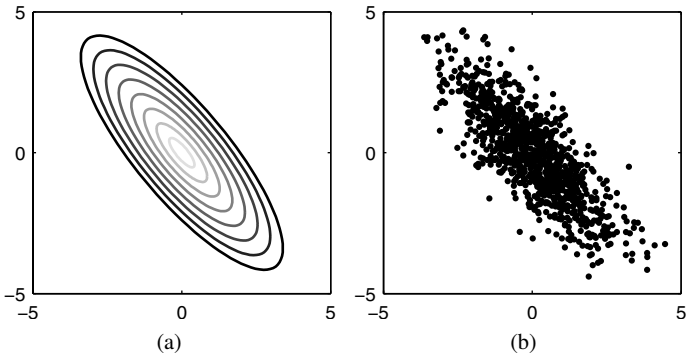


Figure 7.1 (a) Two-dimensional Gaussian distribution. (b) Monte Carlo representation of the same Gaussian distribution.

expectation as

$$\mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{g}(\mathbf{x}^{(i)}). \tag{7.2}$$

Thus Monte Carlo methods approximate the target distribution by a set of samples that are distributed according to the target density. Figure 7.1 represents a two-dimensional Gaussian distribution and its Monte Carlo representation.

The convergence of the Monte Carlo approximation is guaranteed by the central limit theorem (CLT, see, e.g., Liu, 2001) and the error term is $O(N^{-1/2})$, regardless of the dimensionality of $\mathbf{x}$. This invariance with respect to dimensionality is unique to Monte Carlo methods and makes them superior to practically all other numerical methods when the dimensionality of $\mathbf{x}$ is considerable. At least in theory, not necessarily in practice (see Daum and Huang, 2003; Snyder et al., 2008).

7.2 Importance sampling

Often, in practical Bayesian models, it is not possible to obtain samples directly from $p(\mathbf{x} \mid \mathbf{y}_{1:T})$ due to its complicated functional form. In *importance sampling* (IS) (see, e.g., Liu, 2001) we use an approximate distribution called the importance distribution $\pi(\mathbf{x} \mid \mathbf{y}_{1:T})$, from which we can easily draw samples. Importance sampling is based on the following decomposition of the expectation over the posterior probability density

$p(\mathbf{x} \mid \mathbf{y}_{1:T})$:

$$\int \mathbf{g}(\mathbf{x})\, p(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x} = \int \left[\mathbf{g}(\mathbf{x})\, \frac{p(\mathbf{x} \mid \mathbf{y}_{1:T})}{\pi(\mathbf{x} \mid \mathbf{y}_{1:T})}\right] \pi(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x}, \quad (7.3)$$

where the importance density $\pi(\mathbf{x} \mid \mathbf{y}_{1:T})$ is required to be non-zero whenever $p(\mathbf{x} \mid \mathbf{y}_{1:T})$ is non-zero, that is, the *support* of $\pi(\mathbf{x} \mid \mathbf{y}_{1:T})$ needs to be greater than or equal to the support of $p(\mathbf{x} \mid \mathbf{y}_{1:T})$. As the above expression is just the expectation of the term in the brackets over the distribution $\pi(\mathbf{x} \mid \mathbf{y}_{1:T})$, we can form a Monte Carlo approximation to it by drawing N samples from the importance distribution:

$$\mathbf{x}^{(i)} \sim \pi(\mathbf{x} \mid \mathbf{y}_{1:T}), \qquad i = 1, \ldots, N, \quad (7.4)$$

and by forming the approximation as

$$\begin{aligned} \mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] &\approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}{\pi(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}\, \mathbf{g}(\mathbf{x}^{(i)}) \\ &= \sum_{i=1}^{N} \tilde{w}^{(i)}\, \mathbf{g}(\mathbf{x}^{(i)}), \end{aligned} \quad (7.5)$$

where the weights have been defined as

$$\tilde{w}^{(i)} = \frac{1}{N}\, \frac{p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}{\pi(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}. \quad (7.6)$$

Figure 7.2 illustrates the idea of importance sampling. We sample from the importance distribution which is an approximation to the target distribution. Because the distribution of samples is not exact, we need to correct the approximation by associating a weight with each of the samples.

The disadvantage of this direct importance sampling is that we should be able to evaluate $p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})$ in order to use it directly. Recall that by Bayes' rule the evaluation of the posterior probability density can be written as

$$p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T} \mid \mathbf{x}^{(i)})\, p(\mathbf{x}^{(i)})}{\int p(\mathbf{y}_{1:T} \mid \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}}. \quad (7.7)$$

The likelihood $p(\mathbf{y}_{1:T} \mid \mathbf{x}^{(i)})$ and prior terms $p(\mathbf{x}^{(i)})$ are usually easy to evaluate but often the integral in the denominator – the normalization constant – cannot be computed. To overcome this problem, we can form an importance sampling approximation to the expectation integral by also approximating the normalization constant by importance sampling. For this

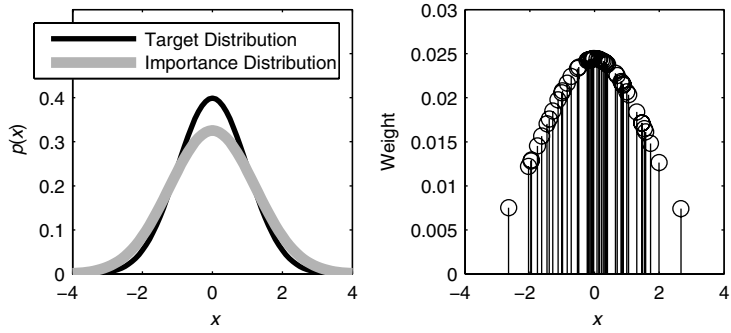


Figure 7.2 (a) The importance distribution approximates the target distribution. (b) Weights are associated with each of the samples to correct the approximation.

purpose we can decompose the expectation integral and form the approximation as follows.

$$
\begin{aligned}
\mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] &= \int \mathbf{g}(\mathbf{x})\, p(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x} \\
&= \frac{\int \mathbf{g}(\mathbf{x})\, p(\mathbf{y}_{1:T} \mid \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}}{\int p(\mathbf{y}_{1:T} \mid \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}} \\
&= \frac{\int \left[\frac{p(\mathbf{y}_{1:T}|\mathbf{x})\, p(\mathbf{x})}{\pi(\mathbf{x}|\mathbf{y}_{1:T})} \mathbf{g}(\mathbf{x}) \right] \pi(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x}}{\int \left[\frac{p(\mathbf{y}_{1:T}|\mathbf{x})\, p(\mathbf{x})}{\pi(\mathbf{x}|\mathbf{y}_{1:T})} \right] \pi(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x}} \\
&\approx \frac{\frac{1}{N} \sum_{i=1}^{N} \frac{p(\mathbf{y}_{1:T}|\mathbf{x}^{(i)})\, p(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)}|\mathbf{y}_{1:T})} \mathbf{g}(\mathbf{x}^{(i)})}{\frac{1}{N} \sum_{j=1}^{N} \frac{p(\mathbf{y}_{1:T}|\mathbf{x}^{(j)})\, p(\mathbf{x}^{(j)})}{\pi(\mathbf{x}^{(j)}|\mathbf{y}_{1:T})}} \\
&= \sum_{i=1}^{N} \underbrace{\left[\frac{\frac{p(\mathbf{y}_{1:T}|\mathbf{x}^{(i)})\, p(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)}|\mathbf{y}_{1:T})}}{\sum_{j=1}^{N} \frac{p(\mathbf{y}_{1:T}|\mathbf{x}^{(j)})\, p(\mathbf{x}^{(j)})}{\pi(\mathbf{x}^{(j)}|\mathbf{y}_{1:T})}} \right]}_{w^{(i)}} \mathbf{g}(\mathbf{x}^{(i)}).
\end{aligned}
\tag{7.8}
$$

Thus we get the following algorithm.

Algorithm 7.1 (Importance sampling) *Given a measurement model* $p(\mathbf{y}_{1:T} \mid \mathbf{x})$ *and a prior* $p(\mathbf{x})$ *we can form an importance sampling approximation to the posterior as follows.*

1 Draw N samples from the importance distribution:

$$\mathbf{x}^{(i)} \sim \pi(\mathbf{x} \mid \mathbf{y}_{1:T}), \qquad i = 1, \ldots, N. \tag{7.9}$$

2 Compute the unnormalized weights by

$$w^{*(i)} = \frac{p(\mathbf{y}_{1:T} \mid \mathbf{x}^{(i)})\, p(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}, \tag{7.10}$$

and the normalized weights by

$$w^{(i)} = \frac{w^{*(i)}}{\sum_{j=1}^{N} w^{*(j)}}. \tag{7.11}$$

3 The approximation to the posterior expectation of $\mathbf{g}(\mathbf{x})$ is then given as

$$\mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] \approx \sum_{i=1}^{N} w^{(i)}\, \mathbf{g}(\mathbf{x}^{(i)}). \tag{7.12}$$

The approximation to the posterior probability density formed by the above algorithm can then be formally written as

$$p(\mathbf{x} \mid \mathbf{y}_{1:T}) \approx \sum_{i=1}^{N} w^{(i)}\, \delta(\mathbf{x} - \mathbf{x}^{(i)}), \tag{7.13}$$

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

7.3 Sequential importance sampling

Sequential importance sampling (SIS) (see, e.g., Doucet et al., 2001) is a sequential version of importance sampling. The SIS algorithm can be used for generating importance sampling approximations to filtering distributions of generic state space models of the form

$$\begin{aligned} \mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}), \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k), \end{aligned} \tag{7.14}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state at time step k and $\mathbf{y}_k \in \mathbb{R}^m$ is the measurement. The state and measurements may contain both discrete and continuous components.

The SIS algorithm uses a weighted set of *particles* $\{(w_k^{(i)}, \mathbf{x}_k^{(i)}) : i = 1, \ldots, N\}$, that is, samples from an importance distribution and their weights, for representing the filtering distribution $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ such that

at every time step k the approximation to the expectation of an arbitrary function $\mathbf{g}(\cdot)$ can be calculated as the weighted sample average

$$\mathrm{E}[\mathbf{g}(\mathbf{x}_k) \mid \mathbf{y}_{1:k}] \approx \sum_{i=1}^{N} w_k^{(i)} \mathbf{g}(\mathbf{x}_k^{(i)}). \tag{7.15}$$

Equivalently, SIS can be interpreted as forming an approximation to the filtering distribution as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}). \tag{7.16}$$

To derive the algorithm, we consider the full posterior distribution of states $\mathbf{x}_{0:k}$ given the measurements $\mathbf{y}_{1:k}$. By using the Markov properties of the model, we get the following recursion for the posterior distribution:

$$\begin{aligned} p(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k}) &\propto p(\mathbf{y}_k \mid \mathbf{x}_{0:k}, \mathbf{y}_{1:k-1})\, p(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k-1}) \\ &= p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k-1})\, p(\mathbf{x}_{0:k-1} \mid \mathbf{y}_{1:k-1}) \\ &= p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{0:k-1} \mid \mathbf{y}_{1:k-1}). \end{aligned} \tag{7.17}$$

Using a similar rationale as in the previous section, we can now construct an importance sampling method which draws samples from a given importance distribution $\mathbf{x}_{0:k}^{(i)} \sim \pi(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k})$ and computes the importance weights by

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})\, p(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}{\pi(\mathbf{x}_{0:k}^{(i)} \mid \mathbf{y}_{1:k})}. \tag{7.18}$$

If we form the importance distribution for the states $\mathbf{x}_k$ recursively as follows:

$$\pi(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k}) = \pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})\, \pi(\mathbf{x}_{0:k-1} \mid \mathbf{y}_{1:k-1}), \tag{7.19}$$

then the expression for the weights can be written as

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})} \frac{p(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}{\pi(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}. \tag{7.20}$$

Let's now assume that we have already drawn the samples $\mathbf{x}_{0:k-1}^{(i)}$ from the importance distribution $\pi(\mathbf{x}_{0:k-1} \mid \mathbf{y}_{1:k-1})$ and computed the corresponding importance weights $w_{k-1}^{(i)}$. We can now draw samples $\mathbf{x}_{0:k}^{(i)}$ from the importance distribution $\pi(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k})$ by drawing the new state samples

for the step k as $\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})$. The importance weights from the previous step are proportional to the last term in Equation (7.20):

$$w_{k-1}^{(i)} \propto \frac{p(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}{\pi(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}, \tag{7.21}$$

and thus the weights satisfy the recursion

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})}\, w_{k-1}^{(i)}. \tag{7.22}$$

The generic sequential importance sampling algorithm can now be described as follows.

Algorithm 7.2 (Sequential importance sampling) *Steps of SIS are the following:*

- *Draw N samples $\mathbf{x}_0^{(i)}$ from the prior*

$$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), \qquad i = 1, \ldots, N, \tag{7.23}$$

  and set $w_0^{(i)} = 1/N$, for all $i = 1, \ldots, N$.
- *For each $k = 1, \ldots, T$ do the following.*

  1 Draw samples $\mathbf{x}_k^{(i)}$ from the importance distributions

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k}), \qquad i = 1, \ldots, N. \tag{7.24}$$

  2 Calculate new weights according to

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})} \tag{7.25}$$

  and normalize them to sum to unity.

Note that it is convenient to select the importance distribution to be Markovian in the sense that

$$\pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) = \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_{1:k}). \tag{7.26}$$

With this form of importance distribution we do not need to store the whole histories $\mathbf{x}_{0:k}^{(i)}$ in the SIS algorithm, only the current states $\mathbf{x}_k^{(i)}$. This form is also convenient in sequential importance resampling (SIR) discussed in the next section, because we do not need to worry about the state histories during the resampling step as in the SIR particle smoother (see Section 11.1). Thus in the following section we assume that the importance distribution has indeed been selected to have the above Markovian form.

7.4 Sequential importance resampling

One problem in the SIS algorithm described in the previous section is that we easily encounter the situation that almost all the particles have zero or nearly zero weights. This is called the *degeneracy problem* in particle filtering literature and it prevented practical applications of particle filters for many years.

The degeneracy problem can be solved by using a *resampling* procedure. It refers to a procedure where we draw N new samples from the discrete distribution defined by the weights and replace the old set of N samples with this new set. This procedure can be be written as the following algorithm:

Algorithm 7.3 (Resampling) *The resampling procedure can be described as follows.*

1 *Interpret each weight* $w_k^{(i)}$ *as the probability of obtaining the sample index* i *in the set* $\{\mathbf{x}_k^{(i)} : i = 1, \ldots, N\}$.
2 *Draw* N *samples from that discrete distribution and replace the old sample set with this new one.*
3 *Set all weights to the constant value* $w_k^{(i)} = 1/N$.

The idea of the resampling procedure is to remove particles with very small weights and duplicate particles with large weights. Although the theoretical distribution represented by the weighted set of samples does not change, resampling introduces additional variance to estimates. This variance introduced by the resampling procedure can be reduced by proper choice of the resampling method. The *stratified resampling* algorithm (Kitagawa, 1996) is optimal in terms of variance.

Adding a resampling step to the sequential importance sampling algorithm leads to *sequential importance resampling (SIR)*[2] (Gordon et al., 1993; Kitagawa, 1996; Doucet et al., 2001; Ristic et al., 2004), which is the algorithm usually referred to as the *particle filter*. In SIR, resampling is not usually performed at every time step, but only when it is actually needed. One way of implementing this is to do resampling on every nth step, where n is some predefined constant. This method has the advantage that it is unbiased. Another way, which is used here, is *adaptive resampling*. In this method, the "effective" number of particles, which is estimated from

[2] *Sequential importance resampling* is also often referred to as *sampling importance resampling or* sequential importance sampling resampling.

the variance of the particle weights (Liu and Chen, 1995), is used for monitoring the need for resampling. The estimate for the effective number of particles can be computed as:

$$n_{\text{eff}} \approx \frac{1}{\sum_{i=1}^{N} \left(w_k^{(i)}\right)^2}, \tag{7.27}$$

where $w_k^{(i)}$ is the normalized weight of particle i at the time step k (Liu and Chen, 1995). Resampling is performed when the effective number of particles is significantly less than the total number of particles, for example, $n_{\text{eff}} < N/10$, where N is the total number of particles.

Algorithm 7.4 (Sequential importance resampling) *The sequential importance resampling (SIR) algorithm, which is also called the particle filter (PF), is the following.*

- *Draw N samples $\mathbf{x}_0^{(i)}$ from the prior*

$$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), \qquad i = 1, \ldots, N, \tag{7.28}$$

  and set $w_0^{(i)} = 1/N$, for all $i = 1, \ldots, N$.
- *For each $k = 1, \ldots, T$ do the following:*
  1 *Draw samples $\mathbf{x}_k^{(i)}$ from the importance distributions*

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k}), \qquad i = 1, \ldots, N. \tag{7.29}$$

  2 *Calculate new weights according to*

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\ p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k})} \tag{7.30}$$

  and normalize them to sum to unity.
  3 *If the effective number of particles* (7.27) *is too low, perform resampling.*

The SIR algorithm forms the following approximation to the filtering distribution:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}), \tag{7.31}$$

and thus the expectation of an arbitrary function $\mathbf{g}(\cdot)$ can be approximated as

$$\mathrm{E}[\mathbf{g}(\mathbf{x}_k) \mid \mathbf{y}_{1:k}] \approx \sum_{i=1}^{N} w_k^{(i)} \mathbf{g}(\mathbf{x}_k^{(i)}). \tag{7.32}$$

Performance of the SIR algorithm depends on the quality of the importance distribution $\pi(\cdot)$. The importance distribution should be in such a functional form that we can easily draw samples from it and that it is possible to evaluate the probability densities of the sample points. *The optimal importance distribution* in terms of variance (see, e.g., Doucet et al., 2001; Ristic et al., 2004) is

$$\pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_k). \tag{7.33}$$

If the optimal importance distribution cannot be directly used, importance distributions can sometimes be obtained by *local linearization* where a mixture of extended Kalman filters (EKF), unscented Kalman filters (UKF) or other types of non-linear Kalman filter are used for forming the importance distribution (Doucet et al., 2000; Van der Merwe et al., 2001). It is also possible to use a Metropolis–Hastings step after (or in place of) the resampling step to smooth the resulting distribution (Van der Merwe et al., 2001). A particle filter with UKF importance distribution is also referred to as the *unscented particle filter* (UPF). Similarly, we could call a particle filter with the Gauss–Hermite Kalman filter importance distribution the *Gauss–Hermite particle filter* (GHPF) and one with the cubature Kalman filter importance distribution the *cubature particle filter* (CPF). However, the use of this kind of importance distribution can also lead to the failure of particle filter convergence and hence they should be used with extreme care. Instead of forming the importance distribution directly by using the Gaussian approximation provided by the EKF, UKF, or other Gaussian filter it may be advisable to artificially increase the covariance of the distribution or to replace the Gaussian distribution with a Student's t distribution with a suitable number of degrees of freedom (see Cappé et al., 2005).

By tuning the resampling algorithm to specific estimation problems and possibly changing the order of weight computation and sampling, accuracy and computational efficiency of the algorithm can be improved (Fearnhead and Clifford, 2003). An important issue is that sampling is more efficient without replacement, such that duplicate samples are not stored. There is also evidence that in some situations it is more efficient to use a simple deterministic algorithm for preserving the N most likely particles. In the article by Punskaya et al. (2002) it is shown that in digital demodulation,

where the sampled space is discrete and the optimization criterion is the minimum error, the deterministic algorithm performs better.

The bootstrap filter (Gordon et al., 1993) is a variation of SIR where the dynamic model $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ is used as the importance distribution. This makes the implementation of the algorithm very easy, but due to the inefficiency of the importance distribution it may require a very large number of Monte Carlo samples for accurate estimation results. In the bootstrap filter the resampling is normally done at each time step.

Algorithm 7.5 (Bootstrap filter) *The bootstrap filter algorithm is as follows.*

1 *Draw a new point* $\mathbf{x}_k^{(i)}$ *for each point in the sample set* $\{\mathbf{x}_{k-1}^{(i)} : i = 1, \ldots, N\}$ *from the dynamic model:*

$$\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}), \qquad i = 1, \ldots, N. \tag{7.34}$$

2 *Calculate the weights*

$$w_k^{(i)} \propto p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}), \qquad i = 1, \ldots, N, \tag{7.35}$$

and normalize them to sum to unity.

3 *Do resampling.*

Another variation of sequential importance resampling is the auxiliary SIR (ASIR) filter (Pitt and Shephard, 1999). The idea of the ASIR is to mimic the availability of the optimal importance distribution by performing the resampling at step $k-1$ using the available measurement at time k.

One problem encountered in particle filtering, even when using a resampling procedure, is called *sample impoverishment* (see, e.g., Ristic et al., 2004). It refers to the effect that when the noise in the dynamic model is very small, many of the particles in the particle set will turn out to have exactly the same value. That is, the resampling step simply multiplies a few (or one) particles and thus we end up having a set of identical copies of certain high weighted particles. This problem can be diminished by using, for example, the resample-move algorithm, regularization, or MCMC steps (Ristic et al., 2004).

Because low noise in the dynamic model causes sample impoverishment, it also implies that pure recursive estimation with particle filters is challenging. This is because in pure recursive estimation the process noise is formally zero and thus a basic SIR based particle filter is likely to perform very badly. Pure recursive estimation, such as recursive estimation of static parameters, can sometimes be done by applying a Rao–Blackwellized particle filter instead of the basic SIR particle filter (see

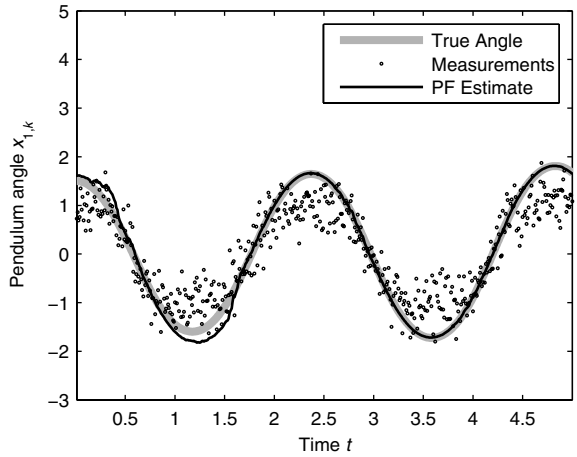


Figure 7.3 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with a bootstrap type of particle filter (PF) with 10 000 particles. The resulting RMSE was 0.12 which is slightly higher than the errors of most of the Gaussian approximation based filters (the EKF had 0.12, others 0.11).

Section 12.3.5). However, the more common use of Rao–Blackwellization is in the conditionally linear Gaussian state space models which we will discuss in the next section.

Example 7.1 (Pendulum tracking with a particle filter) *The result of the bootstrap filter with 10 000 particles in the pendulum model (Example 5.1) is shown in Figure 7.3. The RMSE of* 0.12 *is slightly higher than with most of the other filters – with the EKF we got an RMSE of* 0.12 *and with the SLF/UKF/GHKF/CKF we got an RMSE of* 0.11. *This implies that in this case the filtering distribution is indeed quite well approximated with a Gaussian distribution, and thus using a particle filter is not beneficial.*

In the above example the model is of the type which is suitable for Gaussian approximation based filters and thus the particle filter produces much the same result as they do. But often the noises in the system are not Gaussian or there might be clutter (outlier) measurements which do not fit into the Gaussian non-linear state space modeling framework at all. In these kinds of model, the particle filter still produces good results whereas

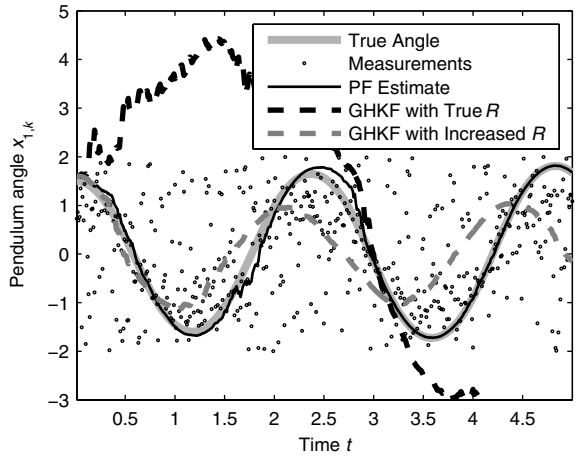


Figure 7.4 Simulated pendulum data in the presence of 50% clutter (outlier) measurements and the result of tracking with a bootstrap type of particle filter (PF) with 10 000 particles (see Example 7.2). The resulting RMSE was 0.16. The corresponding errors of the GHKF with no clutter model at all and the GHKF with artificially increased noise variance were 3.60 and 0.82, respectively.

Gaussian filters do not work at all. The next example illustrates this kind of situation.

Example 7.2 (Cluttered pendulum tracking with a particle filter) *In this scenario, the pendulum sensor is broken such that at each time instant it produces clutter (a random number in the range* $[-2, 2]$*) with probability* 50%. *This kind of situation can be modeled by including an indicator value (data association indicator) as part of the state, which indicates whether the measurement is clutter or not. The result of the bootstrap filter in the pendulum model (Example 5.1) is shown in Figure 7.4. The RMSE of* 0.16 *is slightly higher than in the clutter-free case. In Gaussian filters a clutter model cannot be included into the system as such, but one heuristic way to cope with it is to set the measurement noise variance high enough. In Figure 7.4 we also show the results of a Gauss–Hermite Kalman filter (GHKF) with no clutter model at all, and the result of a GHKF with artificially increased noise variance. The RMSEs of these filters are* 3.60 *and* 0.82*, respectively, which are much higher than the RMSE of the particle*

filter. The estimate trajectories of the GHKFs also indicate that they are having significant trouble in estimating the state.

7.5 Rao–Blackwellized particle filter

One way of improving the efficiency of SIR is to use Rao–Blackwellization. The idea of the *Rao–Blackwellized particle filter* (RBPF) (Akashi and Kumamoto, 1977; Doucet et al., 2001; Ristic et al., 2004), which is also called the *mixture Kalman filter* (MKF) (Chen and Liu, 2000) is that sometimes it is possible to evaluate some of the filtering equations analytically and the others with Monte Carlo sampling instead of computing everything with pure sampling. According to the *Rao–Blackwell theorem* (see, e.g., Berger, 1985) this leads to estimators with less variance than could be obtained with pure Monte Carlo sampling. An intuitive way of understanding this is that the marginalization replaces the finite Monte Carlo particle set representation with an infinite closed form particle set, which is always more accurate than any finite set.

Most commonly Rao–Blackwellized particle filtering refers to marginalized filtering of conditionally linear Gaussian models of the form

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_{k-1}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{A}_{k-1}(\mathbf{u}_{k-1})\,\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}(\mathbf{u}_{k-1})), \\ p(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{u}_k) &= \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}_k(\mathbf{u}_k)\,\mathbf{x}_k, \mathbf{R}_k(\mathbf{u}_k)), \\ p(\mathbf{u}_k \mid \mathbf{u}_{k-1}) &= (\text{any given form}), \end{aligned} \tag{7.36}$$

where $\mathbf{x}_k$ is the state, $\mathbf{y}_k$ is the measurement, and $\mathbf{u}_k$ is an arbitrary latent variable. If also the prior of $\mathbf{x}_k$ is Gaussian, then due to the conditionally linear Gaussian structure of the model the state variables $\mathbf{x}_k$ can be integrated out analytically and only the latent variables $\mathbf{u}_k$ need to be sampled. The Rao–Blackwellized particle filter uses SIR for the latent variables and computes the conditionally Gaussian part in closed form.

To derive the filtering algorithm, first note that the full posterior distribution at step k can be factored as

$$p(\mathbf{u}_{0:k}, \mathbf{x}_{0:k} \mid \mathbf{y}_{1:k}) = p(\mathbf{x}_{0:k} \mid \mathbf{u}_{0:k}, \mathbf{y}_{1:k})\, p(\mathbf{u}_{0:k} \mid \mathbf{y}_{1:k}), \tag{7.37}$$

where the first term is Gaussian and computable with the Kalman filter and RTS smoother. For the second term we get the following recursion

analogously to Equation (7.17):

$$\begin{aligned} & p(\mathbf{u}_{0:k} \mid \mathbf{y}_{1:k}) \\ & \propto p(\mathbf{y}_k \mid \mathbf{u}_{0:k}, \mathbf{y}_{1:k-1})\, p(\mathbf{u}_{0:k} \mid \mathbf{y}_{1:k-1}) \\ & = p(\mathbf{y}_k \mid \mathbf{u}_{0:k}, \mathbf{y}_{1:k-1})\, p(\mathbf{u}_k \mid \mathbf{u}_{0:k-1}, \mathbf{y}_{1:k-1})\, p(\mathbf{u}_{0:k-1} \mid \mathbf{y}_{1:k-1}) \\ & = p(\mathbf{y}_k \mid \mathbf{u}_{0:k}, \mathbf{y}_{1:k-1})\, p(\mathbf{u}_k \mid \mathbf{u}_{k-1})\, p(\mathbf{u}_{0:k-1} \mid \mathbf{y}_{1:k-1}), \end{aligned} \tag{7.38}$$

where we have used the Markovianity of $\mathbf{u}_k$. Now the measurements are not conditionally independent given $\mathbf{u}_k$ and thus the first term differs from the corresponding term in Equation (7.17). The first term can be computed by running the Kalman filter with fixed $\mathbf{u}_{0:k}$ over the measurement sequence. The second term is just the dynamic model and the third term is the posterior from the previous step.

If we form the importance distribution recursively as follows:

$$\pi(\mathbf{u}_{0:k} \mid \mathbf{y}_{1:k}) = \pi(\mathbf{u}_k \mid \mathbf{u}_{0:k-1}, \mathbf{y}_{1:k})\, \pi(\mathbf{u}_{0:k-1} \mid \mathbf{y}_{1:k-1}), \tag{7.39}$$

then by following the same derivation as in Section 7.3, we get the following recursion for the weights:

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k \mid \mathbf{u}_{0:k-1}^{(i)}, \mathbf{y}_{1:k-1})\, p(\mathbf{u}_k^{(i)} \mid \mathbf{u}_{k-1}^{(i)})}{\pi(\mathbf{u}_k^{(i)} \mid \mathbf{u}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})}\, w_{k-1}^{(i)}, \tag{7.40}$$

which corresponds to Equation (7.22). Thus via the above recursion we can form an importance sampling based approximation to the marginal distribution $p(\mathbf{u}_{0:k} \mid \mathbf{y}_{1:k})$. But because, given $\mathbf{u}_{0:k}$, the distribution $p(\mathbf{x}_{0:k} \mid \mathbf{u}_{0:k}, \mathbf{y}_{1:k})$ is Gaussian, we can form the full posterior distribution by using Equation (7.37). Computing the distribution jointly for the full history $\mathbf{x}_{0:k}$ would require running both the Kalman filter and the RTS smoother over the sequences $\mathbf{u}_{0:k}$ and $\mathbf{y}_{1:k}$, but if we are only interested in the posterior of the last time step $\mathbf{x}_k$, we only need to run the Kalman filter. The resulting algorithm is the following.

Algorithm 7.6 (Rao–Blackwellized particle filter) *Given a sequence of importance distributions* $\pi(\mathbf{u}_k \mid \mathbf{u}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})$ *and a set of weighted samples* $\{w_{k-1}^{(i)}, \mathbf{u}_{k-1}^{(i)}, \mathbf{m}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)} : i = 1, \ldots, N\}$*, the Rao–Blackwellized particle filter (RBPF) processes the measurement* $\mathbf{y}_k$ *as follows (Doucet et al., 2001).*

1 *Perform Kalman filter predictions for each of the Kalman filter means and covariances in the particles* $i = 1, \ldots, N$ *conditional on the previ-*

ously drawn latent variable values $\mathbf{u}_{k-1}^{(i)}$*:*

$$\begin{aligned}\mathbf{m}_k^{-(i)} &= \mathbf{A}_{k-1}(\mathbf{u}_{k-1}^{(i)})\,\mathbf{m}_{k-1}^{(i)},\\ \mathbf{P}_k^{-(i)} &= \mathbf{A}_{k-1}(\mathbf{u}_{k-1}^{(i)})\,\mathbf{P}_{k-1}^{(i)}\,\mathbf{A}_{k-1}^{\mathsf{T}}(\mathbf{u}_{k-1}^{(i)}) + \mathbf{Q}_{k-1}(\mathbf{u}_{k-1}^{(i)}).\end{aligned} \tag{7.41}$$

2 *Draw new latent variables* $\mathbf{u}_k^{(i)}$ *for each particle in* $i = 1, \ldots, N$ *from the corresponding importance distributions:*

$$\mathbf{u}_k^{(i)} \sim \pi(\mathbf{u}_k \mid \mathbf{u}_{0:k-1}^{(i)}, \mathbf{y}_{1:k}). \tag{7.42}$$

3 *Calculate new weights as follows:*

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k \mid \mathbf{u}_{0:k}^{(i)}, \mathbf{y}_{1:k-1})\; p(\mathbf{u}_k^{(i)} \mid \mathbf{u}_{k-1}^{(i)})}{\pi(\mathbf{u}_k^{(i)} \mid \mathbf{u}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})}, \tag{7.43}$$

where the likelihood term is the marginal measurement likelihood of the Kalman filter

$$\begin{aligned}&p(\mathbf{y}_k \mid \mathbf{u}_{0:k}^{(i)}, \mathbf{y}_{1:k-1})\\ &= \mathrm{N}\left(\mathbf{y}_k \,\middle|\, \mathbf{H}_k(\mathbf{u}_k^{(i)})\,\mathbf{m}_k^{-(i)}, \mathbf{H}_k(\mathbf{u}_k^{(i)})\,\mathbf{P}_k^{-(i)}\,\mathbf{H}_k^{\mathsf{T}}(\mathbf{u}_k^{(i)}) + \mathbf{R}_k(\mathbf{u}_k^{(i)})\right)\end{aligned} \tag{7.44}$$

such that the model parameters in the Kalman filter are conditioned on the drawn latent variable value $\mathbf{u}_k^{(i)}$*. Then normalize the weights to sum to unity.*

4 *Perform Kalman filter updates for each of the particles conditional on the drawn latent variables* $\mathbf{u}_k^{(i)}$*:*

$$\begin{aligned}\mathbf{v}_k^{(i)} &= \mathbf{y}_k - \mathbf{H}_k(\mathbf{u}_k^{(i)})\,\mathbf{m}_k^{-},\\ \mathbf{S}_k^{(i)} &= \mathbf{H}_k(\mathbf{u}_k^{(i)})\,\mathbf{P}_k^{-(i)}\,\mathbf{H}_k^{\mathsf{T}}(\mathbf{u}_k^{(i)}) + \mathbf{R}_k(\mathbf{u}_k^{(i)}),\\ \mathbf{K}_k^{(i)} &= \mathbf{P}_k^{-(i)}\,\mathbf{H}_k^{\mathsf{T}}(\mathbf{u}_k^{(i)})\,\mathbf{S}_k^{-1},\\ \mathbf{m}_k^{(i)} &= \mathbf{m}_k^{-(i)} + \mathbf{K}_k^{(i)}\,\mathbf{v}_k^{(i)},\\ \mathbf{P}_k^{(i)} &= \mathbf{P}_k^{-(i)} - \mathbf{K}_k^{(i)}\,\mathbf{S}_k^{(i)}\,[\mathbf{K}_k^{(i)}]^{\mathsf{T}}.\end{aligned} \tag{7.45}$$

5 *If the effective number of particles* (7.27) *is too low, perform* resampling.

The Rao–Blackwellized particle filter produces for each time step k a set of weighted samples $\{w_k^{(i)}, \mathbf{u}_k^{(i)}, \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)} \;:\; i = 1, \ldots, N\}$ such that

the expectation of a function $\mathbf{g}(\cdot)$ can be approximated as

$$\mathrm{E}[\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) \mid \mathbf{y}_{1:k}] \approx \sum_{i=1}^{N} w_k^{(i)} \int \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k^{(i)}) \, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)}) \, \mathrm{d}\mathbf{x}_k. \tag{7.46}$$

Equivalently, the RBPF can be interpreted to form an approximation to the filtering distribution as

$$p(\mathbf{x}_k, \mathbf{u}_k \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \, \delta(\mathbf{u}_k - \mathbf{u}_k^{(i)}) \, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)}). \tag{7.47}$$

The optimal importance distribution, that is, the importance distribution that minimizes the variance of the importance weights in the RBPF case is

$$\begin{aligned} \pi(\mathbf{u}_k \mid \mathbf{y}_{1:k}, \mathbf{u}_{0:k-1}^{(i)}) &= p(\mathbf{u}_k \mid \mathbf{y}_{1:k}, \mathbf{u}_{0:k-1}^{(i)}) \\ &\propto p(\mathbf{y}_k \mid \mathbf{u}_k, \mathbf{u}_{0:k-1}^{(i)}) \, p(\mathbf{u}_k \mid \mathbf{u}_{0:k-1}^{(i)}, \mathbf{y}_{1:k-1}). \end{aligned} \tag{7.48}$$

In general, normalizing this distribution or drawing samples from this distribution directly is not possible. But, if the latent variables $\mathbf{u}_k$ are discrete, we can normalize this distribution and use this optimal importance distribution directly.

The class of models where Rao–Blackwellization of some linear state components can be carried out can be extended beyond the conditionally linear Gaussian models presented here. We can, for example, include additional latent-variable dependent non-linear terms into the dynamic and measurement models (Schön et al., 2005). In some cases, when the filtering model is not strictly Gaussian due to slight non-linearities in either the dynamic or measurement models, it is possible to replace the exact Kalman filter update and prediction steps in RBPF with the extended Kalman filter (EKF), the unscented Kalman filter (UKF) prediction and update steps, or with any other Gaussian approximation based filters (Särkkä et al., 2007b).

7.6 Exercises

7.1 Recall the following state space model from Exercise 3.1:

$$\begin{aligned} \mathbf{x}_k &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ y_k &= \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{x}_k + r_k, \end{aligned} \tag{7.49}$$

where $\mathbf{x}_k = (x_k \;\; \dot{x}_k)^\mathsf{T}$ is the state, y_k is the measurement, and $\mathbf{q}_k \sim \mathrm{N}(\mathbf{0}, \mathrm{diag}(1/10^2, 1^2))$ and $r_k \sim \mathrm{N}(0, 10^2)$ are white Gaussian noise processes.

(a) Write down the Kalman filter equations for this model.
(b) Derive an expression for the optimal importance distribution for the model:

$$\pi(\mathbf{x}_k) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_{1:k}). \tag{7.50}$$

(c) Write pseudo-code for the corresponding particle filter algorithm (sequential importance resampling algorithm). Also write down the equations for the weight update.
(d) Compare the number of CPU steps (multiplications and additions) needed by the particle filter and Kalman filter. Which implementation would you choose for a real implementation?

7.2 Implement the bootstrap filter for the model in Exercise 5.1 and test its performance against the non-linear Kalman filters.

7.3 Implement a sequential importance resampling filter with an EKF, UKF, GHKF, or CKF based importance distribution for the model in Exercise 5.1. Note that you might want to use a small non-zero covariance as the prior of the previous step instead of plain zero to get the filters to work better.

7.4 Implement the bootstrap filter and SIR with the CKF importance distribution for the bearings only target tracking model in Exercise 5.5. Plot the results and compare the RMSE values to those of the non-linear Kalman filters.

7.5 Implement a Rao–Blackwellized particle filter for the following clutter model (outlier model) :

$$\begin{aligned} x_k &= x_{k-1} + q_{k-1}, \\ y_k &= \begin{cases} x_k + r_k, & \text{if } u_k = 0, \\ r_k^{\mathrm{c}}, & \text{if } u_k = 1, \end{cases} \end{aligned} \tag{7.51}$$

where $q_{k-1} \sim \mathrm{N}(0, 1)$, $r_k \sim \mathrm{N}(0, 1)$ and $r_k^{\mathrm{c}} \sim \mathrm{N}(0, 10^2)$. The indicator variables u_k are modeled as independent random variables which take the value $u_k = 0$ with prior probability 0.9 and the value $u_k = 1$ with prior probability 0.1. Test the performance of the filter with simulated data and compare the performance to a Kalman filter, where the clutter r_k^{c} is ignored. What is the optimal importance distribution for this model?

8

Bayesian smoothing equations and exact solutions

So far in this book we have only considered filtering algorithms which use the measurements obtained before and at the current step for computing the best possible estimate of the current state (and possibly future states). However, sometimes it is also of interest to estimate states for each time step conditional on all the measurements that we have obtained. This problem can be solved with Bayesian smoothing. In this chapter, we present the Bayesian theory of smoothing. After that we derive the Rauch–Tung–Striebel (RTS) smoother which is the closed form smoothing solution to linear Gaussian models. We also briefly discuss two-filter smoothers.

8.1 Bayesian smoothing equations

The purpose of *Bayesian smoothing*[1] is to compute the marginal posterior distribution of the state $\mathbf{x}_k$ at the time step k after receiving the measurements up to a time step T, where $T > k$:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}). \tag{8.1}$$

The difference between filters and smoothers is that *the Bayesian filter* computes its estimates using only the measurements obtained before and at the time step k, but *the Bayesian smoother* uses also the future measurements for computing its estimates. After obtaining the filtering posterior state distributions, the following theorem gives the equations for computing the marginal posterior distributions for each time step conditionally on all the measurements up to the time step T.

Theorem 8.1 (Bayesian optimal smoothing equations) *The backward recursive equations (the* Bayesian smoother*) for computing the* smoothed distributions $p(\mathbf{x}_k \mid \mathbf{y}_{1:T})$ *for any* $k < T$ *are given by the following*

[1] This definition actually applies to the fixed-interval type of smoothing.

Bayesian (fixed-interval) smoothing equations *(Kitagawa, 1987):*

$$p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) = \int p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k})\, \mathrm{d}\mathbf{x}_k,$$
$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) = p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \int \left[\frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \right] \mathrm{d}\mathbf{x}_{k+1}, \tag{8.2}$$

where $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ is the filtering distribution of the time step k. Note that the term $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})$ is simply the predicted distribution of time step $k+1$. The integrations are replaced by summations if some of the state components are discrete.

Proof Due to the Markov properties the state $\mathbf{x}_k$ is independent of $\mathbf{y}_{k+1:T}$ given $\mathbf{x}_{k+1}$, which gives $p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) = p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k})$. By using *Bayes' rule* the distribution of $\mathbf{x}_k$ given $\mathbf{x}_{k+1}$ and $\mathbf{y}_{1:T}$ can be expressed as

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}) \\ &= \frac{p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \\ &= \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{y}_{1:k})\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \\ &= \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})}. \end{aligned} \tag{8.3}$$

The joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ given $\mathbf{y}_{1:T}$ can be now computed as

$$\begin{aligned} p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T})\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \\ &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k})\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \\ &= \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k})\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})}, \end{aligned} \tag{8.4}$$

where $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})$ is the smoothed distribution of the time step $k+1$. The marginal distribution of $\mathbf{x}_k$ given $\mathbf{y}_{1:T}$ is given by integration (or summation) over $\mathbf{x}_{k+1}$ in Equation (8.4), which gives the desired result. □

8.2 Rauch–Tung–Striebel smoother

The *Rauch–Tung–Striebel smoother* (RTSS, Rauch et al., 1965), which is also called the *Kalman smoother*, can be used for computing the closed

form smoothing solution

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^{\mathrm{s}}, \mathbf{P}_k^{\mathrm{s}}) \tag{8.5}$$

to the linear filtering model (4.17). The difference from the solution computed by the *Kalman filter* is that the smoothed solution is conditional on the whole measurement data $\mathbf{y}_{1:T}$, while the filtering solution is conditional only on the measurements obtained before and at the time step k, that is, on the measurements $\mathbf{y}_{1:k}$.

Theorem 8.2 (RTS smoother) *The backward recursion equations for the (fixed interval) Rauch–Tung–Striebel smoother are given as*

$$\begin{aligned}
\mathbf{m}_{k+1}^{-} &= \mathbf{A}_k \, \mathbf{m}_k, \\
\mathbf{P}_{k+1}^{-} &= \mathbf{A}_k \, \mathbf{P}_k \, \mathbf{A}_k^{\mathsf{T}} + \mathbf{Q}_k, \\
\mathbf{G}_k &= \mathbf{P}_k \, \mathbf{A}_k^{\mathsf{T}} \, [\mathbf{P}_{k+1}^{-}]^{-1}, \\
\mathbf{m}_k^{\mathrm{s}} &= \mathbf{m}_k + \mathbf{G}_k \, [\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^{-}], \\
\mathbf{P}_k^{\mathrm{s}} &= \mathbf{P}_k + \mathbf{G}_k \, [\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^{-}] \, \mathbf{G}_k^{\mathsf{T}},
\end{aligned} \tag{8.6}$$

where $\mathbf{m}_k$ and $\mathbf{P}_k$ are the mean and covariance computed by the Kalman filter. The recursion is started from the last time step T, with $\mathbf{m}_T^{\mathrm{s}} = \mathbf{m}_T$ and $\mathbf{P}_T^{\mathrm{s}} = \mathbf{P}_T$. Note that the first two of the equations are simply the Kalman filter prediction equations.

Proof Similarly to the Kalman filter case, by Lemma A.1, the joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ given $\mathbf{y}_{1:k}$ is

$$\begin{aligned}
p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) &= p(\mathbf{x}_{k+1} \mid \mathbf{x}_k) \, p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \\
&= \mathrm{N}(\mathbf{x}_{k+1} \mid \mathbf{A}_k \, \mathbf{x}_k, \mathbf{Q}_k) \; \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k) \\
&= \mathrm{N}\left(\begin{pmatrix} \mathbf{x}_k \\ \mathbf{x}_{k+1} \end{pmatrix} \,\middle|\, \tilde{\mathbf{m}}_1, \tilde{\mathbf{P}}_1 \right),
\end{aligned} \tag{8.7}$$

where

$$\tilde{\mathbf{m}}_1 = \begin{pmatrix} \mathbf{m}_k \\ \mathbf{A}_k \, \mathbf{m}_k \end{pmatrix}, \qquad \tilde{\mathbf{P}}_1 = \begin{pmatrix} \mathbf{P}_k & \mathbf{P}_k \, \mathbf{A}_k^{\mathsf{T}} \\ \mathbf{A}_k \, \mathbf{P}_k & \mathbf{A}_k \, \mathbf{P}_k \, \mathbf{A}_k^{\mathsf{T}} + \mathbf{Q}_k \end{pmatrix}. \tag{8.8}$$

Due to the Markov property of the states we have

$$p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) = p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}), \tag{8.9}$$

and thus by Lemma A.2 we get the conditional distribution

$$\begin{aligned}
p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}) \\
&= \mathrm{N}(\mathbf{x}_k \mid \tilde{\mathbf{m}}_2, \tilde{\mathbf{P}}_2),
\end{aligned} \tag{8.10}$$

where

$$\begin{aligned}\mathbf{G}_k &= \mathbf{P}_k \, \mathbf{A}_k^\mathsf{T} \, (\mathbf{A}_k \, \mathbf{P}_k \, \mathbf{A}_k^\mathsf{T} + \mathbf{Q}_k)^{-1} \\ \tilde{\mathbf{m}}_2 &= \mathbf{m}_k + \mathbf{G}_k \, (\mathbf{x}_{k+1} - \mathbf{A}_k \, \mathbf{m}_k) \\ \tilde{\mathbf{P}}_2 &= \mathbf{P}_k - \mathbf{G}_k \, (\mathbf{A}_k \, \mathbf{P}_k \, \mathbf{A}_k^\mathsf{T} + \mathbf{Q}_k) \, \mathbf{G}_k^\mathsf{T}. \end{aligned} \tag{8.11}$$

The joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ given all the data is

$$\begin{aligned} p(\mathbf{x}_{k+1}, \mathbf{x}_k \mid \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) \, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \\ &= \mathrm{N}(\mathbf{x}_k \mid \tilde{\mathbf{m}}_2, \tilde{\mathbf{P}}_2) \, \mathrm{N}(\mathbf{x}_{k+1} \mid \mathbf{m}_{k+1}^\mathrm{s}, \mathbf{P}_{k+1}^\mathrm{s}) \\ &= \mathrm{N}\left(\begin{pmatrix} \mathbf{x}_{k+1} \\ \mathbf{x}_k \end{pmatrix} \,\middle|\, \tilde{\mathbf{m}}_3, \tilde{\mathbf{P}}_3 \right), \end{aligned} \tag{8.12}$$

where

$$\begin{aligned} \tilde{\mathbf{m}}_3 &= \begin{pmatrix} \mathbf{m}_{k+1}^\mathrm{s} \\ \mathbf{m}_k + \mathbf{G}_k \, (\mathbf{m}_{k+1}^\mathrm{s} - \mathbf{A}_k \, \mathbf{m}_k) \end{pmatrix}, \\ \tilde{\mathbf{P}}_3 &= \begin{pmatrix} \mathbf{P}_{k+1}^\mathrm{s} & \mathbf{P}_{k+1}^\mathrm{s} \, \mathbf{G}_k^\mathsf{T} \\ \mathbf{G}_k \, \mathbf{P}_{k+1}^\mathrm{s} & \mathbf{G}_k \, \mathbf{P}_{k+1}^\mathrm{s} \, \mathbf{G}_k^\mathsf{T} + \tilde{\mathbf{P}}_2 \end{pmatrix}. \end{aligned} \tag{8.13}$$

Thus by Lemma A.2, the marginal distribution of $\mathbf{x}_k$ is given as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^\mathrm{s}, \mathbf{P}_k^\mathrm{s}), \tag{8.14}$$

where

$$\begin{aligned} \mathbf{m}_k^\mathrm{s} &= \mathbf{m}_k + \mathbf{G}_k \, (\mathbf{m}_{k+1}^\mathrm{s} - \mathbf{A}_k \, \mathbf{m}_k), \\ \mathbf{P}_k^\mathrm{s} &= \mathbf{P}_k + \mathbf{G}_k \, (\mathbf{P}_{k+1}^\mathrm{s} - \mathbf{A}_k \, \mathbf{P}_k \, \mathbf{A}_k^\mathsf{T} - \mathbf{Q}_k) \, \mathbf{G}_k^\mathsf{T}. \end{aligned} \tag{8.15}$$

□

Example 8.1 (RTS smoother for Gaussian random walk) *The RTS smoother for the random walk model given in Example 4.1 is given by the equations*

$$\begin{aligned} m_{k+1}^- &= m_k, \\ P_{k+1}^- &= P_k + Q, \\ m_k^\mathrm{s} &= m_k + \frac{P_k}{P_{k+1}^-}(m_{k+1}^\mathrm{s} - m_{k+1}^-), \\ P_k^\mathrm{s} &= P_k + \left(\frac{P_k}{P_{k+1}^-} \right)^2 [P_{k+1}^\mathrm{s} - P_{k+1}^-], \end{aligned} \tag{8.16}$$

where m_k and P_k are the updated mean and covariance from the Kalman filter in Example 4.2. The result of applying the smoother to simulated data

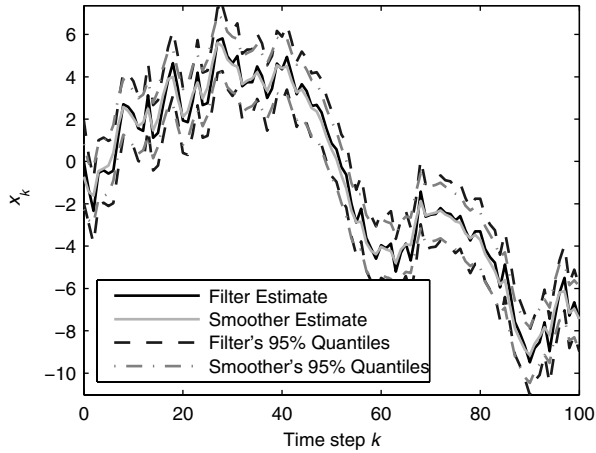


Figure 8.1 Filter and smoother estimates in the Gaussian random walk smoothing example (Example 8.1).

is shown in Figure 8.1. The evolution of the filter and smoother variances is illustrated in Figure 8.2.

Example 8.2 (RTS smoother for car tracking) *The backward recursion equations required for implementing the RTS smoother for the car tracking problem in Example 4.3 are the following:*

$$\mathbf{m}_{k+1}^- = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{m}_k,$$

$$\mathbf{P}_{k+1}^- = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{P}_k \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^\mathsf{T} + \begin{pmatrix} \frac{q_1^c \Delta t^3}{3} & 0 & \frac{q_1^c \Delta t^2}{2} & 0 \\ 0 & \frac{q_2^c \Delta t^3}{3} & 0 & \frac{q_2^c \Delta t^2}{2} \\ \frac{q_1^c \Delta t^2}{2} & 0 & q_1^c \Delta t & 0 \\ 0 & \frac{q_2^c \Delta t^2}{2} & 0 & q_2^c \Delta t \end{pmatrix},$$

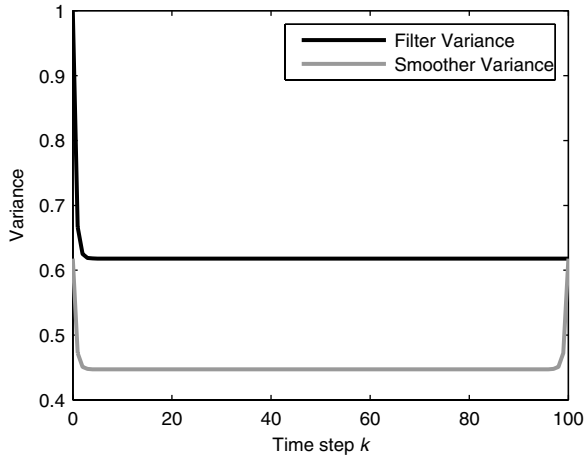


Figure 8.2 Filter and smoother variances in the Gaussian random walk smoothing example (Example 8.1). The variance of the smoother is always smaller than that of the filter. The only exception is at the final step, where the variances are the same.

$$\mathbf{G}_k = \mathbf{P}_k \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{\mathsf{T}} [\mathbf{P}_{k+1}^{-}]^{-1},$$

$$\mathbf{m}_k^{\mathrm{s}} = \mathbf{m}_k + \mathbf{G}_k \left[\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^{-}\right],$$

$$\mathbf{P}_k^{\mathrm{s}} = \mathbf{P}_k + \mathbf{G}_k \left[\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^{-}\right] \mathbf{G}_k^{\mathsf{T}}.$$

The terms $\mathbf{m}_k$ and $\mathbf{P}_k$ are the Kalman filter means and covariance computed with the equations given in Example 4.3. It would also be possible to store the values $\mathbf{m}_{k+1}^{-}$ and $\mathbf{P}_{k+1}^{-}$ during Kalman filtering to avoid recomputation of them in the first two equations above. The gains $\mathbf{G}_k$ could be computed already during the Kalman filtering as well. The result of applying the RTS smoother to simulated data is shown in Figure 8.3.

8.3 Two-filter smoothing

An alternative approach to the RTS-style forward–backward smoothing is the two-filter smoothing approach. Although the approach has many advantages, unfortunately in the non-linear case it requires construction

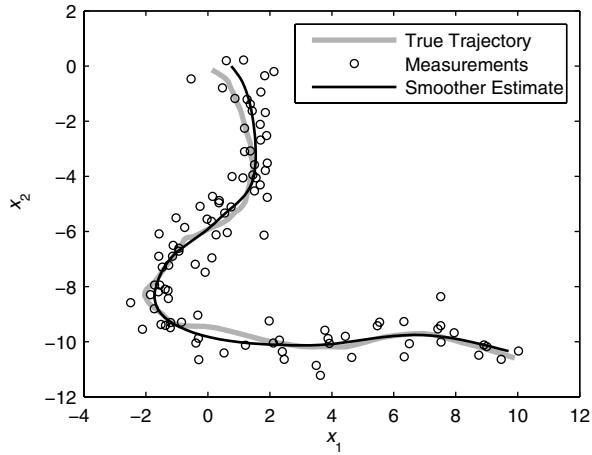


Figure 8.3 Simulated trajectory, measurements and the result of RTS smoother based car tracking in Example 4.3. The starting point is at the top of the trajectory. The RMSE position error based on the measurements only is 0.77, the position RMSE of the Kalman filter estimate is 0.43 and the error of the RTS smoother is 0.27. It can be seen that the estimate is much "smoother" than the result of the Kalman filter in Figure 4.5.

of artificial probability densities which ensure that the backward filter is normalizable. Due to this challenge, in this chapter we base the smoothing algorithms on the forward–backward smoothing approach and only briefly outline the idea of two-filter smoothing here.

The two-filter smoother formulations of Fraser and Potter (1969) and Kitagawa (1994) are based on the following representation of the smoothing distribution:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:n}) \propto p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})\, p(\mathbf{y}_{k:n} \mid \mathbf{x}_k). \tag{8.17}$$

The first term on the right is just the result of the Bayesian filter just after prediction on the step $k-1$. The second term on the right can be evaluated

by using the following backward recursions:

$$\begin{aligned} p(\mathbf{y}_{k+1:n} \mid \mathbf{x}_k) &= \int p(\mathbf{y}_{k+1:n}, \mathbf{x}_{k+1} \mid \mathbf{x}_k)\, \mathrm{d}\mathbf{x}_{k+1} \\ &= \int p(\mathbf{y}_{k+1:n} \mid \mathbf{x}_{k+1}, \mathbf{x}_k)\, p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, \mathrm{d}\mathbf{x}_{k+1} \\ &= \int p(\mathbf{y}_{k+1:n} \mid \mathbf{x}_{k+1})\, p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, \mathrm{d}\mathbf{x}_{k+1}, \end{aligned} \tag{8.18}$$

$$\begin{aligned} p(\mathbf{y}_{k:n} \mid \mathbf{x}_k) &= p(\mathbf{y}_{k+1:n}, \mathbf{y}_k \mid \mathbf{x}_k) \\ &= p(\mathbf{y}_{k+1:n} \mid \mathbf{x}_k)\, p(\mathbf{y}_k \mid \mathbf{y}_{k+1:n}, \mathbf{x}_k) \\ &= p(\mathbf{y}_{k+1:n} \mid \mathbf{x}_k)\, p(\mathbf{y}_k \mid \mathbf{x}_k). \end{aligned} \tag{8.19}$$

The classical linear two-filter smoother (Fraser and Potter, 1969) can be derived from the general equations by assuming that we have

$$p(\mathbf{y}_{k:n} \mid \mathbf{x}_k) \propto \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^{\mathrm{b}}, \mathbf{P}_k^{\mathrm{b}}), \tag{8.20}$$

for some mean $\mathbf{m}_k^{\mathrm{b}}$ and covariance $\mathbf{P}_k^{\mathrm{b}}$. This results in recursions which resemble a Kalman filter which runs backwards in time. However, it turns out that at the initial steps of the backward recursions the distributions are not normalizable, because the covariances $\mathbf{P}_k^{\mathrm{b}}$ are formally singular. In the formulation of Fraser and Potter (1969) this problem is avoided by using the so-called information filter, which is a formulation of the Kalman filter in terms of inverses of covariance matrices instead of the plain covariances.

Unfortunately, when starting from Equations (8.18) and (8.19), it is difficult to go beyond the linear case because, in the more general case, a simple information filter formulation does not work. For example, there is no Monte Carlo version of an information filter. It is indeed possible to form reasonable approximations by forming backward dynamic models by using, for example, the unscented transform (Wan and Van der Merwe, 2001), but in general this might not lead to good or valid approximations (Briers et al., 2010). The key problem is that $p(\mathbf{y}_{k:n} \mid \mathbf{x}_k)$ is not generally normalizable with respect to $\mathbf{x}_k$, that is, we have $\int p(\mathbf{y}_{k:n} \mid \mathbf{x}_k)\, \mathrm{d}\mathbf{x}_k = \infty$.

One solution to the problem was presented by Briers et al. (2010), who proposed a generalized version of the two-filter smoothing formulas of Kitagawa (1994). The solution is based on the introduction of artificial probability densities $\{\gamma_k(\mathbf{x}_k)\}$ such that if $p(\mathbf{y}_{k:n} \mid \mathbf{x}_k) > 0$ then $\gamma_k(\mathbf{x}_k) > 0$. The backward recursions are then replaced with the following recur-

sions:

$$\tilde{p}(\mathbf{x}_k \mid \mathbf{y}_{k+1:n}) = \int \tilde{p}(\mathbf{x}_{k+1} \mid \mathbf{y}_{k+1:n}) \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, \gamma_k(\mathbf{x}_k)}{\gamma_{k+1}(\mathbf{x}_{k+1})} \, \mathrm{d}\mathbf{x}_{k+1}, \tag{8.21}$$

$$\tilde{p}(\mathbf{x}_k \mid \mathbf{y}_{k:n}) = \frac{\tilde{p}(\mathbf{x}_k \mid \mathbf{y}_{k+1:n})\, p(\mathbf{y}_k \mid \mathbf{x}_k)}{\int \tilde{p}(\mathbf{x}_k \mid \mathbf{y}_{k+1:n})\, p(\mathbf{y}_k \mid \mathbf{x}_k) \, \mathrm{d}\mathbf{x}_k}, \tag{8.22}$$

where $\tilde{p}(\mathbf{x}_k \mid \mathbf{y}_{k:n})$ is an auxiliary probability density such that

$$p(\mathbf{y}_{k:n} \mid \mathbf{x}_k) \propto \frac{\tilde{p}(\mathbf{x}_k \mid \mathbf{y}_{k:n})}{\gamma_k(\mathbf{x}_k)}. \tag{8.23}$$

From this starting point it is possible to derive particle based smoothing algorithms as well as other non-linear two-filter smoothers provided that we can select the artificial probability densities suitably. For details, the reader is referred to Briers et al. (2010).

8.4 Exercises

8.1 Derive the linear RTS smoother for the non-zero-mean noise model in Exercise 4.1.

8.2 Write down the Bayesian smoothing equations for finite-state HMM models described in Exercise 4.2 assuming that the filtering distributions $p(x_k \mid \mathbf{y}_{1:k})$ have already been computed.

8.3 Implement the Gaussian random walk model smoother in Example 8.1 and compare its performance to the corresponding Kalman filter. Plot the evolution of the smoothing distribution.

8.4 The Gaussian random walk model considered in Example 4.1 also defines a joint Gaussian prior distribution $p(x_0, \ldots, x_T)$. The measurement model $p(y_1, \ldots, y_T \mid x_0, \ldots, x_T)$ is Gaussian as well. Construct these distributions and compute the posterior distribution $p(x_0, \ldots, x_T \mid y_1, \ldots, y_T)$. Check numerically that the mean and the diagonal covariance entries of this distribution exactly match the smoother means and variances.

8.5 Form a grid-based approximation to the Gaussian random walk model smoother in the same way as was done for the filtering equations in Exercise 4.4. Verify that the result is practically the same as in the RTS smoother above.

8.6 Write down the smoother equations for the Gaussian random walk model, when the stationary filter is used as the filter. Note that the smoother becomes a stationary backward filter. Compare the performance of this stationary smoother to that of the non-stationary smoother.

8.7 Implement the RTS smoother for the resonator model in Exercise 4.6. Compare its RMSE performance to the filtering and baseline solutions and plot the results.

9

Extended and unscented smoothing

In this chapter we present the linearization, statistical linearization, and unscented transform based RTS smoothers, which are based on analogous approximations to the EKF, SLF, and UKF presented in Chapter 5. These smoothers can be used for forming Gaussian approximations to the Bayesian smoothing solutions for non-linear state space models.

9.1 Extended Rauch–Tung–Striebel smoother

The first order (i.e., linearized) extended Rauch–Tung–Striebel smoother (ERTSS) (Cox, 1964; Sage and Melsa, 1971) can be obtained from the basic RTS smoother equations by replacing the prediction equations with first order approximations. Higher order extended Kalman smoothers are also possible (see, e.g., Cox, 1964; Sage and Melsa, 1971), but only the first order version is presented here. The ERTSS forms (or assumes) a Gaussian approximation to the smoothing distribution as follows:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) \simeq \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^{\mathrm{s}}, \mathbf{P}_k^{\mathrm{s}}). \tag{9.1}$$

For the additive model Equation (5.24) the extended Rauch–Tung–Striebel smoother algorithm is the following.

Algorithm 9.1 (Extended RTS smoother) *The equations for the extended RTS smoother are*

$$\begin{aligned}
\mathbf{m}_{k+1}^{-} &= \mathbf{f}(\mathbf{m}_k), \\
\mathbf{P}_{k+1}^{-} &= \mathbf{F}_{\mathbf{x}}(\mathbf{m}_k)\, \mathbf{P}_k\, \mathbf{F}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}_k) + \mathbf{Q}_k, \\
\mathbf{G}_k &= \mathbf{P}_k\, \mathbf{F}_{\mathbf{x}}^{\mathsf{T}}(\mathbf{m}_k)\, [\mathbf{P}_{k+1}^{-}]^{-1}, \\
\mathbf{m}_k^{\mathrm{s}} &= \mathbf{m}_k + \mathbf{G}_k\, [\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^{-}], \\
\mathbf{P}_k^{\mathrm{s}} &= \mathbf{P}_k + \mathbf{G}_k\, [\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^{-}]\, \mathbf{G}_k^{\mathsf{T}},
\end{aligned} \tag{9.2}$$

where the matrix $\mathbf{F}_{\mathbf{x}}(\mathbf{m}_k)$ *is the Jacobian matrix of* $\mathbf{f}(\mathbf{x})$ *evaluated at* $\mathbf{m}_k$.

The above procedure is a recursion which can be used for computing the smoothing distribution of time step k from the smoothing distribution of time step $k+1$. Because the smoothing distribution and filtering distribution of the last time step T are the same, we have $\mathbf{m}_T^{\mathrm{s}} = \mathbf{m}_T$, $\mathbf{P}_T^{\mathrm{s}} = \mathbf{P}_T$, and thus the recursion can be used for computing the smoothing distributions of all time steps by starting from the last step $k = T$ and proceeding backwards to the initial step $k = 0$.

Derivation Assume that the filtering distributions for the model (5.24) are approximately Gaussian:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \simeq \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k),$$

and we have already computed the means and covariance using the extended Kalman filter or a similar method. Further assume that the smoothing distribution of time step $k+1$ is known and approximately Gaussian

$$p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \simeq \mathrm{N}(\mathbf{x}_{k+1} \mid \mathbf{m}_{k+1}^{\mathrm{s}}, \mathbf{P}_{k+1}^{\mathrm{s}}).$$

As in the derivation of the prediction step of the EKF in Section 5.2, the approximate joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ given $\mathbf{y}_{1:k}$ is

$$p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) = \mathrm{N}\left(\begin{pmatrix}\mathbf{x}_k \\ \mathbf{x}_{k+1}\end{pmatrix} \,\middle|\, \tilde{\mathbf{m}}_1, \tilde{\mathbf{P}}_1\right), \tag{9.3}$$

where

$$\begin{aligned}\tilde{\mathbf{m}}_1 &= \begin{pmatrix}\mathbf{m}_k \\ \mathbf{f}(\mathbf{m}_k)\end{pmatrix}, \\ \tilde{\mathbf{P}}_1 &= \begin{pmatrix}\mathbf{P}_k & \mathbf{P}_k \mathbf{F}_x^{\mathsf{T}} \\ \mathbf{F}_x \mathbf{P}_k & \mathbf{F}_x \mathbf{P}_k \mathbf{F}_x^{\mathsf{T}} + \mathbf{Q}_k\end{pmatrix},\end{aligned} \tag{9.4}$$

and the Jacobian matrix $\mathbf{F}_x$ of $\mathbf{f}(\mathbf{x})$ is evaluated at $\mathbf{x} = \mathbf{m}_k$. By conditioning on $\mathbf{x}_{k+1}$ as in the RTS derivation in Section 8.2 we get

$$\begin{aligned}p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}) \\ &= \mathrm{N}(\mathbf{x}_k \mid \tilde{\mathbf{m}}_2, \tilde{\mathbf{P}}_2),\end{aligned} \tag{9.5}$$

where

$$\begin{aligned}\mathbf{G}_k &= \mathbf{P}_k \mathbf{F}_x^{\mathsf{T}} (\mathbf{F}_x \mathbf{P}_k \mathbf{F}_x^{\mathsf{T}} + \mathbf{Q}_k)^{-1}, \\ \tilde{\mathbf{m}}_2 &= \mathbf{m}_k + \mathbf{G}_k (\mathbf{x}_{k+1} - \mathbf{f}(\mathbf{m}_k)), \\ \tilde{\mathbf{P}}_2 &= \mathbf{P}_k - \mathbf{G}_k (\mathbf{F}_x \mathbf{P}_k \mathbf{F}_x^{\mathsf{T}} + \mathbf{Q}_k) \mathbf{G}_k^{\mathsf{T}}.\end{aligned} \tag{9.6}$$

The joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ given all the data is now

$$\begin{aligned} p(\mathbf{x}_{k+1}, \mathbf{x}_k \mid \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T})\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \\ &= \mathrm{N}\left(\begin{pmatrix} \mathbf{x}_{k+1} \\ \mathbf{x}_k \end{pmatrix} \,\middle|\, \tilde{\mathbf{m}}_3, \tilde{\mathbf{P}}_3 \right), \end{aligned} \tag{9.7}$$

where

$$\begin{aligned} \tilde{\mathbf{m}}_3 &= \begin{pmatrix} \mathbf{m}^{\mathrm{s}}_{k+1} \\ \mathbf{m}_k + \mathbf{G}_k \left(\mathbf{m}^{\mathrm{s}}_{k+1} - \mathbf{f}(\mathbf{m}_k)\right) \end{pmatrix}, \\ \tilde{\mathbf{P}}_3 &= \begin{pmatrix} \mathbf{P}^{\mathrm{s}}_{k+1} & \mathbf{P}^{\mathrm{s}}_{k+1} \mathbf{G}_k^{\mathsf{T}} \\ \mathbf{G}_k \mathbf{P}^{\mathrm{s}}_{k+1} & \mathbf{G}_k \mathbf{P}^{\mathrm{s}}_{k+1} \mathbf{G}_k^{\mathsf{T}} + \tilde{\mathbf{P}}_2 \end{pmatrix}. \end{aligned} \tag{9.8}$$

The marginal distribution of $\mathbf{x}_k$ is then

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}^{\mathrm{s}}_k, \mathbf{P}^{\mathrm{s}}_k), \tag{9.9}$$

where

$$\begin{aligned} \mathbf{m}^{\mathrm{s}}_k &= \mathbf{m}_k + \mathbf{G}_k \left(\mathbf{m}^{\mathrm{s}}_{k+1} - \mathbf{f}(\mathbf{m}_k)\right), \\ \mathbf{P}^{\mathrm{s}}_k &= \mathbf{P}_k + \mathbf{G}_k \left(\mathbf{P}^{\mathrm{s}}_{k+1} - \mathbf{F}_x \mathbf{P}_k \mathbf{F}_x^{\mathsf{T}} - \mathbf{Q}_k\right) \mathbf{G}_k^{\mathsf{T}}. \end{aligned} \tag{9.10}$$

□

The generalization to the non-additive model (5.37) is analogous to the filtering case – we just need to replace the first two of Equations (9.2) with their non-additive versions as in Algorithm 5.5.

Example 9.1 (Pendulum tracking with ERTSS) *The result of applying the ERTSS to the pendulum model in Example 5.1 is shown in Figure 9.1. The resulting RMSE was* 0.033*, which is much lower than the error of the EKF which was* 0.12*. It is also much lower than the errors of any other filters, which were in the range* 0.10–0.12*.*

9.2 Statistically linearized Rauch–Tung–Striebel smoother

The statistically linearized Rauch–Tung–Striebel smoother (SLRTSS) is analogous to the extended Rauch–Tung–Striebel smoother, except that we use statistical linearization instead of ordinary linearization. The statistically linearized Rauch–Tung–Striebel smoother for the additive model (5.24) is the following.

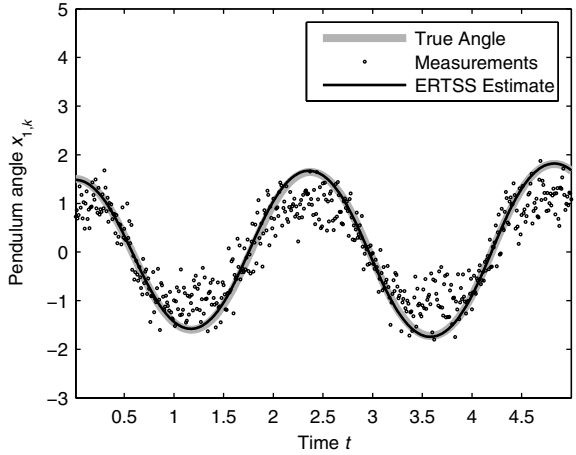


Figure 9.1 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with the ERTSS. The resulting RMSE is 0.033 (recall that the RMSE of the EKF was 0.12).

Algorithm 9.2 (Statistically linearized RTS smoother) *The equations for the statistically linearized RTS smoother are*

$$\begin{aligned}
\mathbf{m}_{k+1}^{-} &= \mathrm{E}[\mathbf{f}(\mathbf{x}_k)], \\
\mathbf{P}_{k+1}^{-} &= \mathrm{E}[\mathbf{f}(\mathbf{x}_k)\,\delta\mathbf{x}_k^{\mathsf{T}}]\,\mathbf{P}_k^{-1}\,\mathrm{E}[\mathbf{f}(\mathbf{x}_k)\,\delta\mathbf{x}_k^{\mathsf{T}}]^{\mathsf{T}} + \mathbf{Q}_k, \\
\mathbf{G}_k &= \mathrm{E}[\mathbf{f}(\mathbf{x}_k)\,\delta\mathbf{x}_k^{\mathsf{T}}]^{\mathsf{T}}\,[\mathbf{P}_{k+1}^{-}]^{-1}, \\
\mathbf{m}_k^{\mathrm{s}} &= \mathbf{m}_k + \mathbf{G}_k\,[\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^{-}], \\
\mathbf{P}_k^{\mathrm{s}} &= \mathbf{P}_k + \mathbf{G}_k\,[\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^{-}]\,\mathbf{G}_k^{\mathsf{T}},
\end{aligned} \tag{9.11}$$

where the expectations are taken with respect to the filtering distribution $\mathbf{x}_k \sim \mathrm{N}(\mathbf{m}_k, \mathbf{P}_k)$.

The generalization to the non-additive case is also straightforward and just amounts to replacing the first two of Equations (9.11) with their non-additive versions as in Algorithm 5.8. In the gain computation we then also need to average over $\mathbf{q}_k$.

It is also possible to use Equation (5.61) for rewriting the cross-terms above in terms of Jacobians of $\mathbf{f}$, but this is left as an exercise to the reader. The SLRTSS can also be considered as a first order truncation of the

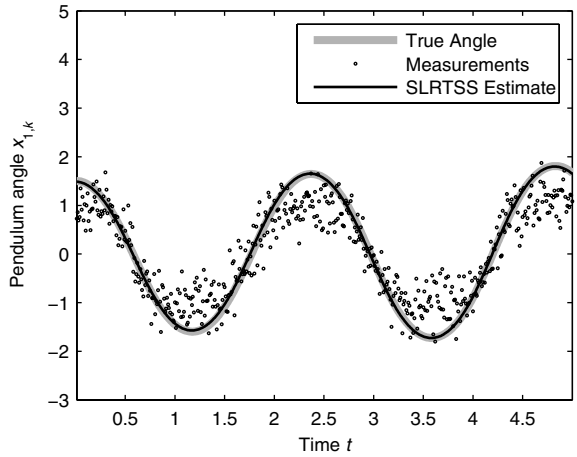


Figure 9.2 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with the SLRTSS. The resulting RMSE is 0.028 which is slightly lower than the error of the ERTSS (which was 0.033).

Fourier–Hermite Rauch–Tung–Striebel smoother (FHRTSS, Sarmavuori and Särkkä, 2012b), where this derivative version also arises naturally.

Example 9.2 (Pendulum tracking with SLRTSS) *The result of applying the SLRTSS to the pendulum model in Example 5.1 is shown in Figure 9.2. The resulting RMSE error of* 0.028 *is slightly lower than the error of the ERTSS which was* 0.033*. The error is also significantly lower than errors of any of the filters, which were in the range* 0.10–0.12*.*

9.3 Unscented Rauch–Tung–Striebel smoother

The *unscented Rauch–Tung–Striebel smoother* (URTSS, Särkkä, 2006; Šimandl and Duník, 2006; Särkkä, 2008) is a Gaussian approximation based smoother where the non-linearity is approximated using the unscented transform. The smoother equations for the additive model (5.37) are given as follows.

Algorithm 9.3 (Unscented Rauch–Tung–Striebel smoother I) *The* additive form unscented RTS smoother algorithm *is the following.*

1 *Form the sigma points:*

$$\begin{aligned}
\mathcal{X}_k^{(0)} &= \mathbf{m}_k, \\
\mathcal{X}_k^{(i)} &= \mathbf{m}_k + \sqrt{n+\lambda}\,\left[\sqrt{\mathbf{P}_k}\right]_i, \\
\mathcal{X}_k^{(i+n)} &= \mathbf{m}_k - \sqrt{n+\lambda}\,\left[\sqrt{\mathbf{P}_k}\right]_i, \quad i = 1, \ldots, n,
\end{aligned} \tag{9.12}$$

where the parameter λ was defined in Equation (5.75)*.*

2 *Propagate the sigma points through the dynamic model:*

$$\hat{\mathcal{X}}_{k+1}^{(i)} = \mathbf{f}\,(\mathcal{X}_k^{(i)}), \quad i = 0, \ldots, 2n.$$

3 *Compute the predicted mean $\mathbf{m}_{k+1}^-$, the predicted covariance $\mathbf{P}_{k+1}^-$, and the cross-covariance $\mathbf{D}_{k+1}$:*

$$\begin{aligned}
\mathbf{m}_{k+1}^- &= \sum_{i=0}^{2n} W_i^{(\mathrm{m})}\, \hat{\mathcal{X}}_{k+1}^{(i)}, \\
\mathbf{P}_{k+1}^- &= \sum_{i=0}^{2n} W_i^{(\mathrm{c})}\, (\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)\, (\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)^{\mathsf{T}} + \mathbf{Q}_k, \\
\mathbf{D}_{k+1} &= \sum_{i=0}^{2n} W_i^{(\mathrm{c})}\, (\mathcal{X}_k^{(i)} - \mathbf{m}_k)\, (\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)^{\mathsf{T}},
\end{aligned} \tag{9.13}$$

where the weights were defined in Equation (5.77)*.*

4 *Compute the smoother gain $\mathbf{G}_k$, the smoothed mean $\mathbf{m}_k^{\mathrm{s}}$, and the covariance $\mathbf{P}_k^{\mathrm{s}}$ as follows:*

$$\begin{aligned}
\mathbf{G}_k &= \mathbf{D}_{k+1}\, [\mathbf{P}_{k+1}^-]^{-1}, \\
\mathbf{m}_k^{\mathrm{s}} &= \mathbf{m}_k + \mathbf{G}_k\, (\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^-), \\
\mathbf{P}_k^{\mathrm{s}} &= \mathbf{P}_k + \mathbf{G}_k\, (\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^-)\, \mathbf{G}_k^{\mathsf{T}}.
\end{aligned} \tag{9.14}$$

The above computations are started from the filtering result of the last time step $\mathbf{m}_T^{\mathrm{s}} = \mathbf{m}_T$, $\mathbf{P}_T^{\mathrm{s}} = \mathbf{P}_T$ and the recursion runs backwards for $k = T-1, \ldots, 0$.

Derivation Assume that the approximate means and covariances of the filtering distributions are available:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \simeq \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k),$$

and the smoothing distribution of time step $k+1$ is known and approximately Gaussian:

$$p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \simeq \mathrm{N}(\mathbf{x}_{k+1} \mid \mathbf{m}_{k+1}^{\mathrm{s}}, \mathbf{P}_{k+1}^{\mathrm{s}}).$$

An unscented transform based approximation to the optimal smoothing solution can be derived as follows.

1 Generate an unscented transform based Gaussian approximation to the joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$:

$$p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) \simeq \mathrm{N}\left(\begin{pmatrix}\mathbf{x}_k \\ \mathbf{x}_{k+1}\end{pmatrix} \middle| \begin{pmatrix}\mathbf{m}_k \\ \mathbf{m}^-_{k+1}\end{pmatrix}, \begin{pmatrix}\mathbf{P}_k & \mathbf{D}_{k+1} \\ \mathbf{D}^\mathsf{T}_{k+1} & \mathbf{P}^-_{k+1}\end{pmatrix}\right). \tag{9.15}$$

This can be done by using the additive form of the unscented transformation in Algorithm 5.12 for the non-linearity $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{q}_k$. This is done in Equations (9.13).

2 Because the distribution (9.15) is Gaussian, by the computation rules of Gaussian distributions the conditional distribution of $\mathbf{x}_k$ is given as

$$p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) \simeq \mathrm{N}(\mathbf{x}_k \mid \tilde{\mathbf{m}}_2, \tilde{\mathbf{P}}_2),$$

where

$$\begin{aligned}\mathbf{G}_k &= \mathbf{D}_{k+1}\,[\mathbf{P}^-_{k+1}]^{-1}, \\ \tilde{\mathbf{m}}_2 &= \mathbf{m}_k + \mathbf{G}_k(\mathbf{x}_{k+1} - \mathbf{m}^-_{k+1}), \\ \tilde{\mathbf{P}}_2 &= \mathbf{P}_k - \mathbf{G}_k\,\mathbf{P}^-_{k+1}\,\mathbf{G}^\mathsf{T}_k.\end{aligned}$$

3 The rest of the derivation is completely analogous to the derivation of the ERTSS in Section 9.1.

□

The corresponding augmented version of the smoother for non-additive models of the form (5.37) is almost the same, except that the augmented UT in Algorithm 5.13 is used instead of the additive UT in Algorithm 5.12. The smoother can be formulated as follows (Särkkä, 2008).

Algorithm 9.4 (Unscented Rauch–Tung–Striebel smoother II) *A single step of the* augmented form unscented RTS smoother *for non-additive models is as follows.*

1 *Form the sigma points for the* $n' = n + n_q$ *-dimensional augmented random variable* $(\mathbf{x}_k, \mathbf{q}_k)$*:*

$$\begin{aligned}
\tilde{\mathcal{X}}_k^{(0)} &= \tilde{\mathbf{m}}_k, \\
\tilde{\mathcal{X}}_k^{(i)} &= \tilde{\mathbf{m}}_k + \sqrt{n' + \lambda'} \left[\sqrt{\tilde{\mathbf{P}}_k}\right]_i, \\
\tilde{\mathcal{X}}_k^{(i+n')} &= \tilde{\mathbf{m}}_k - \sqrt{n' + \lambda'} \left[\sqrt{\tilde{\mathbf{P}}_k}\right]_i, \quad i = 1, \ldots, n',
\end{aligned} \tag{9.16}$$

where

$$\tilde{\mathbf{m}}_k = \begin{pmatrix} \mathbf{m}_k \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\mathbf{P}}_k = \begin{pmatrix} \mathbf{P}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_k \end{pmatrix}.$$

2 *Propagate the sigma points through the dynamic model:*

$$\hat{\mathcal{X}}_{k+1}^{(i)} = \mathbf{f}\,(\tilde{\mathcal{X}}_k^{(i),x}, \tilde{\mathcal{X}}_k^{(i),q}), \quad i = 0, \ldots, 2n',$$

where $\tilde{\mathcal{X}}_k^{(i),x}$ *and* $\tilde{\mathcal{X}}_k^{(i),q}$ *denote the parts of the augmented sigma point* i *which correspond to* $\mathbf{x}_k$ *and* $\mathbf{q}_k$*, respectively.*

3 *Compute the predicted mean* $\mathbf{m}_{k+1}^-$*, the predicted covariance* $\mathbf{P}_{k+1}^-$*, and the cross-covariance* $\mathbf{D}_{k+1}$*:*

$$\begin{aligned}
\mathbf{m}_{k+1}^- &= \sum_{i=0}^{2n'} W_i^{(\mathrm{m})'}\, \hat{\mathcal{X}}_{k+1}^{(i)}, \\
\mathbf{P}_{k+1}^- &= \sum_{i=0}^{2n'} W_i^{(\mathrm{c})'}\, (\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)\, (\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)^{\mathsf{T}}, \\
\mathbf{D}_{k+1} &= \sum_{i=0}^{2n'} W_i^{(\mathrm{c})'}\, (\tilde{\mathcal{X}}_k^{(i),x} - \mathbf{m}_k)\, (\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)^{\mathsf{T}},
\end{aligned} \tag{9.17}$$

where the definitions of the parameter λ' *and the weights* $W_i^{(\mathrm{m})'}$ *and* $W_i^{(\mathrm{c})'}$ *are the same as in Section 5.5.*

4 *Compute the smoother gain* $\mathbf{G}_k$*, the smoothed mean* $\mathbf{m}_k^{\mathrm{s}}$*, and the covariance* $\mathbf{P}_k^{\mathrm{s}}$*:*

$$\begin{aligned}
\mathbf{G}_k &= \mathbf{D}_{k+1}\, [\mathbf{P}_{k+1}^-]^{-1}, \\
\mathbf{m}_k^{\mathrm{s}} &= \mathbf{m}_k + \mathbf{G}_k \left[\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^-\right], \\
\mathbf{P}_k^{\mathrm{s}} &= \mathbf{P}_k + \mathbf{G}_k \left[\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^-\right] \mathbf{G}_k^{\mathsf{T}}.
\end{aligned} \tag{9.18}$$

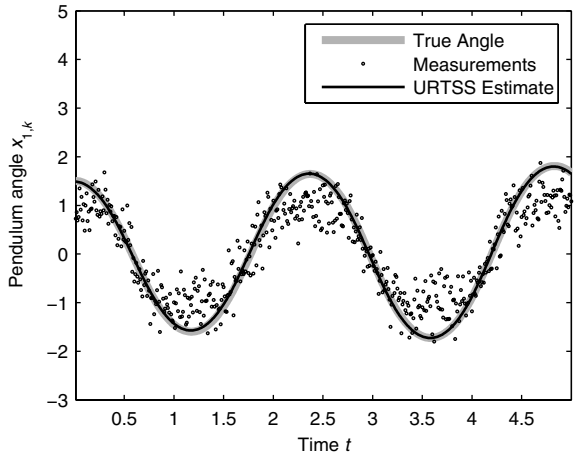


Figure 9.3 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with the URTSS. The resulting RMSE is 0.028 which is the same as with the SLRTSS. Recall that the RMSE of the UKF was 0.11.

Example 9.3 (Pendulum tracking with URTSS) *The result of applying the URTSS to the pendulum model in Example 5.1 is shown in Figure 9.3. The resulting RMSE is* 0.028 *which is the same as for the SLRTSS and lower than that of the ETRSS which is* 0.033. *Thus we get the same error with the URTSS as with statistical linearization but without needing to compute the analytical expectations.*

9.4 Exercises

9.1 Derive and implement the extended RTS smoother to the model in Exercise 5.1 and compare the errors of filters and smoothers.

9.2 Write down the detailed derivation of the (additive form) statistically linearized RTS smoother. You can follow the same steps as in the derivation of the extended RTS smoother.

9.3 Derive and implement the statistically linearized RTS smoother to the model in Exercise 5.1 and compare the errors of filters and smoothers.

9.4 In Exercise 5.3 you derived an alternative (derivative) form of the SLF. Write down the corresponding alternative form of the SLRTS. Derive the smoothing equations for the model in Exercise 9.3 above and compare the equations that you obtain with the equations obtained in Exercise 9.3.

9.5 Implement the unscented transform based RTS smoother to the model in Exercise 5.1 and compare the errors of filters and smoothers.

9.6 Implement the RTS smoother to the bearings only target tracking problem in Exercise 5.5. Note that even though the full model is non-linear, due to the linear dynamic model the smoother is linear.

10

General Gaussian smoothing

As pointed out by Särkkä and Hartikainen (2010a), the unscented Rauch–Tung–Striebel smoother (URTSS) can be considered a special case of more general non-linear Gaussian smoothers, in the same way as the UKF is a special case of Gaussian filters. In this chapter we present the Gaussian smoother and Gauss–Hermite cubature and spherical cubature integration based approximations. We also discuss Gaussian approximation based fixed-lag and fixed-point smoothers.

10.1 General Gaussian Rauch–Tung–Striebel smoother

The Gaussian moment matching described in Section 6.1 can be used in smoothers in an analogous manner to the Gaussian filters in Section 6.2. If we follow the extended RTS smoother derivation in Section 9.1, we get the following algorithm (Särkkä and Hartikainen, 2010a).

Algorithm 10.1 (Gaussian RTS smoother I) *The equations of the additive form* Gaussian RTS smoother *are the following:*

$$
\begin{aligned}
\mathbf{m}_{k+1}^{-} &= \int \mathbf{f}(\mathbf{x}_k)\, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k)\, \mathrm{d}\mathbf{x}_k, \\
\mathbf{P}_{k+1}^{-} &= \int [\mathbf{f}(\mathbf{x}_k) - \mathbf{m}_{k+1}^{-}]\, [\mathbf{f}(\mathbf{x}_k) - \mathbf{m}_{k+1}^{-}]^{\mathsf{T}}\, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k)\, \mathrm{d}\mathbf{x}_k + \mathbf{Q}_k, \\
\mathbf{D}_{k+1} &= \int [\mathbf{x}_k - \mathbf{m}_k]\, [\mathbf{f}(\mathbf{x}_k) - \mathbf{m}_{k+1}^{-}]^{\mathsf{T}}\, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k)\, \mathrm{d}\mathbf{x}_k, \\
\mathbf{G}_k &= \mathbf{D}_{k+1}\, [\mathbf{P}_{k+1}^{-}]^{-1}, \\
\mathbf{m}_k^{\mathrm{s}} &= \mathbf{m}_k + \mathbf{G}_k\, (\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^{-}), \\
\mathbf{P}_k^{\mathrm{s}} &= \mathbf{P}_k + \mathbf{G}_k\, (\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^{-})\, \mathbf{G}_k^{\mathsf{T}}.
\end{aligned}
\tag{10.1}
$$

The integrals above can be approximated using analogous numerical integration or analytical approximation schemes as in the filtering case, that

is, with Gauss–Hermite cubatures (Ito and Xiong, 2000; Wu et al., 2006), spherical cubature rules (Arasaratnam and Haykin, 2009), or with many other numerical integration schemes. In the non-additive case the Gaussian smoother becomes the following (Särkkä and Hartikainen, 2010a).

Algorithm 10.2 (Gaussian RTS smoother II) *The equations of the non-additive form* Gaussian RTS smoother *are the following:*

$$
\begin{aligned}
\mathbf{m}_{k+1}^- &= \int \mathbf{f}(\mathbf{x}_k, \mathbf{q}_k)\, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k)\, \mathrm{N}(\mathbf{q}_k \mid \mathbf{0}, \mathbf{Q}_k)\, \mathrm{d}\mathbf{x}_k\, \mathrm{d}\mathbf{q}_k, \\
\mathbf{P}_{k+1}^- &= \int [\mathbf{f}(\mathbf{x}_k, \mathbf{q}_k) - \mathbf{m}_{k+1}^-]\, [\mathbf{f}(\mathbf{x}_k, \mathbf{q}_k) - \mathbf{m}_{k+1}^-]^\mathsf{T} \\
&\qquad \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k)\, \mathrm{N}(\mathbf{q}_k \mid \mathbf{0}, \mathbf{Q}_k)\, \mathrm{d}\mathbf{x}_k\, \mathrm{d}\mathbf{q}_k, \\
\mathbf{D}_{k+1} &= \int [\mathbf{x}_k - \mathbf{m}_k]\, [\mathbf{f}(\mathbf{x}_k, \mathbf{q}_k) - \mathbf{m}_{k+1}^-]^\mathsf{T} \\
&\qquad \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k)\, \mathrm{N}(\mathbf{q}_k \mid \mathbf{0}, \mathbf{Q}_k)\, \mathrm{d}\mathbf{x}_k\, \mathrm{d}\mathbf{q}_k, \\
\mathbf{G}_k &= \mathbf{D}_{k+1}\, [\mathbf{P}_{k+1}^-]^{-1}, \\
\mathbf{m}_k^\mathrm{s} &= \mathbf{m}_k + \mathbf{G}_k\, (\mathbf{m}_{k+1}^\mathrm{s} - \mathbf{m}_{k+1}^-), \\
\mathbf{P}_k^\mathrm{s} &= \mathbf{P}_k + \mathbf{G}_k\, (\mathbf{P}_{k+1}^\mathrm{s} - \mathbf{P}_{k+1}^-)\, \mathbf{G}_k^\mathsf{T}.
\end{aligned} \tag{10.2}
$$

As in the Gaussian filtering case, the above algorithms are mainly theoretical, because the integrals can be solved in closed form only in special cases.

10.2 Gauss–Hermite Rauch–Tung–Striebel smoother

By using the Gauss–Hermite cubature integration approximation from Section 6.3 in the additive form Gaussian RTS smoother, we get the following Gauss–Hermite Rauch–Tung–Striebel smoother (GHRTSS) algorithm.

Algorithm 10.3 (Gauss–Hermite Rauch–Tung–Striebel smoother) *The* additive form Gauss–Hermite RTS smoother algorithm *is the following.*

1 Form the sigma points as

$$
\mathcal{X}_k^{(i_1,\ldots,i_n)} = \mathbf{m}_k + \sqrt{\mathbf{P}_k}\, \boldsymbol{\xi}^{(i_1,\ldots,i_n)}, \qquad i_1, \ldots, i_n = 1, \ldots, p, \tag{10.3}
$$

where the unit sigma points $\boldsymbol{\xi}^{(i_1,\ldots,i_n)}$ *were defined in Equation* (6.24).

2 Propagate the sigma points through the dynamic model:

$$
\hat{\mathcal{X}}_{k+1}^{(i_1,\ldots,i_n)} = \mathbf{f}(\mathcal{X}_k^{(i_1,\ldots,i_n)}), \quad i_1, \ldots, i_n = 1, \ldots, p. \tag{10.4}
$$

3 *Compute the predicted mean* $\mathbf{m}_{k+1}^-$*, the predicted covariance* $\mathbf{P}_{k+1}^-$*, and the cross-covariance* $\mathbf{D}_{k+1}$*:*

$$\begin{aligned}
\mathbf{m}_{k+1}^- &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} \hat{\mathcal{X}}_{k+1}^{(i_1,\ldots,i_n)}, \\
\mathbf{P}_{k+1}^- &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} (\hat{\mathcal{X}}_{k+1}^{(i_1,\ldots,i_n)} - \mathbf{m}_{k+1}^-) (\hat{\mathcal{X}}_{k+1}^{(i_1,\ldots,i_n)} - \mathbf{m}_{k+1}^-)^\mathsf{T} + \mathbf{Q}_k, \\
\mathbf{D}_{k+1} &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} (\mathcal{X}_k^{(i_1,\ldots,i_n)} - \mathbf{m}_k) (\hat{\mathcal{X}}_{k+1}^{(i_1,\ldots,i_n)} - \mathbf{m}_{k+1}^-)^\mathsf{T},
\end{aligned} \tag{10.5}$$

where the weights $W_{i_1,\ldots,i_n}$ *were defined in Equation* (6.23)*.*

4 *Compute the gain* $\mathbf{G}_k$*, mean* $\mathbf{m}_k^\mathrm{s}$ *and covariance* $\mathbf{P}_k^\mathrm{s}$ *as follows:*

$$\begin{aligned}
\mathbf{G}_k &= \mathbf{D}_{k+1} [\mathbf{P}_{k+1}^-]^{-1}, \\
\mathbf{m}_k^\mathrm{s} &= \mathbf{m}_k + \mathbf{G}_k (\mathbf{m}_{k+1}^\mathrm{s} - \mathbf{m}_{k+1}^-), \\
\mathbf{P}_k^\mathrm{s} &= \mathbf{P}_k + \mathbf{G}_k (\mathbf{P}_{k+1}^\mathrm{s} - \mathbf{P}_{k+1}^-) \mathbf{G}_k^\mathsf{T}.
\end{aligned} \tag{10.6}$$

It would also be possible to formulate a non-additive version of the above smoother analogously, but due to unpleasant exponential computational scaling of the Gauss–Hermite cubature method in the state dimension, that extension is not very useful in practice. Recall that the state dimension doubles when using the non-additive transform, because we need to integrate over the state and process noise jointly.

Example 10.1 (Pendulum tracking with GHRTSS) *The result of applying GHRTSS to the pendulum model in Example 5.1 is shown in Figure 10.1. The resulting RMSE error is* 0.028*, which is the same as what we obtained with the SLRTS and URTSS. Recall that the error of the ERTSS was a bit higher,* 0.033*. However, it seems that using the higher order integration method does not really help much in this particular problem, as we already concluded when comparing the performance of the GHKF to the other filters in this same problem.*

10.3 Cubature Rauch–Tung–Striebel smoother

By using the third order spherical cubature approximation (Section 6.5) to the additive form Gaussian RTS smoother, we get the following cubature Rauch–Tung–Striebel smoother (CRTSS) algorithm (see Arasaratnam and Haykin, 2011).

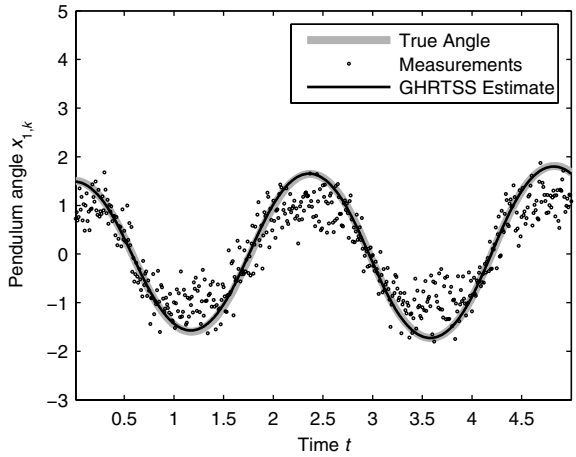


Figure 10.1 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with the GHRTSS. The resulting RMSE is 0.028, which is the same as with the other smoothers encountered so far (except the ERTSS which had error 0.033).

Algorithm 10.4 (Cubature Rauch–Tung–Striebel smoother I) *The* additive form cubature RTS smoother algorithm *is the following.*

1 Form the sigma points:

$$\mathcal{X}_k^{(i)} = \mathbf{m}_k + \sqrt{\mathbf{P}_k}\,\boldsymbol{\xi}^{(i)}, \qquad i = 1, \ldots, 2n, \tag{10.7}$$

where the unit sigma points are defined as

$$\boldsymbol{\xi}^{(i)} = \begin{cases} \sqrt{n}\,\mathbf{e}_i, & i = 1, \ldots, n, \\ -\sqrt{n}\,\mathbf{e}_{i-n}, & i = n+1, \ldots, 2n, \end{cases} \tag{10.8}$$

where $\mathbf{e}_i$ *denotes a unit vector in the direction of the coordinate axis* i.

2 Propagate the sigma points through the dynamic model:

$$\hat{\mathcal{X}}_{k+1}^{(i)} = \mathbf{f}(\mathcal{X}_k^{(i)}), \quad i = 1, \ldots, 2n.$$

3 Compute the predicted mean $\mathbf{m}_{k+1}^-$*, the predicted covariance* $\mathbf{P}_{k+1}^-$*, and the cross-covariance* $\mathbf{D}_{k+1}$*:*

$$\begin{aligned}
\mathbf{m}_{k+1}^- &= \frac{1}{2n}\sum_{i=1}^{2n} \hat{\mathcal{X}}_{k+1}^{(i)}, \\
\mathbf{P}_{k+1}^- &= \frac{1}{2n}\sum_{i=1}^{2n} (\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)\,(\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)^\mathsf{T} + \mathbf{Q}_k, \\
\mathbf{D}_{k+1} &= \frac{1}{2n}\sum_{i=1}^{2n} (\mathcal{X}_k^{(i)} - \mathbf{m}_k)\,(\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)^\mathsf{T}.
\end{aligned} \tag{10.9}$$

4 Compute the gain $\mathbf{G}_k$*, mean* $\mathbf{m}_k^{\mathrm{s}}$ *and covariance* $\mathbf{P}_k^{\mathrm{s}}$ *as follows:*

$$\begin{aligned}
\mathbf{G}_k &= \mathbf{D}_{k+1}\,[\mathbf{P}_{k+1}^-]^{-1}, \\
\mathbf{m}_k^{\mathrm{s}} &= \mathbf{m}_k + \mathbf{G}_k\,(\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^-), \\
\mathbf{P}_k^{\mathrm{s}} &= \mathbf{P}_k + \mathbf{G}_k\,(\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^-)\,\mathbf{G}_k^\mathsf{T}.
\end{aligned} \tag{10.10}$$

By using the third order spherical cubature approximation to the non-additive form Gaussian RTS smoother, we get the following algorithm.

Algorithm 10.5 (Cubature Rauch–Tung–Striebel smoother II) *A single step of the* augmented form cubature RTS smoother *is as follows.*

1 Form the sigma points for the $n' = n + n_q$*-dimensional augmented random variable* $(\mathbf{x}_k, \mathbf{q}_k)$*:*

$$\tilde{\mathcal{X}}_k^{(i)} = \tilde{\mathbf{m}}_k + \sqrt{\tilde{\mathbf{P}}_k}\,\boldsymbol{\xi}^{(i)'}, \qquad i = 1, \ldots, 2n', \tag{10.11}$$

where

$$\tilde{\mathbf{m}}_k = \begin{pmatrix} \mathbf{m}_k \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\mathbf{P}}_k = \begin{pmatrix} \mathbf{P}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_k \end{pmatrix}.$$

2 Propagate the sigma points through the dynamic model:

$$\hat{\mathcal{X}}_{k+1}^{(i)} = \mathbf{f}\,(\tilde{\mathcal{X}}_k^{(i),x}, \tilde{\mathcal{X}}_k^{(i),q}), \quad i = 1, \ldots, 2n',$$

where $\tilde{\mathcal{X}}_k^{(i),x}$ *and* $\tilde{\mathbf{X}}_k^{(i),q}$ *denote the parts of the augmented sigma point* i *which correspond to* $\mathbf{x}_k$ *and* $\mathbf{q}_k$*, respectively.*

3 Compute the predicted mean $\mathbf{m}_{k+1}^-$*, the predicted covariance* $\mathbf{P}_{k+1}^-$*, and the cross-covariance* $\mathbf{D}_{k+1}$*:*

$$\begin{aligned}
\mathbf{m}_{k+1}^- &= \frac{1}{2n'} \sum_{i=1}^{2n'} \hat{\mathcal{X}}_{k+1}^{(i)}, \\
\mathbf{P}_{k+1}^- &= \frac{1}{2n'} \sum_{i=1}^{2n'} (\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)\,(\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)^{\mathsf{T}}, \\
\mathbf{D}_{k+1} &= \frac{1}{2n'} \sum_{i=1}^{2n'} (\tilde{\mathcal{X}}_{k}^{(i),x} - \mathbf{m}_{k})\,(\hat{\mathcal{X}}_{k+1}^{(i)} - \mathbf{m}_{k+1}^-)^{\mathsf{T}}.
\end{aligned} \tag{10.12}$$

4 Compute the gain $\mathbf{G}_k$*, mean* $\mathbf{m}_k^{\mathrm{s}}$*, and covariance* $\mathbf{P}_k^{\mathrm{s}}$*:*

$$\begin{aligned}
\mathbf{G}_k &= \mathbf{D}_{k+1}\,[\mathbf{P}_{k+1}^-]^{-1}, \\
\mathbf{m}_k^{\mathrm{s}} &= \mathbf{m}_k + \mathbf{G}_k\,\left(\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^-\right), \\
\mathbf{P}_k^{\mathrm{s}} &= \mathbf{P}_k + \mathbf{G}_k\,\left(\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^-\right)\,\mathbf{G}_k^{\mathsf{T}}.
\end{aligned} \tag{10.13}$$

It is easy to see that the above algorithms are indeed special cases of the URTSS methods with parameters $\alpha = 1$, $\beta = 0$, $\kappa = 0$. However, this particular selection of parameters tends to work well in practice and due to the simplicity of sigma-point and weight selection rules, the method is very simple to implement.

Example 10.2 (Pendulum tracking with CRTSS) *The result of applying the CRTSS for the pendulum model in Example 5.1 is shown in Figure 10.2. As expected, the error* 0.028 *and the overall result are practically identical to the result of the URTSS, as well as to the SLRTSS and GHRTSS.*

10.4 General fixed-point smoother equations

The smoother algorithms that we have considered this far have all been *fixed-interval* smoothing algorithms, which can be used for computing estimates of a fixed time interval of states given the measurements on the same interval. However, there exist a couple of other types of smoothing problem as well.

- *Fixed-point smoothing* refers to a methodology which can be used for efficiently computing the optimal estimate of the *initial state* or some other fixed-time state of a state space model, given an increasing number of measurements after it. This kind of estimation problem arises, for example, in estimation of the state of a spacecraft at a given point of time

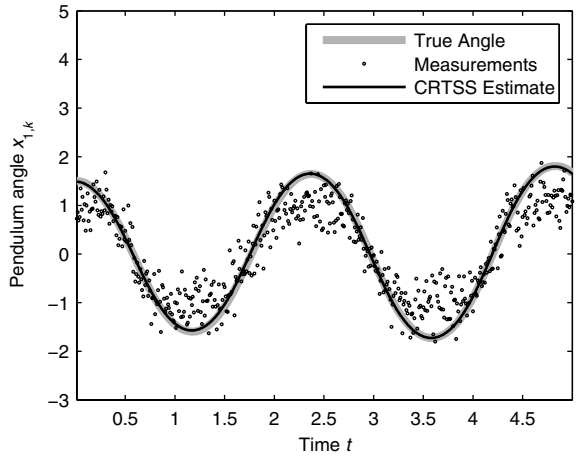


Figure 10.2 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with the CRTSS. The resulting RMSE is 0.028, which is the same as with the other Gaussian approximation based non-linear RTS smoothers, except the ETRSS.

in the past (Meditch, 1969) or in alignment and calibration of inertial navigation systems (Grewal et al., 1988).

- *Fixed-lag smoothing* is a methodology for computing delayed estimates of state space models given measurements up to the current time and a constant horizon in the future. Fixed-lag smoothing can be considered as optimal filtering, where a constant delay is tolerated in the estimates. Potential applications of fixed-lag smoothing are all the problems where optimal filters are typically applied, and where the small delay is tolerated. An example of such application is digital demodulation (Tam et al., 1973).

The presentation here is based on the results presented in Särkkä and Hartikainen (2010a), except that the derivations are presented in a bit more detail than in the original reference.

The general fixed-interval RTS smoothers described in this book have the property that, given the gain sequence, we only need linear operations for performing the smoothing and, in this sense, the smoothing is a completely *linear operation*. The only non-linear operations in the smoother are in the approximations of the Gaussian integrals. However, these operations

are performed on the filtering results and thus we can compute the smoothing gain sequence $\mathbf{G}_k$ from the filtering results in a causal manner. Because of these properties we may now derive a fixed-point smoother using similar methods as have been used for deriving the linear fixed-point smoother from the linear Rauch–Tung–Striebel smoother in Meditch (1969).

We will now denote the smoothing means and covariances using notation of the type $\mathbf{m}_{k|n}$ and $\mathbf{P}_{k|n}$, which refer to the mean and covariance of the state $\mathbf{x}_k$, which is conditioned to the measurements $\mathbf{y}_1, \ldots, \mathbf{y}_n$. With this notation, the filter estimates are $\mathbf{m}_{k|k}$, $\mathbf{P}_{k|k}$ and the RTS smoother estimates, which are conditioned to the measurements $\mathbf{y}_1, \ldots, \mathbf{y}_T$, have the form $\mathbf{m}_{k|T}$, $\mathbf{P}_{k|T}$. The RTS smoothers have the following common recursion equations:

$$\begin{aligned}
\mathbf{G}_k &= \mathbf{D}_{k+1}\,[\mathbf{P}_{k+1|k}]^{-1}, \\
\mathbf{m}_{k|T} &= \mathbf{m}_{k|k} + \mathbf{G}_k\,\left(\mathbf{m}_{k+1|T} - \mathbf{m}_{k+1|k}\right), \\
\mathbf{P}_{k|T} &= \mathbf{P}_{k|k} + \mathbf{G}_k\,\left(\mathbf{P}_{k+1|T} - \mathbf{P}_{k+1|k}\right)\,\mathbf{G}_k^{\mathsf{T}},
\end{aligned} \tag{10.14}$$

which are indeed linear recursion equations for the smoother mean and covariance. Note that the gains $\mathbf{G}_k$ depend only on the filtering results, not on the smoother mean and covariance. Because the gains $\mathbf{G}_k$ are independent of T, from Equations (10.14) we get for $i = j, \ldots, k$ the identity

$$\mathbf{m}_{i|k} - \mathbf{m}_{i|i} = \mathbf{G}_i[\mathbf{m}_{i+1|k} - \mathbf{m}_{i+1|i}]. \tag{10.15}$$

Similarly, for $i = j, \ldots, k-1$ we have

$$\mathbf{m}_{i|k-1} - \mathbf{m}_{i|i} = \mathbf{G}_i[\mathbf{m}_{i+1|k-1} - \mathbf{m}_{i+1|i}]. \tag{10.16}$$

Subtracting these equations gives the identity

$$\mathbf{m}_{i|k} - \mathbf{m}_{i|k-1} = \mathbf{G}_i[\mathbf{m}_{i+1|k} - \mathbf{m}_{i+1|k-1}]. \tag{10.17}$$

By varying i from j to $k-1$ we get the identities

$$\begin{aligned}
\mathbf{m}_{j|k} - \mathbf{m}_{j|k-1} &= \mathbf{G}_j[\mathbf{m}_{j+1|k} - \mathbf{m}_{j+1|k-1}], \\
\mathbf{m}_{j+1|k} - \mathbf{m}_{j+1|k-1} &= \mathbf{G}_{j+1}[\mathbf{m}_{j+2|k} - \mathbf{m}_{j+2|k-1}], \\
&\vdots \\
\mathbf{m}_{k-1|k} - \mathbf{m}_{k-1|k-1} &= \mathbf{G}_{k-1}[\mathbf{m}_{k|k} - \mathbf{m}_{k|k-1}].
\end{aligned} \tag{10.18}$$

If we sequentially substitute the above equations to each other starting from the last and proceeding to the first, we get the equation

$$\mathbf{m}_{j|k} = \mathbf{m}_{j|k-1} + \mathbf{B}_{j|k}[\mathbf{m}_{k|k} - \mathbf{m}_{k|k-1}], \tag{10.19}$$

where

$$\mathbf{B}_{j|k} = \mathbf{G}_j \times \cdots \times \mathbf{G}_{k-1}. \tag{10.20}$$

Analogously, for the covariance we get

$$\mathbf{P}_{j|k} = \mathbf{P}_{j|k-1} + \mathbf{B}_{j|k}[\mathbf{P}_{k|k} - \mathbf{P}_{k|k-1}]\mathbf{B}_{j|k}^{\mathsf{T}}. \tag{10.21}$$

Algorithm 10.6 (General Gaussian fixed-point smoother) *The* general Gaussian fixed-point smoother *algorithm for smoothing the time point j can be implemented by performing the following operations at each time step $k = 1, 2, 3, \ldots$*

1 Gain computation: *Compute the predicted mean $\mathbf{m}_{k|k-1}$, predicted covariance $\mathbf{P}_{k|k-1}$ and cross-covariance $\mathbf{D}_k$ from the filtering results. Then compute the gain from the equation*

$$\mathbf{G}_{k-1} = \mathbf{D}_k\,[\mathbf{P}_{k|k-1}]^{-1}. \tag{10.22}$$

2 Fixed-point smoothing:

(a) If $k < j$, just store the filtering result.

(b) If $k = j$, set $\mathbf{B}_{j|j} = \mathbf{I}$. The fixed-point smoothed mean and covariance on step j are equal to the filtered mean and covariance $\mathbf{m}_{j|j}$ and $\mathbf{P}_{j|j}$.

(c) If $k > j$, compute the smoothing gain and the fixed-point smoother mean and covariance:

$$\begin{aligned}
\mathbf{B}_{j|k} &= \mathbf{B}_{j|k-1}\mathbf{G}_{k-1}, \\
\mathbf{m}_{j|k} &= \mathbf{m}_{j|k-1} + \mathbf{B}_{j|k}[\mathbf{m}_{k|k} - \mathbf{m}_{k|k-1}], \\
\mathbf{P}_{j|k} &= \mathbf{P}_{j|k-1} + \mathbf{B}_{j|k}[\mathbf{P}_{k|k} - \mathbf{P}_{k|k-1}]\mathbf{B}_{j|k}^{\mathsf{T}}.
\end{aligned} \tag{10.23}$$

Because only a constant number of computations is needed on each time step, the algorithm can be easily implemented in real time.

10.5 General fixed-lag smoother equations

It is also possible to derive a general fixed-lag smoother by using a similar procedure as in the previous section. However, this approach will lead to a numerically unstable algorithm, as we will see shortly. Let n be the number of lags. From the fixed-point smoother we get

$$\begin{aligned}
\mathbf{m}_{k-n-1|k} &= \mathbf{m}_{k-n-1|k-1} \\
&\quad + \mathbf{B}_{k-n-1|k}[\mathbf{m}_{k|k} - \mathbf{m}_{k|k-1}].
\end{aligned} \tag{10.24}$$

From the fixed-interval smoother we get

$$\begin{aligned}\mathbf{m}_{k-n-1|k} &= \mathbf{m}_{k-n-1|k-n-1} \\ &\quad + \mathbf{G}_{k-1-n}[\mathbf{m}_{k-n|k} - \mathbf{m}_{k-n|k-n-1}].\end{aligned} \tag{10.25}$$

Equating the right-hand sides, and solving for $\mathbf{m}_{k-n|k}$ while remembering the identity $\mathbf{B}_{k-n|k} = \mathbf{G}_{k-n-1}^{-1}\mathbf{B}_{k-n-1|k}$ results in the smoothing equation

$$\begin{aligned}\mathbf{m}_{k-n|k} &= \mathbf{m}_{k-n|k-n-1} \\ &\quad + \mathbf{G}_{k-n-1}^{-1}[\mathbf{m}_{k-n-1|k-1} - \mathbf{m}_{k-n-1|k-n-1}] \\ &\quad + \mathbf{B}_{k-n|k}[\mathbf{m}_{k|k} - \mathbf{m}_{k|k-1}].\end{aligned} \tag{10.26}$$

Similarly, for covariance we get

$$\begin{aligned}\mathbf{P}_{k-n|k} &= \mathbf{P}_{k-n|k-n-1} \\ &\quad + \mathbf{G}_{k-n-1}^{-1}[\mathbf{P}_{k-n-1|k-1} - \mathbf{P}_{k-n-1|k-n-1}]\mathbf{G}_{k-n-1}^{-\mathsf{T}} \\ &\quad + \mathbf{B}_{k-n|k}[\mathbf{P}_{k|k} - \mathbf{P}_{k|k-1}]\mathbf{B}_{k-n|k}^{\mathsf{T}}.\end{aligned} \tag{10.27}$$

Equations (10.26) and (10.27) can, *in principle*, be used for recursively computing the fixed-lag smoothing solution. The number of computations does not depend on the lag length. This solution can be seen to be of the same form as the fixed-lag smoother given in Rauch (1963), Meditch (1969), and Crassidis and Junkins (2004). Unfortunately, it has been shown (Kelly and Anderson, 1971) that this form of smoother is *numerically unstable* and thus not usable in practice. However, sometimes the equations do indeed work and can be used if the user is willing to take the risk of potential instability.

Moore (1973) and Moore and Tam (1973) have derived stable algorithms for optimal fixed-lag smoothing by augmenting the n lagged states to a Kalman filter. This approach ensures the stability of the algorithm. Using certain simplifications it is possible to reduce the computations, and this is also possible when certain types of extended Kalman filters are used (Moore, 1973; Moore and Tam, 1973). Unfortunately, such simplifications cannot be done in the more general case and, for example, when the unscented transformation (Julier et al., 1995, 2000) or a quadrature rule (Ito and Xiong, 2000) is used, the required number of computations becomes high, because the Cholesky factorization of the whole joint covariance of the n lagged states would be needed in the computations. Another possibility, which is employed here, is to take advantage of the fact that Rauch–Tung–Striebel smoother equations are numerically stable and can be used for fixed-lag smoothing. The fixed-lag smoothing can be efficiently implemented by taking into account that the gain sequence needs to be evaluated

only once, and the same gains can be used in different smoothers operating on different intervals.

Algorithm 10.7 (General Gaussian fixed-lag smoother) *Thus the* general Gaussian fixed-lag smoother *can be implemented by performing the following on each time step* $k = 1, 2, 3, \ldots$

1 Gain computation: *During the Gaussian filter prediction step compute and store the predicted mean* $\mathbf{m}_{k|k-1}$*, predicted covariance* $\mathbf{P}_{k|k-1}$ *and cross-covariance* $\mathbf{D}_k$*. Also compute and store the smoothing gain*

$$\mathbf{G}_{k-1} = \mathbf{D}_k \left[\mathbf{P}_{k|k-1}\right]^{-1}. \tag{10.28}$$

2 Fixed-lag smoothing: *Using the stored gain sequence, compute the smoothing solutions for steps* $j = k - n, \ldots, k$ *using the following backward recursion, starting from the filtering solution on step* $j = k$*:*

$$\begin{aligned} \mathbf{m}_{j|k} &= \mathbf{m}_{j|j} + \mathbf{G}_j \left[\mathbf{m}_{j+1|k} - \mathbf{m}_{j+1|j}\right], \\ \mathbf{P}_{j|k} &= \mathbf{P}_{j|j} + \mathbf{G}_j \left[\mathbf{P}_{j+1|k} - \mathbf{P}_{j+1|j}\right] \mathbf{G}_j^{\mathsf{T}}. \end{aligned} \tag{10.29}$$

The required number of computations per time step grows linearly with the length of lag. Thus the computational requirements are comparable to the algorithms presented in Moore (1973) and Moore and Tam (1973). The algorithm defined in Equations (10.26) and (10.27) would be computationally more efficient but, as already stated, it would be numerically unstable.

10.6 Exercises

10.1 Implement the third order spherical cubature integration based RTS smoother for the model in Exercise 5.1 and compare the errors of the filters and smoothers.

10.2 Implement a fixed point smoother (you can choose the brand freely) for inferring the initial state of the above model. Check that the result matches the corresponding RTS smoother result and plot the error as a function of time steps.

11

Particle smoothing

When smoothing solutions to non-linear/non-Gaussian problems are sought, Gaussian approximations might not lead to accurate enough approximations. In that case it is better to use Monte Carlo (particle) approximations which in principle can be used for approximating arbitrary smoothing distributions. Although the same SIR algorithm which is used for particle filtering provides an approximation to the smoothing distribution as a by-product, it does not yet solve the problem of particle smoothing. The challenge is that the resulting approximation tends to be degenerate. For this reason other types of particle smoothers have been developed and here we present the most commonly used ones, the backward-simulation smoother and the reweighting based (marginal) particle smoother.

11.1 SIR particle smoother

The *SIR particle smoother* (SIR-PS) of Kitagawa (1996) is based on direct usage of the SIR for smoothing. Recall that in Section 7.3 we derived the sequential importance sampling (SIS) method to approximate the full posterior distribution, not just the filtering distributions. We then discarded the sample histories $\mathbf{x}_{0:k-1}^{(i)}$ and only kept the current states $\mathbf{x}_k^{(i)}$, because we were interested in the filtering distributions. But we can get an approximation to the smoothing distribution by keeping the full histories. To get the smoothing solution from sequential importance resampling (SIR) we also need to resample the state histories, not only the current states, to prevent the resampling from breaking the state histories. The resulting algorithm is the following.

Algorithm 11.1 (SIR particle smoother) *The direct sequential importance resampling (SIR) based smoother algorithm is the following.*

- *Draw N samples $\mathbf{x}_0^{(i)}$ from the prior:*

$$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), \qquad i = 1, \ldots, N, \tag{11.1}$$

and set $w_0^{(i)} = 1/N$, for all $i = 1, \ldots, N$. Initialize the state histories to contain the prior samples $\mathbf{x}_0^{(i)}$.

- *For each $k = 1, \ldots, T$ do the following:*

  1 *Draw N new samples $\mathbf{x}_k^{(i)}$ from the importance distributions:*

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k}), \qquad i = 1, \ldots, N, \tag{11.2}$$

  where $\mathbf{x}_{k-1}^{(i)}$ is the $k-1$th (last) element in the sample history $\mathbf{x}_{0:k-1}^{(i)}$.

  2 *Calculate the new weights according to*

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k})} \tag{11.3}$$

  and normalize them to sum to unity.

  3 *Append the samples to the state histories:*

$$\mathbf{x}_{0:k}^{(i)} = (\mathbf{x}_{0:k-1}^{(i)}, \mathbf{x}_k^{(i)}). \tag{11.4}$$

  4 *If the effective number of particles* (7.27) *is too low, perform resampling on the state histories.*

The approximation to the full posterior (smoothed) distribution is (Kitagawa, 1996; Doucet et al., 2000)

$$p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}) \approx \sum_{i=1}^{N} w_T^{(i)} \delta(\mathbf{x}_{0:T} - \mathbf{x}_{0:T}^{(i)}). \tag{11.5}$$

The approximation to the smoothed posterior distribution at time step k, given the measurements up to the time step $T > k$, is

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) \approx \sum_{i=1}^{N} w_T^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}), \tag{11.6}$$

where $\mathbf{x}_k^{(i)}$ is the kth component in $\mathbf{x}_{0:T}^{(i)}$. However, if $T \gg k$ the direct SIR smoother algorithm is known to produce very degenerate approximations (Kitagawa, 1996; Doucet et al., 2000).

11.2 Backward-simulation particle smoother

A less degenerate particle smoother than the SIR particle smoother can be obtained by reusing the filtering results instead of simply storing the full particle histories in SIR. The *backward-simulation particle smoother* (BS-PS) of Godsill et al. (2004) is based on simulation of individual trajectories backwards, starting from the last step and proceeding to the first. The algorithm is the following.

Algorithm 11.2 (Backward-simulation particle smoother) *Given the weighted set of particles* $\{w_k^{(i)}, \mathbf{x}_k^{(i)} : i = 1, \ldots, N,\ k = 1, \ldots, T\}$ *representing the filtering distributions:*

- *Choose* $\tilde{\mathbf{x}}_T = \mathbf{x}_T^{(i)}$ *with probability* $w_T^{(i)}$.
- *For* $k = T - 1, \ldots, 0$*:*

  1 *Compute new weights by*

$$w_{k|k+1}^{(i)} \propto w_k^{(i)}\, p(\tilde{\mathbf{x}}_{k+1} \mid \mathbf{x}_k^{(i)}). \tag{11.7}$$

  2 *Choose* $\tilde{\mathbf{x}}_k = \mathbf{x}_k^{(i)}$ *with probability* $w_{k|k+1}^{(i)}$.

Derivation Assume that we have already simulated a trajectory $\tilde{\mathbf{x}}_{k+1:T}$ from the smoothing distribution. By using Equation (8.3) we get

$$\begin{aligned} p(\mathbf{x}_k \mid \tilde{\mathbf{x}}_{k+1}, \mathbf{y}_{1:T}) &= \frac{p(\tilde{\mathbf{x}}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k})}{p(\tilde{\mathbf{x}}_{k+1} \mid \mathbf{y}_{1:k})} \\ &= Z\, p(\tilde{\mathbf{x}}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k}), \end{aligned} \tag{11.8}$$

where Z is a normalization constant. By substituting the SIR filter approximation in Equation (7.31) we get

$$p(\mathbf{x}_k \mid \tilde{\mathbf{x}}_{k+1}, \mathbf{y}_{1:T}) \approx Z \sum_i w_k^{(i)}\, p(\tilde{\mathbf{x}}_{k+1} \mid \mathbf{x}_k)\, \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}). \tag{11.9}$$

We can now draw a sample from this distribution by sampling $\mathbf{x}_k^{(i)}$ with probability $\propto w_k^{(i)}\, p(\tilde{\mathbf{x}}_{k+1} \mid \mathbf{x}_k^{(i)})$. □

Given S iterations of Algorithm 11.2, resulting in sample trajectories $\tilde{\mathbf{x}}_{0:T}^{(j)}$ for $j = 1, \ldots, S$, the smoothing distribution can now be approximated as

$$p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}) \approx \frac{1}{S} \sum_j \delta(\mathbf{x}_{0:T} - \tilde{\mathbf{x}}_{0:T}^{(j)}). \tag{11.10}$$

The marginal distribution samples for a step k can be obtained by extracting the kth components from the above trajectories. The computational

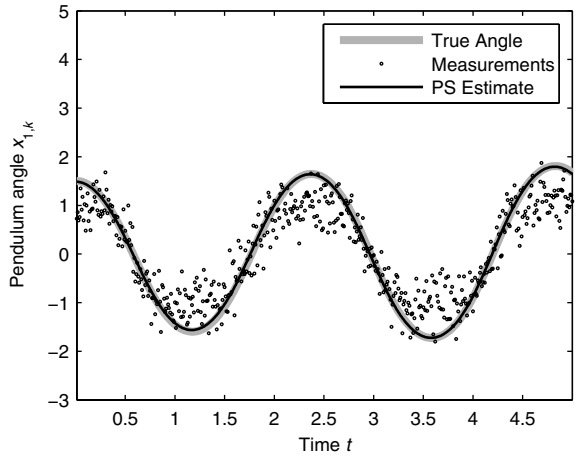


Figure 11.1 Simulated pendulum data and the result of tracking the pendulum described in Example 5.1 with the backward-simulation smoother. The resulting RMSE is 0.028 which is the same as with the non-linear Gaussian RTS smoothers (all of them had RMSE 0.028, except the ERTSS which had 0.033). Recall that the RMSE of the bootstrap filter was 0.12 and thus the smoother reduces the error significantly.

complexity of the method is $O(S\,T\,N)$. However, the result is much less degenerate than that of the particle smoother of Kitagawa (1996). Under suitable conditions, it is also possible to reduce the number of computations to linear in the number of particles by implementing the backward simulation using rejection sampling (Douc et al., 2011).

Example 11.1 (Pendulum tracking with BS-PS) *The result of applying the backward-simulation smoother with 100 samples (with the bootstrap filter with 10 000 samples as the filter part) to the pendulum model in Example 5.1 is shown in Figure 11.1. The RMSE error* 0.028 *is smaller than the RMSE of the ERTSS, which was* 0.033*, and the same as the errors of the other Gaussian approximation based RTS smoothers, which all were* 0.028*. As already concluded in Example 7.1 the use of particle approximations is not really beneficial in this particular model.*

Example 11.2 (Cluttered pendulum tracking with BS-PS) *The result of applying the backward-simulation smoother with 100 samples (with the*

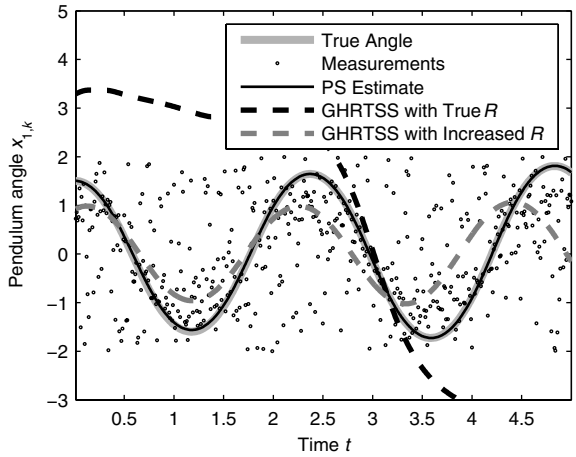


Figure 11.2 Smoothing results for cluttered pendulum tracking with particle smoother, a GHRTSS with no clutter model, and a GHRTSS with artificially increased process noise. The RMSE errors of the methods were 0.028, 3.36, and 0.74, respectively.

bootstrap filter with 10 000 samples as the filter part) to the cluttered pendulum model in Example 7.2 is shown in Figure 11.2. The RMSE error of the particle smoother was 0.028*. The RMSE of a GHRTSS without any clutter model was* 3.36 *and the RMSE of a GHRTSS with artificially increased measurement noise was* 0.74*. Thus in this case the particle smoother gives a significant improvement over the Gaussian approximation based smoothers.*

11.3 Reweighting particle smoother

The *reweighting particle smoother* of Hürzeler and Kunsch (1998) and Doucet et al. (2000), which is also called the *marginal particle smoother*, is based on computing new weights for the SIR filtering particles such that we get an approximation to the smoothing distribution.

Algorithm 11.3 (Reweighting particle smoother) *Given the weighted set of particles* $\{w_k^{(i)}, \mathbf{x}_k^{(i)} : i = 1, \ldots, N\}$ *representing the filtering distribution, we can form approximations to the marginal smoothing distributions as follows.*

- *Start by setting* $w_{T|T}^{(i)} = w_T^{(i)}$ *for* $i = 1, \ldots, N$.
- *For each* $k = T-1, \ldots, 0$ *compute new weights by*

$$w_{k|T}^{(i)} = \sum_j w_{k+1|T}^{(j)} \frac{w_k^{(i)}\, p(\mathbf{x}_{k+1}^{(j)} \mid \mathbf{x}_k^{(i)})}{\left[\sum_l w_k^{(l)}\, p(\mathbf{x}_{k+1}^{(j)} \mid \mathbf{x}_k^{(l)})\right]}. \tag{11.11}$$

At each step k *the marginal smoothing distribution can be approximated as*

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) \approx \sum_i w_{k|T}^{(i)}\, \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}). \tag{11.12}$$

Derivation Assume that we have already computed the weights for the following approximation, where the particles $\mathbf{x}_{k+1}^{(i)}$ are from the SIR filter:

$$p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \approx \sum_i w_{k+1|T}^{(i)}\, \delta(\mathbf{x}_{k+1} - \mathbf{x}_{k+1}^{(i)}). \tag{11.13}$$

The integral in the second of Equations (8.2) can be now approximated as

$$\begin{aligned}
&\int \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})}\, \mathrm{d}\mathbf{x}_{k+1} \\
&\quad \approx \int \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \sum_i \left[w_{k+1|T}^{(i)}\, \delta(\mathbf{x}_{k+1} - \mathbf{x}_{k+1}^{(i)}) \right] \mathrm{d}\mathbf{x}_{k+1} \\
&\quad = \sum_i w_{k+1|T}^{(i)} \frac{p(\mathbf{x}_{k+1}^{(i)} \mid \mathbf{x}_k)}{p(\mathbf{x}_{k+1}^{(i)} \mid \mathbf{y}_{1:k})}.
\end{aligned} \tag{11.14}$$

By using SIR filter approximation in Equation (7.31) we get the following approximation for the predicted distribution in the denominator:

$$p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) \approx \sum_j w_k^{(j)}\, p(\mathbf{x}_{k+1} \mid \mathbf{x}_k^{(j)}), \tag{11.15}$$

which gives

$$\begin{aligned}
&\int \frac{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})\, p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})}\, \mathrm{d}\mathbf{x}_{k+1} \\
&\qquad \approx \sum_i w_{k+1|T}^{(i)} \frac{p(\mathbf{x}_{k+1}^{(i)} \mid \mathbf{x}_k)}{\left[\sum_j w_k^{(j)}\, p(\mathbf{x}_{k+1}^{(i)} \mid \mathbf{x}_k^{(j)})\right]}.
\end{aligned} \tag{11.16}$$

By substituting the SIR filter approximation and the approximation above to the Bayesian optimal smoothing equation we get

$$\begin{aligned} & p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) \\ & \quad = p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \int \left[\frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \right] \mathrm{d}\mathbf{x}_{k+1} \\ & \quad \approx \sum_l w_k^{(l)}\, \delta(\mathbf{x}_k - \mathbf{x}_k^{(l)}) \sum_i w_{k+1|T}^{(i)} \frac{p(\mathbf{x}_{k+1}^{(i)} \mid \mathbf{x}_k^{(l)})}{\left[\sum_j w_k^{(j)}\, p(\mathbf{x}_{k+1}^{(i)} \mid \mathbf{x}_k^{(j)}) \right]}, \end{aligned} \tag{11.17}$$

where we can identify the weights as

$$w_{k|T}^{(l)} = \sum_i w_{k+1|T}^{(i)} \frac{w_k^{(l)}\, p(\mathbf{x}_{k+1}^{(i)} \mid \mathbf{x}_k^{(l)})}{\left[\sum_j w_k^{(j)}\, p(\mathbf{x}_{k+1}^{(i)} \mid \mathbf{x}_k^{(j)}) \right]}. \tag{11.18}$$

□

The computational complexity of the method is $O(T\,N^2)$, that is, the same as of the backward-simulation smoother with $S = N$ simulated trajectories.

11.4 Rao–Blackwellized particle smoothers

Rao–Blackwellized particle smoothers (RBPS) can be used for computing approximate smoothing solutions to the conditionally Gaussian models defined in Equation (7.36). A simple way to implement an RBPS is to store the histories instead of the single states in the RBPF, as in the case of the SIR particle smoother (Algorithm 11.1). The corresponding histories of the means and the covariances are then conditional on the *latent variable histories* $\mathbf{u}_{0:T}$. However, the means and covariances at time step k are only conditional on the *measurement histories* up to k, not on the later measurements. In order to correct this, RTS smoothers have to be applied to each history of the means and the covariances.

Algorithm 11.4 (Rao–Blackwellized SIR particle smoother) *A set of weighted samples* $\{w_T^{\mathrm{s},(i)}, \mathbf{u}_{0:T}^{\mathrm{s},(i)}, \mathbf{m}_{0:T}^{\mathrm{s},(i)}, \mathbf{P}_{0:T}^{\mathrm{s},(i)} : i = 1, \ldots, N\}$ *representing the smoothed distribution can be computed as follows.*

1 Compute the weighted set of Rao–Blackwellized state histories

$$\{w_T^{(i)}, \mathbf{u}_{0:T}^{(i)}, \mathbf{m}_{0:T}^{(i)}, \mathbf{P}_{0:T}^{(i)} : i = 1, \ldots, N\} \tag{11.19}$$

by storing histories in the Rao–Blackwellized particle filter analogously to the SIR particle smoother in Algorithm 11.1.

2 *Set*

$$w_T^{\mathrm{s},(i)} = w_T^{(i)},$$
$$\mathbf{u}_{0:T}^{\mathrm{s},(i)} = \mathbf{u}_{0:T}^{(i)}. \tag{11.20}$$

3 *Apply the RTS smoother to each of the mean and covariance histories* $\mathbf{m}_{0:T}^{(i)}, \mathbf{P}_{0:T}^{(i)}$ *for* $i = 1, \ldots, N$ *to produce the smoothed mean and covariance histories* $\mathbf{m}_{0:T}^{\mathrm{s},(i)}, \mathbf{P}_{0:T}^{\mathrm{s},(i)}$.

The Rao–Blackwellized particle smoother in this simple form also has the same disadvantage as the SIR particle smoother, that is, the smoothed estimate of $\mathbf{u}_k$ can be quite degenerate if $T \gg k$. Fortunately, the smoothed estimates of the actual states $\mathbf{x}_k$ can still be relatively good, because their degeneracy is avoided by Rao–Blackwellization.

To avoid the degeneracy in estimates of $\mathbf{u}_k$ it is possible to use better sampling procedures for generating samples from the smoothing distributions analogously to the plain particle smoothing. The backward-simulation has indeed been generalized to the Rao–Blackwellized case, but the implementation of the Rao–Blackwellized reweighting smoother seems to be quite problematic. The Rao–Blackwellized backward-simulation smoother (see Särkkä et al., 2012a) can be used for drawing backward trajectories from the marginal posterior of the latent variables $\mathbf{u}_k$ and the posterior of the conditionally Gaussian part is obtained via Kalman filtering and RTS smoothing. Another option is to simulate backward trajectories from the joint distribution of $(\mathbf{x}_k, \mathbf{u}_k)$ (Fong et al., 2002; Lindsten, 2011). However, this approach does not really lead to Rao–Blackwellized estimates of the smoothing distribution, because the Gaussian part of the state is sampled as well.

It is also possible to construct approximate Rao–Blackwellized backward-simulation smoothers by using Kim's approximation (see Kim, 1994; Barber, 2006; Särkkä et al., 2012a)

$$\int p(\mathbf{u}_k \mid \mathbf{u}_{k+1}, \mathbf{x}_{k+1}, \mathbf{y}_{1:k})\, p(\mathbf{x}_{k+1} \mid \mathbf{u}_{k+1}, \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x}_{k+1} \simeq p(\mathbf{u}_k \mid \mathbf{u}_{k+1}, \mathbf{y}_{1:k}). \tag{11.21}$$

The result is an algorithm where we first apply the backward-simulation smoother in the Algorithm 11.2 to the marginal samples of $\mathbf{u}_k$ alone, that is, we simply use $p(\mathbf{u}_{k+1} \mid \mathbf{u}_k)$ instead of $p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)$ in the algorithm. Given a trajectory of the non-Gaussian variable, the linear Gaussian

part may be recovered with a Kalman filter and RTS smoother. However, this procedure is only an approximation and does not lead to a true Rao–Blackwellized Monte Carlo representation of the smoothing distribution.

11.5 Exercises

11.1 Implement the backward-simulation smoother for the model in Exercise 5.1 and compare its performance to the Gaussian approximation based smoothers.

11.2 Implement the reweighting smoother for the model in Exercise 5.1 and compare its performance to the other smoothers.

11.3 Show that the latent variable sequence in conditionally Gaussian models is not Markovian in general in the sense that

$$p(\mathbf{u}_k \mid \mathbf{u}_{k+1}, \mathbf{y}_{1:T}) \neq p(\mathbf{u}_k \mid \mathbf{u}_{k+1}, \mathbf{y}_{1:k}) \tag{11.22}$$

when $T > k$, and thus simple backward smoothing in $\mathbf{u}_k$ does not lead to the correct result.

11.4 Implement the Rao–Blackwellized SIR particle smoother for the clutter model in Exercise 7.5.

11.5 Let's again consider the clutter model in Exercise 7.5. Assume that you have implemented the filter as a Rao–Blackwellized particle filter with resampling at every step (thus the weights are all equal). Write down the algorithm for the Kim's approximation based backward simulation smoother for the model. What peculiar property does the smoother have? Does this have something to do with the property in Equation (11.22)?

12

Parameter estimation

In the previous chapters we have assumed that the parameters of the state space model are known and only the state needs to be estimated. However, in practical models, the parameters are unknown as well. In this chapter we concentrate on three types of method for parameter estimation: optimization based methods for computing maximum a posteriori (MAP) or maximum likelihood (ML) estimates, expectation-maximization (EM) algorithms for computing the MAP or ML estimates, and Markov chain Monte Carlo (MCMC) methods for generating Monte Carlo approximations of the posterior distributions. We also show how Kalman filters and RTS smoothers, Gaussian filters and smoothers, as well as particle filters and smoothers can be used for approximating the marginal likelihoods, parameter posteriors, and other quantities needed by the methods.

12.1 Bayesian estimation of parameters in state space models

The Bayesian way of treating unknown parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ is to model them as random variables with a certain prior distribution $p(\boldsymbol{\theta})$. A state space model with unknown parameters can be written in the form

$$\begin{aligned} \boldsymbol{\theta} &\sim p(\boldsymbol{\theta}), \\ \mathbf{x}_0 &\sim p(\mathbf{x}_0 \mid \boldsymbol{\theta}), \\ \mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}), \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}). \end{aligned} \tag{12.1}$$

A straightforward way to proceed would now be to form the full posterior distribution via Bayes' rule:

$$p(\mathbf{x}_{0:T}, \boldsymbol{\theta} \mid \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{\theta})\, p(\mathbf{x}_{0:T} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathbf{y}_{1:T})}, \tag{12.2}$$

where the terms $p(\mathbf{x}_{0:T} \mid \boldsymbol{\theta})$ and $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T}, \boldsymbol{\theta})$ can be computed as

$$p(\mathbf{x}_{0:T} \mid \boldsymbol{\theta}) = p(\mathbf{x}_0 \mid \boldsymbol{\theta}) \prod_{k=1}^{T} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}),$$

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

If we are only interested in the parameters $\boldsymbol{\theta}$, the proper Bayesian way to proceed is to integrate the states out, which gives the marginal posterior of parameters:

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) = \int p(\mathbf{x}_{0:T}, \boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \, \mathrm{d}\mathbf{x}_{0:T}. \tag{12.3}$$

Unfortunately, computation of this high-dimensional integral is hard and becomes even harder as we obtain more measurements. In this approach we encounter the same computational problem as was discussed in Sections 1.3 and 4.1, which led us to consider optimal filtering and smoothing instead of the straightforward Bayesian approach. Thus it is again advantageous to look at recursive, filtering, and smoothing kinds of solution.

In this chapter we present methods for parameter estimation which are based on approximating the marginal posterior distribution

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \propto p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) \, p(\boldsymbol{\theta}), \tag{12.4}$$

without explicitly forming the joint posterior distribution of the states and parameters as in Equation (12.2). Instead, we present recursive algorithms for direct computation of the above distribution. For linear state space models, this can be done exactly, and in non-linear and non-Gaussian models we can use Gaussian filtering or particle filtering based approximations. Once we know how to evaluate the above distribution, we can estimate the parameters, for example, by finding their maximum a posteriori (MAP) estimate or by sampling from the posterior by Markov chain Monte Carlo (MCMC) methods. If the direct evaluation of the distribution is not feasible, we can use the expectation maximization (EM) algorithm for iteratively finding the ML or MAP estimate.

12.1.1 Parameter posterior and energy function

The difficult part in Equation (12.4) is the evaluation of the marginal likelihood $p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta})$. The prior distribution can usually be selected such that

it is easy to evaluate. Although evaluation of the normalization constant for the posterior distribution is a difficult problem, its evaluation is usually avoided by Bayesian computational methods and thus we do not need to worry about it.

The key to recursive computation of the parameter posterior in state space models is the following factorization (often called prediction error decomposition):

$$p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) = \prod_{k=1}^{T} p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}), \tag{12.5}$$

where we have denoted $p(\mathbf{y}_1 \mid \mathbf{y}_{1:0}, \boldsymbol{\theta}) \triangleq p(\mathbf{y}_1 \mid \boldsymbol{\theta})$ for notational convenience. Because each of the terms in the above product can be computed recursively, the whole marginal likelihood can be computed recursively as follows.

Theorem 12.1 (Recursion for marginal likelihood of parameters) *The marginal likelihood of parameters is given by Equation* (12.5)*, where the terms in the product can be computed recursively as*

$$p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) = \int p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta})\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})\, \mathrm{d}\mathbf{x}_k, \tag{12.6}$$

where $p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta})$ *is the measurement model and* $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})$ *is the predictive distribution of the state, which obeys the recursion*

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) &= \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})\, \mathrm{d}\mathbf{x}_{k-1} \\ p(\mathbf{x}_k \mid \mathbf{y}_{1:k}, \boldsymbol{\theta}) &= \frac{p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta})\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})}. \end{aligned} \tag{12.7}$$

Note that the latter equations are just the Bayesian filtering Equations (4.11) *and* (4.12)*, where we have explicitly written down the parameter dependence.*

Proof Due to the conditional independence of the measurements (Property 4.2) we have

$$\begin{aligned} p(\mathbf{y}_k, \mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) &= p(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1}, \boldsymbol{\theta})\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) \\ &= p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta})\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}). \end{aligned} \tag{12.8}$$

Integrating over $\mathbf{x}_k$ gives Equation (12.6). □

The marginal likelihood obtained via Theorem 12.1 can then be substituted into Equation (12.4) to give the marginal posterior distribution of the parameters. However, instead of working with marginal likelihood or marginal posterior explicitly, in parameter estimation, it is often convenient to define the unnormalized negative log-posterior or *energy function* as follows.

Definition 12.1 (Energy function)

$$\varphi_T(\boldsymbol{\theta}) = -\log p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}). \tag{12.9}$$

Remark 12.1 *The definition of energy function thus implies*

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \propto \exp(-\varphi_T(\boldsymbol{\theta})). \tag{12.10}$$

The energy function can be seen to obey the following simple recursion.

Theorem 12.2 (Recursion for energy function) *The energy function defined in Equation* (12.9) *can be evaluated recursively as follows.*

- *Start from* $\varphi_0(\theta) = -\log p(\boldsymbol{\theta})$.
- *At each step* $k = 1, 2, \ldots, T$ *compute the following:*

$$\varphi_k(\boldsymbol{\theta}) = \varphi_{k-1}(\boldsymbol{\theta}) - \log p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}), \tag{12.11}$$

  where the terms $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})$ *can be computed recursively by Theorem 12.1.*

Proof The result follows from substituting Equation (12.5) into the definition of the energy function in Equation (12.9) and identifying the terms corresponding to $\varphi_{k-1}(\boldsymbol{\theta})$. □

12.2 Computational methods for parameter estimation

In this section we briefly go through the underlying ideas in ML and MAP based parameter estimation, their implementation by direct optimization and by the EM algorithm, as well as the basics of Markov chain Monte Carlo (MCMC) methods. There exist many other parameter estimation methods for state space models and for more general statistical models, but here we concentrate on these, because these approaches are the most widely used (probabilistic methods) in the state space context.

12.2.1 Maximum a posteriori and Laplace approximations

The *maximum a posteriori (MAP)* estimate is obtained by determining the location of the maximum of the posterior distribution and using it as the point estimate:

$$\hat{\boldsymbol{\theta}}^{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} \left[p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \right]. \tag{12.12}$$

The MAP estimate can be equivalently computed as the minimum of the error function defined in Equation (12.9):

$$\hat{\boldsymbol{\theta}}^{\mathrm{MAP}} = \arg\min_{\boldsymbol{\theta}} \left[\varphi_T(\boldsymbol{\theta}) \right], \tag{12.13}$$

which is usually numerically more stable and easier to compute. The maximum likelihood (ML) estimate of the parameter is a MAP estimate with a formally uniform prior $p(\boldsymbol{\theta}) \propto 1$.

The minimum of the energy function can be computed by using various gradient-free or gradient based general optimization algorithms (see, e.g., Luenberger and Ye, 2008). However, to be able to use gradient based optimization we will need to evaluate the derivatives of the energy function as well. It is possible to find the derivatives in basically two ways (see, e.g., Cappé et al., 2005).

1 By formally differentiating the energy function recursion equations for a particular method. This results in so-called *sensitivity equations* which can be implemented as additional recursion equations computed along with filtering.
2 Using *Fisher's identity* which expresses the gradient of the energy function as an expectation of the derivative of the complete data log likelihood over the smoothing distribution. The advantage of this approach over direct differentiation is that there is no need for an additional recursion.

The disadvantage of the MAP-estimate is that it essentially approximates the posterior distribution with the Dirac delta function

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \simeq \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{\mathrm{MAP}}), \tag{12.14}$$

and thus ignores the spread of the distribution completely.

It is also possible to use a Laplace approximation (Gelman et al., 2004) which uses the second derivative (Hessian) of the energy function to form a Gaussian approximation to the posterior distribution:

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \simeq \mathrm{N}(\boldsymbol{\theta} \mid \hat{\boldsymbol{\theta}}^{\mathrm{MAP}}, [\mathbf{H}(\hat{\boldsymbol{\theta}}^{\mathrm{MAP}})]^{-1}), \tag{12.15}$$

where $\mathbf{H}(\hat{\boldsymbol{\theta}}^{\mathrm{MAP}})$ is the Hessian matrix evaluated at the MAP estimate. However, to implement the Laplace approximation, we need to have a method to compute (or approximate) the second order derivatives of the energy function.

12.2.2 Parameter inference via Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods (see, e.g., Liu, 2001; Brooks et al., 2011) are a class of algorithms for drawing random variables from a given distribution by simulating a Markov chain which has the desired distribution as its stationary distribution. The methods are particularly suited for simulating samples from Bayesian posterior distributions $p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T})$, because to implement the methods, we only need to know the unnormalized posterior distribution $\tilde{p}(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) = p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})$ or equivalently the energy function in Equation (12.9), and knowledge of the normalization constant of the posterior distribution is not required. The usage of MCMC methods in the state space context has been recently discussed, for example, by Ninness and Henriksen (2010) and Andrieu et al. (2010).

The *Metropolis–Hastings* (MH) algorithm is the most common type of MCMC method. MH uses a *proposal density* $q(\boldsymbol{\theta}^{(i)} \mid \boldsymbol{\theta}^{(i-1)})$ for suggesting new samples $\boldsymbol{\theta}^{(i)}$ given the previous ones $\boldsymbol{\theta}^{(i-1)}$. The algorithm is the following.

Algorithm 12.1 (Metropolis–Hastings) *The Metropolis–Hastings (MH) algorithm consists of the following steps.*

- *Draw the starting point, $\boldsymbol{\theta}^{(0)}$ from an arbitrary initial distribution.*
- *For $i = 1, 2, \ldots, N$ do*

  1 Sample a candidate point $\boldsymbol{\theta}^*$ from the proposal distribution:

$$\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)}). \tag{12.16}$$

  2 Evaluate the acceptance probability

$$\alpha_i = \min\left\{1, \exp(\varphi_T(\boldsymbol{\theta}^{(i-1)}) - \varphi_T(\boldsymbol{\theta}^*))\, \frac{q(\boldsymbol{\theta}^{(i-1)} \mid \boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(i-1)})}\right\}. \tag{12.17}$$

  3 Generate a uniform random variable $u \sim \mathrm{U}(0,1)$ and set

$$\boldsymbol{\theta}^{(i)} = \begin{cases} \boldsymbol{\theta}^*, & \text{if } u \leq \alpha_i, \\ \boldsymbol{\theta}^{(i-1)}, & \text{otherwise.} \end{cases} \tag{12.18}$$

The Metropolis algorithm is a commonly used special case of Metropolis–Hastings, where the proposal distribution is symmetric, $q(\boldsymbol{\theta}^{(i-1)} \mid \boldsymbol{\theta}^{(i)}) = q(\boldsymbol{\theta}^{(i)} \mid \boldsymbol{\theta}^{(i-1)})$. In this case the acceptance probability reduces to

$$\alpha_i = \min \left\{ 1, \exp(\varphi_T(\boldsymbol{\theta}^{(i-1)}) - \varphi_T(\boldsymbol{\theta}^*)) \right\}. \tag{12.19}$$

The choice of the proposal distribution is crucial for performance of the Metropolis–Hastings method and finding a good one is a hard task (see, e.g., Liu, 2001; Brooks et al., 2011). Some choices will lead to Metropolis–Hastings methods where the samples are highly correlated, whereas with some choices the rejection rate becomes too high.

One commonly used choice is to use Gaussian distribution as the proposal distribution,

$$q(\boldsymbol{\theta}^{(i)} \mid \boldsymbol{\theta}^{(i-1)}) = \mathrm{N}(\boldsymbol{\theta}^{(i)} \mid \boldsymbol{\theta}^{(i-1)}, \boldsymbol{\Sigma}_{i-1}), \tag{12.20}$$

where $\boldsymbol{\Sigma}_{i-1}$ is some suitable covariance matrix. The resulting algorithm is called the random walk Metropolis algorithm, because the transition distribution above defines a Gaussian random walk in parameter space. With this selection of proposal distribution the challenge is now to find a suitable covariance matrix for the random walk.

One approach to the problem of selection of the covariance matrix is to use *adaptive Markov chain Monte Carlo (AMCMC)* methods where the covariance of the Gaussian proposal in the Metropolis algorithm is automatically adapted during the MCMC run (see, e.g., Haario et al., 1999, 2001; Andrieu and Thoms, 2008; Vihola, 2012). A typical idea in AMCMC methods is to use the covariance of the previously generated samples as an estimate of the actual covariance of the posterior distribution. Given the covariance, it is possible to compute the covariance of the proposal distribution such that it causes an acceptance rate $\bar{\alpha}_*$ which is optimal in some suitable sense. For the random walk Metropolis algorithm, the optimal acceptance rate in certain ideal conditions is $\bar{\alpha}_* = 0.234$ (Roberts and Rosenthal, 2001).

For example, the *robust adaptive Metropolis (RAM)* algorithm of Vihola (2012) is similar to the adaptive Metropolis (AM) algorithm of Haario et al. (2001) except that the adaptation of the covariance $\boldsymbol{\Sigma}_i$ is done in a slightly different way. The algorithm is the following.

Algorithm 12.2 (RAM algorithm) *The RAM algorithm consists of the following steps.*

1 *Draw* $\boldsymbol{\theta}^{(0)}$ *from an initial distribution* $p_0(\boldsymbol{\theta})$*, and initialize* $\mathbf{S}_0$ *to be the lower-triangular Cholesky factor of the initial covariance* $\boldsymbol{\Sigma}_0$.
2 *Sample a candidate point by* $\boldsymbol{\theta}^* = \boldsymbol{\theta}_{i-1} + \mathbf{S}_{i-1}\,\mathbf{r}_i$*, where* $\mathbf{r}_i \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$.
3 *Compute the acceptance probability*

$$\alpha_i = \min\{1,\, \exp\left(\varphi(\boldsymbol{\theta}_{i-1}) - \varphi(\boldsymbol{\theta}_*)\right)\}. \tag{12.21}$$

4 *Sample a uniform random variable* u *from the uniform distribution* $\mathrm{U}(0, 1)$.
5 *If* $u \leq \alpha_i$*, set* $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^*$*. Otherwise set* $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)}$.
6 *Compute a lower-triangular matrix* $\mathbf{S}_i$ *with positive diagonal elements satisfying the equation*

$$\mathbf{S}_i\,\mathbf{S}_i^{\mathsf{T}} = \mathbf{S}_{i-1}\left(\mathbf{I} + \eta_i\;(\alpha_i - \bar{\alpha}_*)\,\frac{\mathbf{r}_i\,\mathbf{r}_i^{\mathsf{T}}}{||\mathbf{r}_i||^2}\right)\mathbf{S}_{i-1}^{\mathsf{T}}, \tag{12.22}$$

where $\{\eta_i\}_{i\geq 1} \subset (0, 1]$ *is an adaptation step size sequence decaying to zero. Although any such sequence will do, Vihola (2012) suggests* $\eta_i = i^{-\gamma}$ *with a suitable exponent* $\gamma \in (1/2, 1]$.
7 *Set* $i \leftarrow i + 1$ *and go to step 2 until the desired number of samples has been generated.*

Instead of the random walk Metropolis algorithm with covariance adaptation it is also possible to use the gradient information in the construction of the proposal distribution. This is the idea used in the Hamiltonian Monte Carlo (HMC) or hybrid Monte Carlo (HMC) method (Duane et al., 1987; Neal, 2011). In HMC the proposal distribution is constructed by simulating a physical system consisting of particles moving under the influence of a potential (the energy function) and heat bath. The gradient of the energy function enters the equations as the force caused by the potential. The HMC method was recently applied in the state space context by Mbalawata et al. (2013).

Another commonly used MCMC method is Gibbs' sampling (see, e.g., Liu, 2001; Brooks et al., 2011), which samples components of the parameters one at a time from their conditional distributions given the other parameters. The advantage of this method is that no rejections are needed, the acceptance probability is identically one. However, in order to implement the method it is necessary to be able to generate samples from the conditional distributions of parameters, which is possible only in a restricted class of models. For various other methods the reader is referred to Brooks et al. (2011).

12.2.3 Expectation maximization

The expectation-maximization (EM) algorithm is a method to iteratively find an ML estimate of the parameters when the direct optimization of the posterior distribution (or equivalently, energy function) is not feasible. The algorithm was originally introduced by Dempster et al. (1977) and applications to state space models have been discussed, for example, in Shumway and Stoffer (1982); Roweis and Ghahramani (2001); Schön et al. (2011). Gaussian smoothing and sigma-point based approximations in an EM context have also been recently discussed in Väänänen (2012). Although the EM algorithm was originally an algorithm for computing ML estimates, it can also be easily modified for computation of MAP estimates, as discussed below.

The EM algorithm is based on the result that even when we cannot evaluate the marginal likelihood as such, we are still often able to compute a lower bound for it as follows. Let $q(\mathbf{x}_{0:T})$ be an arbitrary probability density over the states, then we have

$$\log p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) \geq F[q(\mathbf{x}_{0:T}), \boldsymbol{\theta}], \tag{12.23}$$

where the functional F is defined as

$$F[q(\mathbf{x}_{0:T}), \boldsymbol{\theta}] = \int q(\mathbf{x}_{0:T}) \log \frac{p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \boldsymbol{\theta})}{q(\mathbf{x}_{0:T})} \, \mathrm{d}\mathbf{x}_{0:T}. \tag{12.24}$$

The key idea behind the EM algorithm is that it is possible to maximize the left-hand side of Equation (12.23) by iteratively maximizing the lower bound $F[q(\mathbf{x}_{0:T}), \boldsymbol{\theta}]$. A simple way to do that is the following iteration (Neal and Hinton, 1999).

Algorithm 12.3 (Abstract EM) *The maximization of the lower bound in Equation* (12.24) *can be done by coordinate ascend as follows.*

- *Start from initial guesses* $q^{(0)}$, $\boldsymbol{\theta}^{(0)}$.
- *For* $n = 0, 1, 2, \ldots$ *do the following steps:*
  1. E-step: *Find* $q^{(n+1)} = \arg\max_q F[q, \boldsymbol{\theta}^{(n)}]$.
  2. M-step: *Find* $\boldsymbol{\theta}^{(n+1)} = \arg\max_{\boldsymbol{\theta}} F[q^{(n+1)}, \boldsymbol{\theta}]$.

In order to implement the EM algorithm we need to be able to do the above maximizations in practice. Fortunately, it can be shown (see, e.g., Neal and Hinton, 1999) that the result of the maximization at the E-step is

$$q^{(n+1)}(\mathbf{x}_{0:T}) = p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}). \tag{12.25}$$

Plugging this into the expression of $F[q^{(n+1)}(\mathbf{x}_{0:T}), \boldsymbol{\theta}]$ gives

$$\begin{aligned} &F[q^{(n+1)}(\mathbf{x}_{0:T}), \boldsymbol{\theta}] \\ &= \int p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \log p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \boldsymbol{\theta}) \, \mathrm{d}\mathbf{x}_{0:T} \\ &- \int p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \log p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \, \mathrm{d}\mathbf{x}_{0:T}. \end{aligned} \tag{12.26}$$

Because the latter term does not depend on $\boldsymbol{\theta}$, maximizing $F[q^{(n+1)}, \boldsymbol{\theta}]$ is equivalent to maximizing the first term above, which in the EM context is commonly denoted as

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) = \int p(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \log p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \boldsymbol{\theta}) \, \mathrm{d}\mathbf{x}_{0:T}, \tag{12.27}$$

which is thus the expectation of the logarithm of the complete data likelihood $p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \boldsymbol{\theta})$ over the full joint posterior of the states given the parameters $\boldsymbol{\theta}^{(n)}$. The resulting algorithm is the following.

Algorithm 12.4 (EM algorithm) *The EM algorithm consists of the following steps.*

- *Start from an initial guess* $\boldsymbol{\theta}^{(0)}$.
- *For* $n = 0, 1, 2, \ldots$ *do the following steps:*
  1. E-step: *compute* $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})$.
  2. M-step: *compute* $\boldsymbol{\theta}^{(n+1)} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})$.

Due to the Markovian structure of the state space model in Equation (12.1), the complete data log-likelihood has the form

$$\begin{aligned} &\log p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \boldsymbol{\theta}) \\ &= \log p(\mathbf{x}_0 \mid \boldsymbol{\theta}) + \sum_{k=1}^{T} \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}) + \sum_{k=1}^{T} \log p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}). \end{aligned} \tag{12.28}$$

The expression for $\mathcal{Q}$ in Equation (12.27) and thus the E-step in Algorithm 12.4 now reduces to computation of (see Schön et al., 2011)

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) = I_1(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) + I_2(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) + I_3(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}), \tag{12.29}$$

where

$$I_1(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) = \int p(\mathbf{x}_0 \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \log p(\mathbf{x}_0 \mid \boldsymbol{\theta}) \, \mathrm{d}\mathbf{x}_0,$$

$$I_2(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) = \sum_{k=1}^{T} \int p(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \times \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}) \, \mathrm{d}\mathbf{x}_k \, \mathrm{d}\mathbf{x}_{k-1},$$

$$I_3(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) = \sum_{k=1}^{T} \int p(\mathbf{x}_k \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \log p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}) \, \mathrm{d}\mathbf{x}_k. \quad (12.30)$$

The above expectations are over the smoothing distribution and the key thing is to observe that we do not need to compute expectations over the full posterior, but only over the smoothing distributions $p(\mathbf{x}_k \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)})$ and pairwise smoothing distributions $p(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)})$. It turns out that the required expectations can be easily (approximately) evaluated using smoother results. In the case of linear state space models we can find a closed form expression for the above integrals in terms of RTS smoother results. In the non-linear case we can approximate the integrals by using non-linear smoothers such as Gaussian smoothers. In the more general probabilistic state space model we can use particle smoothers to approximate them.

On the E-step of Algorithm 12.4 we need to maximize the expression for $\mathcal{Q}$ in Equation (12.29) with respect to the parameters $\boldsymbol{\theta}$. In principle, we can utilize various gradient-free and gradient based optimization methods (see, e.g., Luenberger and Ye, 2008) for doing that, but the most useful case occurs when we can do the maximization analytically via setting the gradient to zero:

$$\frac{\partial \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})}{\partial \boldsymbol{\theta}} = 0. \quad (12.31)$$

This happens, for example, when estimating the parameters of linear state space models and in certain classes of non-linear state space models.

It turns out that we can calculate MAP estimates using the EM algorithm by replacing $p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} \mid \boldsymbol{\theta})$ in Equation (12.27) with $p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T}, \boldsymbol{\theta})$. In practice, it can be implemented by maximizing $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) + \log p(\boldsymbol{\theta})$ at the M-step instead of the plain $\mathcal{Q}$.

As a side product of the EM formulation above we also get a method to compute the gradient of the energy function needed in gradient based optimization for finding the MAP or ML estimates. *Fisher's identity* (see,

e.g., Cappé et al., 2005) states that if we evaluate the gradient of $\mathcal{Q}$ at $\boldsymbol{\theta}^{(n)} = \boldsymbol{\theta}$, we get exactly the gradient of the marginal log-likelihood. This implies that the gradient of the energy function can be evaluated as

$$\frac{\partial \varphi_T(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{\partial \log p(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \left.\frac{\partial \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}^{(n)}=\boldsymbol{\theta}}. \tag{12.32}$$

Note that here we refer to the above identity as Fisher's identity although the original identity is the relationship with the log marginal likelihood and $\mathcal{Q}$, not with the log posterior and $\mathcal{Q}$. In any case this identity is useful in linear state space models, because it is often easier to compute and computationally lighter (Segal and Weinstein, 1989; Olsson et al., 2007). However, in non-linear state space models it is not as useful, because the approximations involved in computation of the filtering and smoothing solutions often cause the gradient to have different approximations from the energy function approximation implied by the same method. That is, the gradient approximation computed with Fisher's identity and Gaussian smoothing might not exactly match the gradient of the energy function approximation computed with the corresponding Gaussian filter. However, in the case of particle filters, Fisher's identity provides a feasible way to approximate the gradient of the energy function.

12.3 Practical parameter estimation in state space models

In this section we discuss practical parameter estimation methods for state space models using linear Kalman filters and RTS smoothers, Gaussian approximation based non-linear Kalman filters and RTS smoothers, and particle filters and smoothers. But before going to them, we outline the simple but sometimes effective state augmentation approach.

12.3.1 State augmentation approach

Before going to more elaborate parameter estimation methods for state space models, we recall that already in Chapter 3 we used the Kalman filter for estimating static parameters in a regression model. The same approach can be generalized to the *state augmentation approach* which simply means that we augment the parameter as part of the state. For example, let's say that we have a non-linear model with unknown parameters $\boldsymbol{\theta}$:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}) + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}) + \mathbf{r}_k. \end{aligned} \tag{12.33}$$

We can now rewrite the model as

$$\begin{aligned}\boldsymbol{\theta}_k &= \boldsymbol{\theta}_{k-1},\\ \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}_{k-1}) + \mathbf{q}_{k-1},\\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}_k) + \mathbf{r}_k,\end{aligned} \tag{12.34}$$

where the dynamic model for the parameter essentially says that it is constant. If we now redefine the state as $\tilde{\mathbf{x}}_k = (\mathbf{x}_k, \boldsymbol{\theta}_k)$, we get a state space model of the form

$$\begin{aligned}\tilde{\mathbf{x}}_k &= \tilde{\mathbf{f}}(\tilde{\mathbf{x}}_{k-1}) + \tilde{\mathbf{q}}_{k-1},\\ \mathbf{y}_k &= \mathbf{h}(\tilde{\mathbf{x}}_k) + \mathbf{r}_k,\end{aligned} \tag{12.35}$$

which does not contain any unknown parameter anymore. The problem in this *state augmentation* is the singularity of the dynamic model for the parameter. It works well when the whole system is linear and we do not have any approximation errors in the estimator. If the parameters appear linearly in the system, it sometimes is a good idea to include the parameters as part of the state. However, this might fail sometimes as well.

With approximate non-linear filters the singularity of the parameter dynamic model indeed causes problems. With non-linear Kalman filters the Gaussian approximation tends to become singular which causes the filter to diverge. As discussed in Section 7.4, particle filters have a problem with small noises in the dynamic model because it causes sample impoverishment. As the noise in the dynamic model above is exactly zero, this case is particularly problematic for particle filters.

A common way to circumvent the problem is to introduce an artificial noise to the dynamic model of the parameter, that is, replace $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1}$ with

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{\varepsilon}_{k-1}, \tag{12.36}$$

where $\boldsymbol{\varepsilon}_{k-1}$ is a "small" noise process. But the problem is that we are no longer solving the original parameter estimation problem, but another one with a time-varying parameter. Anyway, this approach is sometimes applicable and should be considered before jumping into more complicated parameter estimation methods.

There is also a form of Rao–Blackwellization that sometimes helps. This approach is discussed in Section 12.3.5 and the idea is to use a closed form solution for the static parameter ("Rao–Blackwellize" the parameter) and sample only the original state part. This works if the parameter appears in the model in a suitable conjugate form.

12.3.2 Parameter estimation in linear state space models

Consider the following linear Gaussian state space model with unknown parameters $\boldsymbol{\theta}$:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}(\boldsymbol{\theta})\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{H}(\boldsymbol{\theta})\,\mathbf{x}_k + \mathbf{r}_k, \end{aligned} \tag{12.37}$$

where $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}(\boldsymbol{\theta}))$, $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}(\boldsymbol{\theta}))$, and $\mathbf{x}_0 \sim \mathrm{N}(\mathbf{m}_0(\boldsymbol{\theta}), \mathbf{P}_0(\boldsymbol{\theta}))$. In the above model, for notational convenience, we have assumed that the model matrices do not explicitly depend on time. The energy function and thus the marginal parameter posterior for the linear Gaussian model above can be obtained as follows.

Theorem 12.3 (Energy function for linear Gaussian model) *The recursion for the energy function is given as*

$$\varphi_k(\boldsymbol{\theta}) = \varphi_{k-1}(\boldsymbol{\theta}) + \frac{1}{2}\log|2\pi\,\mathbf{S}_k(\boldsymbol{\theta})| + \frac{1}{2}\mathbf{v}_k^{\mathsf{T}}(\boldsymbol{\theta})\,\mathbf{S}_k^{-1}(\boldsymbol{\theta})\,\mathbf{v}_k(\boldsymbol{\theta}), \tag{12.38}$$

where the terms $\mathbf{v}_k(\boldsymbol{\theta})$ and $\mathbf{S}_k(\boldsymbol{\theta})$ are given by the Kalman filter with the parameters fixed to $\boldsymbol{\theta}$.

- Prediction:

$$\begin{aligned} \mathbf{m}_k^-(\boldsymbol{\theta}) &= \mathbf{A}(\boldsymbol{\theta})\,\mathbf{m}_{k-1}(\boldsymbol{\theta}), \\ \mathbf{P}_k^-(\boldsymbol{\theta}) &= \mathbf{A}(\boldsymbol{\theta})\,\mathbf{P}_{k-1}(\boldsymbol{\theta})\,\mathbf{A}^{\mathsf{T}}(\boldsymbol{\theta}) + \mathbf{Q}(\boldsymbol{\theta}). \end{aligned} \tag{12.39}$$

- Update:

$$\begin{aligned} \mathbf{v}_k(\boldsymbol{\theta}) &= \mathbf{y}_k - \mathbf{H}(\boldsymbol{\theta})\,\mathbf{m}_k^-(\boldsymbol{\theta}), \\ \mathbf{S}_k(\boldsymbol{\theta}) &= \mathbf{H}(\boldsymbol{\theta})\,\mathbf{P}_k^-(\boldsymbol{\theta})\,\mathbf{H}^{\mathsf{T}}(\boldsymbol{\theta}) + \mathbf{R}(\boldsymbol{\theta}), \\ \mathbf{K}_k(\boldsymbol{\theta}) &= \mathbf{P}_k^-(\boldsymbol{\theta})\,\mathbf{H}^{\mathsf{T}}(\boldsymbol{\theta})\,\mathbf{S}_k^{-1}(\boldsymbol{\theta}), \\ \mathbf{m}_k(\boldsymbol{\theta}) &= \mathbf{m}_k^-(\boldsymbol{\theta}) + \mathbf{K}_k(\boldsymbol{\theta})\,\mathbf{v}_k(\boldsymbol{\theta}), \\ \mathbf{P}_k(\boldsymbol{\theta}) &= \mathbf{P}_k^-(\boldsymbol{\theta}) - \mathbf{K}_k(\boldsymbol{\theta})\,\mathbf{S}_k(\boldsymbol{\theta})\,\mathbf{K}_k^{\mathsf{T}}(\boldsymbol{\theta}). \end{aligned} \tag{12.40}$$

Proof The Kalman filter gives us the Gaussian predictive distribution $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-(\boldsymbol{\theta}), \mathbf{P}_k^-(\boldsymbol{\theta}))$ which via Theorem 12.1 thus gives

$$\begin{aligned} &p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) \\ &= \int \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}(\boldsymbol{\theta})\,\mathbf{x}_k, \mathbf{R}(\boldsymbol{\theta}))\ \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-(\boldsymbol{\theta}), \mathbf{P}_k^-(\boldsymbol{\theta}))\ \mathrm{d}\mathbf{x}_k \\ &= \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}(\boldsymbol{\theta})\,\mathbf{m}_k^-(\boldsymbol{\theta}), \mathbf{H}(\boldsymbol{\theta})\,\mathbf{P}_k^-(\boldsymbol{\theta})\,\mathbf{H}^{\mathsf{T}}(\boldsymbol{\theta}) + \mathbf{R}(\boldsymbol{\theta})). \end{aligned} \tag{12.41}$$

The rest follows from Theorem 12.2. □

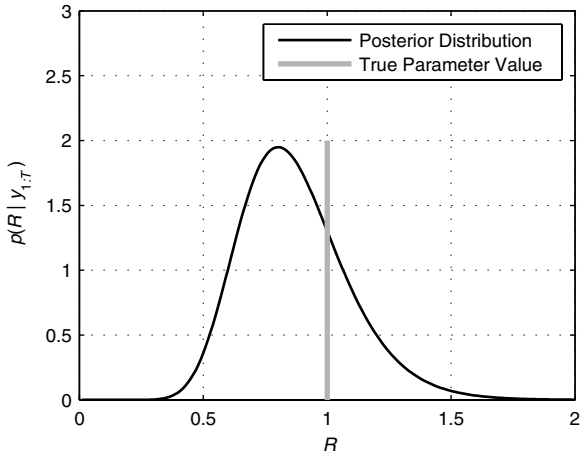


Figure 12.1 Posterior distribution of noise variance R in the Gaussian random walk model (see Example 4.1).

Thus if we fix $\boldsymbol{\theta}$ and run the above algorithm from $\varphi_0(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta})$ at $k = 0$ to the step $k = T$, then the full energy function is $\varphi_T(\boldsymbol{\theta})$. That is, the marginal posterior density at the point $\boldsymbol{\theta}$ can be evaluated up to a normalization constant by Equation (12.10) as

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T}) \propto \exp(-\varphi_T(\boldsymbol{\theta})).$$

Given the energy function it is now easy to implement, for example, a Metropolis–Hastings based MCMC sampler for generating a Monte Carlo approximation of the posterior distribution, or to use the energy function in a gradient-free optimization for finding the ML or MAP estimates of the parameters.

Example 12.1 (Parameter posterior for Gaussian random walk) *The posterior distribution of the noise variance $p(R \mid y_{1:T})$ for the Gaussian random walk model in Example 4.1 is shown in Figure 12.1. A formally uniform prior $p(R) \propto 1$ was assumed. The true value used in the simulation is indeed well within the high density area of the posterior distribution. However, it can also been seen that if we computed the MAP (or equivalently ML) estimate of the parameter, we would get a smaller value than the true one.*

In order to implement a gradient based optimization method, we need to have the gradient of the energy function as well. One way to implement the gradient computation is by first differentiating the energy function expression in Theorem 12.3 term-by-term and then each of the Kalman filter equations. This results in a recursion called *sensitivity equations* (Gupta and Mehra, 1974; Cappé et al., 2005) which can be evaluated along with the Kalman filtering computations. The equations are given in Theorem A.2 in Section A.3. Another way to compute the gradient is by using Fisher's identity (12.32), but before going into that, let's take a look at the EM algorithm for linear Gaussian models.

Recall that for implementing the EM algorithm we need to be able to compute the expectations in Equations (12.30), which in terms requires the knowledge of the smoothing distributions and pairwise smoothing distributions. Fortunately, by Equations (8.5) and (8.12) we know that

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^{\mathrm{s}}, \mathbf{P}_k^{\mathrm{s}}),$$

$$\begin{aligned} &p(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(n)}) \\ &\quad = \mathrm{N}\left(\begin{pmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \end{pmatrix} \,\middle|\, \begin{pmatrix} \mathbf{m}_k^{\mathrm{s}} \\ \mathbf{m}_{k-1}^{\mathrm{s}} \end{pmatrix}, \begin{pmatrix} \mathbf{P}_k^{\mathrm{s}} & \mathbf{P}_k^{\mathrm{s}} \mathbf{G}_{k-1}^{\mathsf{T}} \\ \mathbf{G}_{k-1} \mathbf{P}_k^{\mathrm{s}} & \mathbf{P}_{k-1}^{\mathrm{s}} \end{pmatrix}\right), \end{aligned} \tag{12.42}$$

where the means, covariances, and gains are computed with an RTS smoother with the model parameters fixed to $\boldsymbol{\theta}^{(n)}$. Note that in the EM algorithms appearing in the literature the cross term $\mathbf{P}_k^{\mathrm{s}} \mathbf{G}_{k-1}^{\mathsf{T}}$ is sometimes computed with a separate recursion (see, e.g., Shumway and Stoffer, 1982), but in fact it is unnecessary due to the above. The required expectations for EM in Equations (12.30) can now be computed in closed form and the result is the following (see Shumway and Stoffer, 1982).

Theorem 12.4 (Evaluation of $\mathcal{Q}$ for linear Gaussian model) *The expression for $\mathcal{Q}$ for the linear Gaussian model in Equation* (12.37) *can be*

written as

$$\begin{aligned}
&\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) \\
&= -\frac{1}{2}\log|2\pi\,\mathbf{P}_0(\boldsymbol{\theta})| - \frac{T}{2}\log|2\pi\,\mathbf{Q}(\boldsymbol{\theta})| - \frac{T}{2}\log|2\pi\,\mathbf{R}(\boldsymbol{\theta})| \\
&- \frac{1}{2}\,\mathrm{tr}\left\{\mathbf{P}_0^{-1}(\boldsymbol{\theta})\left[\mathbf{P}_0^{\mathrm{s}} + (\mathbf{m}_0^{\mathrm{s}} - \mathbf{m}_0(\boldsymbol{\theta}))\,(\mathbf{m}_0^{\mathrm{s}} - \mathbf{m}_0(\boldsymbol{\theta}))^{\mathsf{T}}\right]\right\} \\
&- \frac{T}{2}\,\mathrm{tr}\left\{\mathbf{Q}^{-1}(\boldsymbol{\theta})\left[\boldsymbol{\Sigma} - \mathbf{C}\,\mathbf{A}^{\mathsf{T}}(\boldsymbol{\theta}) - \mathbf{A}(\boldsymbol{\theta})\,\mathbf{C}^{\mathsf{T}} + \mathbf{A}(\boldsymbol{\theta})\,\boldsymbol{\Phi}\,\mathbf{A}^{\mathsf{T}}(\boldsymbol{\theta})\right]\right\} \\
&- \frac{T}{2}\,\mathrm{tr}\left\{\mathbf{R}^{-1}(\boldsymbol{\theta})\left[\mathbf{D} - \mathbf{B}\,\mathbf{H}^{\mathsf{T}}(\boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\theta})\,\mathbf{B}^{\mathsf{T}} + \mathbf{H}(\boldsymbol{\theta})\,\boldsymbol{\Sigma}\,\mathbf{H}^{\mathsf{T}}(\boldsymbol{\theta})\right]\right\},
\end{aligned} \tag{12.43}$$

where the following quantities are computed from the results of RTS smoothers run with parameter values $\boldsymbol{\theta}^{(n)}$*:*

$$\begin{aligned}
\boldsymbol{\Sigma} &= \frac{1}{T}\sum_{k=1}^{T} \mathbf{P}_k^{\mathrm{s}} + \mathbf{m}_k^{\mathrm{s}}\,[\mathbf{m}_k^{\mathrm{s}}]^{\mathsf{T}}, \\
\boldsymbol{\Phi} &= \frac{1}{T}\sum_{k=1}^{T} \mathbf{P}_{k-1}^{\mathrm{s}} + \mathbf{m}_{k-1}^{\mathrm{s}}\,[\mathbf{m}_{k-1}^{\mathrm{s}}]^{\mathsf{T}}, \\
\mathbf{B} &= \frac{1}{T}\sum_{k=1}^{T} \mathbf{y}_k\,[\mathbf{m}_k^{\mathrm{s}}]^{\mathsf{T}}, \\
\mathbf{C} &= \frac{1}{T}\sum_{k=1}^{T} \mathbf{P}_k^{\mathrm{s}}\,\mathbf{G}_{k-1}^{\mathsf{T}} + \mathbf{m}_k^{\mathrm{s}}\,[\mathbf{m}_{k-1}^{\mathrm{s}}]^{\mathsf{T}}, \\
\mathbf{D} &= \frac{1}{T}\sum_{k=1}^{T} \mathbf{y}_k\,\mathbf{y}_k^{\mathsf{T}}.
\end{aligned} \tag{12.44}$$

The usefulness of the EM algorithm for linear state space models stems from the fact that if the parameters are selected to be some of the full model matrices (or initial mean) we can indeed perform the M-step of the EM algorithm in closed form. The same thing happens if the parameters appear linearly in one of the model matrices (e.g., are some subcomponents of the matrices), but application of the EM algorithm to the estimation of the full matrices is the classical result. By setting the gradients of $\partial\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})/\partial\boldsymbol{\theta}$ to zero for each $\boldsymbol{\theta} = \{\mathbf{A}, \mathbf{H}, \mathbf{Q}, \mathbf{R}, \mathbf{P}_0, \mathbf{m}_0\}$ separately, we get the following result.

Theorem 12.5 (Maximization of $\mathcal{Q}$ for linear model parameters) *The maximum* $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})$*, when the parameters are selected to be one of the model parameters* $\boldsymbol{\theta} \in \{\mathbf{A}, \mathbf{H}, \mathbf{Q}, \mathbf{R}, \mathbf{P}_0, \mathbf{m}_0\}$*, can be computed as follows.*

- *When* $\boldsymbol{\theta} = \mathbf{P}_0$ *we get*

$$\mathbf{P}_0^* = \mathbf{P}_0^s + (\mathbf{m}_0^s - \mathbf{m}_0)(\mathbf{m}_0^s - \mathbf{m}_0)^\mathsf{T}. \tag{12.45}$$

- *When* $\boldsymbol{\theta} = \mathbf{A}$ *we get*

$$\mathbf{A}^* = \mathbf{C}\,\boldsymbol{\Phi}^{-1}. \tag{12.46}$$

- *When* $\boldsymbol{\theta} = \mathbf{Q}$ *we get*

$$\mathbf{Q}^* = \boldsymbol{\Sigma} - \mathbf{C}\,\mathbf{A}^\mathsf{T} - \mathbf{A}\,\mathbf{C}^\mathsf{T} + \mathbf{A}\,\boldsymbol{\Phi}\,\mathbf{A}^\mathsf{T}. \tag{12.47}$$

- *When* $\boldsymbol{\theta} = \mathbf{H}$ *we get*

$$\mathbf{H}^* = \mathbf{B}\,\boldsymbol{\Sigma}^{-1}. \tag{12.48}$$

- *When* $\boldsymbol{\theta} = \mathbf{R}$ *we get*

$$\mathbf{R}^* = \mathbf{D} - \mathbf{H}\,\mathbf{B}^\mathsf{T} - \mathbf{B}\,\mathbf{H}^\mathsf{T} + \mathbf{H}\,\boldsymbol{\Sigma}\,\mathbf{H}^\mathsf{T}. \tag{12.49}$$

- *Finally, the maximum with respect to the initial mean* $\boldsymbol{\theta} = \mathbf{m}_0$ *is*

$$\mathbf{m}_0^* = \mathbf{m}_0^s. \tag{12.50}$$

Obviously the above theorem can also be used for solving the maximum of $\mathcal{Q}$ with respect to any subset of model matrices by solving the resulting equations jointly. The EM algorithm for finding the maximum likelihood estimates of the linear state space model parameters is now the following.

Algorithm 12.5 (EM algorithm for linear state space models) *Let* $\boldsymbol{\theta}$ *contain some subset of the model parameters* $\{\mathbf{A}, \mathbf{H}, \mathbf{Q}, \mathbf{R}, \mathbf{P}_0, \mathbf{m}_0\}$*. We can find maximum likelihood estimates of them via the following iteration.*

- *Start from some initial guess* $\boldsymbol{\theta}^{(0)}$*.*
- *For* $n = 0, 1, 2, \ldots$ *do the following steps.*
  1 E-step: *Run the RTS smoother using the current parameter values in* $\boldsymbol{\theta}^{(n)}$ *and compute the quantities in Equation* (12.44) *from the smoother results.*
  2 M-step: *Find new parameters values by using Equations* (12.45) – (12.50) *and store them in* $\boldsymbol{\theta}^{(n+1)}$*.*

The expression for $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})$ also provides an "easy gradient recipe" (Olsson et al., 2007) for computation of the energy function gradient via Fisher's identity (Equation (12.32)), as it enables the computation of the gradient without an additional recursion (sensitivity equations). The resulting expression is given in Theorem A.3 in Section A.3.

12.3.3 Parameter estimation with Gaussian filtering and smoothing

One way to implement parameter estimation in non-linear models is by replacing the Kalman filters and RTS smoothers used in the linear case with their non-linear counterparts. Let's consider parameter estimation in models of the form

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}) + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}) + \mathbf{r}_k, \end{aligned} \tag{12.51}$$

where $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}(\boldsymbol{\theta}))$, $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}(\boldsymbol{\theta}))$, and $\mathbf{x}_0 \sim \mathrm{N}(\mathbf{m}_0(\boldsymbol{\theta}), \mathbf{P}_0(\boldsymbol{\theta}))$. The energy function can now be approximated with the following Gaussian filtering based algorithm.

Algorithm 12.6 (Gaussian filtering based energy function) *The recursion for the approximate energy function is*

$$\varphi_k(\boldsymbol{\theta}) \simeq \varphi_{k-1}(\boldsymbol{\theta}) + \frac{1}{2}\log|2\pi\,\mathbf{S}_k(\boldsymbol{\theta})| + \frac{1}{2}\mathbf{v}_k^{\mathsf{T}}(\boldsymbol{\theta})\,\mathbf{S}_k^{-1}(\boldsymbol{\theta})\,\mathbf{v}_k(\boldsymbol{\theta}), \tag{12.52}$$

where the terms $\mathbf{v}_k(\boldsymbol{\theta})$ and $\mathbf{S}_k(\boldsymbol{\theta})$ are given by the Gaussian filter with the parameters fixed to $\boldsymbol{\theta}$.

- *Prediction:*

$$\begin{aligned} \mathbf{m}_k^-(\boldsymbol{\theta}) &= \int \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta})\ \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}(\boldsymbol{\theta}), \mathbf{P}_{k-1}(\boldsymbol{\theta}))\ \mathrm{d}\mathbf{x}_{k-1}, \\ \mathbf{P}_k^-(\boldsymbol{\theta}) &= \int (\mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}) - \mathbf{m}_k^-(\boldsymbol{\theta}))\,(\mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}) - \mathbf{m}_k^-(\boldsymbol{\theta}))^{\mathsf{T}} \\ &\qquad \times \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}(\boldsymbol{\theta}), \mathbf{P}_{k-1}(\boldsymbol{\theta}))\ \mathrm{d}\mathbf{x}_{k-1} + \mathbf{Q}_{k-1}(\boldsymbol{\theta}). \end{aligned} \tag{12.53}$$

- *Update:*

$$
\begin{aligned}
\boldsymbol{\mu}_k(\boldsymbol{\theta}) &= \int \mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta})\ \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-(\boldsymbol{\theta}), \mathbf{P}_k^-(\boldsymbol{\theta}))\ \mathrm{d}\mathbf{x}_k, \\
\mathbf{v}_k(\boldsymbol{\theta}) &= \mathbf{y}_k - \boldsymbol{\mu}_k(\boldsymbol{\theta}), \\
\mathbf{S}_k(\boldsymbol{\theta}) &= \int (\mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}) - \boldsymbol{\mu}_k(\boldsymbol{\theta}))\, (\mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}) - \boldsymbol{\mu}_k(\boldsymbol{\theta}))^\mathsf{T} \\
&\qquad \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-(\boldsymbol{\theta}), \mathbf{P}_k^-(\boldsymbol{\theta}))\ \mathrm{d}\mathbf{x}_k + \mathbf{R}_k(\boldsymbol{\theta}), \\
\mathbf{C}_k(\boldsymbol{\theta}) &= \int (\mathbf{x}_k - \mathbf{m}_k^-(\boldsymbol{\theta}))\, (\mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}) - \boldsymbol{\mu}_k(\boldsymbol{\theta}))^\mathsf{T} \\
&\qquad \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-(\boldsymbol{\theta}), \mathbf{P}_k^-(\boldsymbol{\theta}))\ \mathrm{d}\mathbf{x}_k, \\
\mathbf{K}_k(\boldsymbol{\theta}) &= \mathbf{C}_k(\boldsymbol{\theta})\, \mathbf{S}_k^{-1}(\boldsymbol{\theta}), \\
\mathbf{m}_k(\boldsymbol{\theta}) &= \mathbf{m}_k^-(\boldsymbol{\theta}) + \mathbf{K}_k(\boldsymbol{\theta})\, \mathbf{v}_k(\boldsymbol{\theta}), \\
\mathbf{P}_k(\boldsymbol{\theta}) &= \mathbf{P}_k^-(\boldsymbol{\theta}) - \mathbf{K}_k(\boldsymbol{\theta})\, \mathbf{S}_k(\boldsymbol{\theta})\, \mathbf{K}_k^\mathsf{T}(\boldsymbol{\theta}).
\end{aligned} \tag{12.54}
$$

Derivation This approximation can be derived in the same way as the linear case in Theorem 12.3 except that Gaussian moment matching based approximations are used instead of the true Gaussian distributions. □

The above energy function can now be directly used in MCMC sampling or in gradient-free optimization algorithms for computing ML or MAP estimates. However, because the energy function is based on a Gaussian approximation, the implied posterior distribution is an approximation as well and thus the parameter estimates will be biased. The posterior distribution approximation is also typically thinner than the true posterior distribution and thus the uncertainty in the parameter is underestimated. This issue is illustrated in Example 12.2.

It is also possible to compute the derivatives of the above energy function analogously to the linear case. In the case of the extended Kalman filter (EKF), the derivatives can be easily derived by formally differentiating the EKF equations (see Mbalawata et al., 2013). When sigma-point filters are used, a similar approach works, but additional care is needed for computation of the derivative of the square root matrix $\partial \sqrt{\mathbf{P}(\boldsymbol{\theta})}/\partial \theta_i$ arising in the equations. The equations for computing the derivatives of the energy function are given in Algorithm A.3.

To compute the expectations required for implementing the EM algorithm, we can approximate the integrals in Equations (12.30) using the Gaussian assumed density approximation (i.e., moment matching). The resulting expression for $\mathcal{Q}$ is the following.

Algorithm 12.7 (Evaluation of $\mathcal{Q}$ via Gaussian smoothing) *The expression for $\mathcal{Q}$ for the non-linear state space model in Equation* (12.51) *can be written as*

$$\begin{aligned}
&\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) \\
&\simeq -\frac{1}{2}\log|2\pi\,\mathbf{P}_0(\boldsymbol{\theta})| - \frac{T}{2}\log|2\pi\,\mathbf{Q}(\boldsymbol{\theta})| - \frac{T}{2}\log|2\pi\,\mathbf{R}(\boldsymbol{\theta})| \\
&- \frac{1}{2}\operatorname{tr}\left\{\mathbf{P}_0^{-1}(\boldsymbol{\theta})\left[\mathbf{P}_0^{\mathrm{s}} + (\mathbf{m}_0^{\mathrm{s}} - \mathbf{m}_0(\boldsymbol{\theta}))\,(\mathbf{m}_0^{\mathrm{s}} - \mathbf{m}_0(\boldsymbol{\theta}))^{\mathsf{T}}\right]\right\} \\
&- \frac{1}{2}\sum_{k=1}^{T}\operatorname{tr}\left\{\mathbf{Q}^{-1}(\boldsymbol{\theta})\,\mathrm{E}\left[(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}))\,(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}))^{\mathsf{T}} \mid \mathbf{y}_{1:T}\right]\right\} \\
&- \frac{1}{2}\sum_{k=1}^{T}\operatorname{tr}\left\{\mathbf{R}^{-1}(\boldsymbol{\theta})\,\mathrm{E}\left[(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}))\,(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}))^{\mathsf{T}} \mid \mathbf{y}_{1:T}\right]\right\},
\end{aligned} \tag{12.55}$$

where the expectations are over the counterparts of the distributions in Equations (12.42) *obtained from the Gaussian smoother.*

In practice, we can approximate the Gaussian smoother and Gaussian integrals above with Taylor series approximations (EKF/ERTSS) or by sigma-point methods such as Gauss–Hermite or spherical cubature integration or the unscented transform. The M-step for the noise parameters can indeed be implemented analogously to the linear case in Theorem 12.5, because the maxima of the above $\mathcal{Q}$ with respect to the noise covariance are simply

$$\begin{aligned}
\mathbf{Q}^* &= \frac{1}{T}\sum_{k=1}^{T}\mathrm{E}\left[(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}))\,(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}))^{\mathsf{T}} \mid \mathbf{y}_{1:T}\right], \\
\mathbf{R}^* &= \frac{1}{T}\sum_{k=1}^{T}\mathrm{E}\left[(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}))\,(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}))^{\mathsf{T}} \mid \mathbf{y}_{1:T}\right].
\end{aligned} \tag{12.56}$$

The details of implementation of the M-step for other kinds of parameter depends on the actual functional form of $\mathbf{f}$ and $\mathbf{h}$. If the parameters appear linearly in the functions, it is possible to find closed form solutions for the maxima analogously to the linear case (Theorem 12.5). Obviously, even when analytical solutions cannot be found, it would be possible to use iterative optimization methods inside EM. But if iterative methods need to be used anyway, then with the same effort we can try to find the minimum

of the energy function directly (recall that it is what EM tries to find as well).

Also in the non-linear case Fisher's identity (Equation (12.32)), in principle, gives an easy way to evaluate the gradients of the energy function. The problem is that both the energy function and the gradient given by Fisher's identity are approximations, and there is no guarantee that the approximations involved are the same. That is, the derivative given by Fisher's identity might not be exactly the derivative of the approximate energy function given by the Gaussian filter. The derivation of the Fisher's identity based derivative expression is left as an exercise to the reader.

12.3.4 Parameter estimation via particle filtering and smoothing

Particle filtering can also be used for approximate evaluation of the marginal likelihood and also the energy function needed in parameter estimation. In the particle filtering approach we can consider generic models of the form

$$\begin{aligned} \boldsymbol{\theta} &\sim p(\boldsymbol{\theta}), \\ \mathbf{x}_0 &\sim p(\mathbf{x}_0 \mid \boldsymbol{\theta}), \\ \mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}), \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}), \end{aligned} \tag{12.57}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the unknown parameter to be estimated. The approximate evaluation of the marginal likelihood can be done with the following modification of the SIR particle filter (see, e.g., Andrieu et al., 2004; Creal, 2012).

Algorithm 12.8 (SIR based energy function approximation) *An approximation to the marginal likelihood of the parameters can be evaluated during the sequential importance resampling (SIR) algorithm (particle filter), as follows.*

- *Draw N samples $\mathbf{x}_0^{(i)}$ from the prior:*

$$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0 \mid \boldsymbol{\theta}), \qquad i = 1, \ldots, N, \tag{12.58}$$

  and set $w_0^{(i)} = 1/N$, for all $i = 1, \ldots, N$.
- *For each $k = 1, \ldots, T$ do the following.*

  1 Draw samples $\mathbf{x}_k^{(i)}$ from the importance distributions:

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k}, \boldsymbol{\theta}), \qquad i = 1, \ldots, N. \tag{12.59}$$

2 *Compute the following weights:*

$$v_k^{(i)} = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}, \boldsymbol{\theta})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \boldsymbol{\theta})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k}, \boldsymbol{\theta})} \tag{12.60}$$

and compute the estimate of $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})$ *as*

$$\hat{p}(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) = \sum_i w_{k-1}^{(i)}\, v_k^{(i)}. \tag{12.61}$$

3 *Compute the normalized weights as*

$$w_k^{(i)} \propto w_{k-1}^{(i)}\, v_k^{(i)}. \tag{12.62}$$

4 *If the effective number of particles* (7.27) *is too low, perform resampling.*

The approximation of the marginal likelihood of the parameters is

$$p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}) \approx \prod_k \hat{p}(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}), \tag{12.63}$$

and the corresponding energy function approximation is

$$\varphi_T(\boldsymbol{\theta}) \approx -\log p(\boldsymbol{\theta}) - \sum_{k=1}^{T} \log \hat{p}(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}). \tag{12.64}$$

The energy function approximation could also be computed recursively during the SIR algorithm above.

The particle filter based energy function approximation can now be used, for example, in the Metropolis–Hastings based MCMC algorithm. The result is the particle Markov chain Monte Carlo (PMCMC) method (Andrieu et al., 2010) and, more specifically, the particle marginal Metropolis–Hastings (PMMH) algorithm variant of it. Although the idea of using a particle filter based likelihood approximations in MCMC is an old and obvious idea (see Andrieu et al., 2004), it was only proved recently by Andrieu et al. (2010) that the resulting algorithm is exact in the sense that it samples from the right distribution. Thus the result is (asymptotically) the same as if we had used the exact energy function instead of the particle filter based approximation in the MCMC method.

Example 12.2 (Estimation of noise variance in the pendulum model) *Figure 12.2 shows the posterior distribution approximation for the noise variance* R *computed with a Gaussian filter for the pendulum model in Example 3.7. The figure also shows the histogram of samples from the PMCMC*

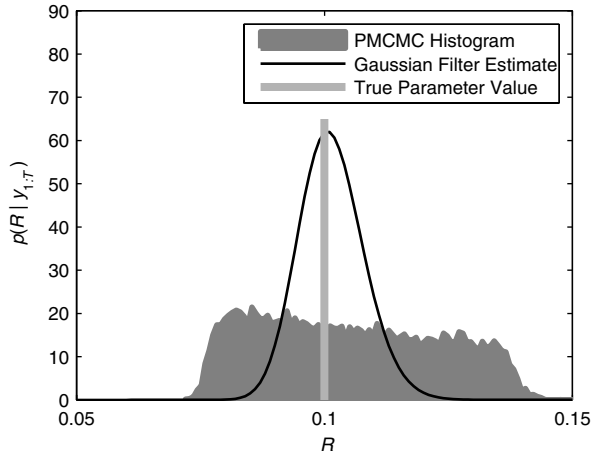


Figure 12.2 Posterior distribution of the noise variance R in the pendulum model (see Example 12.2). The approximation computed with the the Gaussian filter is thinner than the reference result computed with the PMCMC. Thus the uncertainty in the parameter estimate is not properly captured by the Gaussian filter based approximation.

method which thus should approach the true posterior of the parameter. As can be seen, the posterior distribution estimate of the Gaussian filter (fifth order Gauss–Hermite filter) is a bit thinner than the true posterior distribution. That is, the uncertainty in the parameter value is underestimated. In this case we are lucky, because the true parameter value still remains inside the high density area of the posterior distribution approximation, but this might not always be the case.

In principle, it would also be possible to use the likelihood or energy function approximation in gradient-free optimization methods for finding MAP or ML estimates. However, this might turn out to be hard, because even if we fixed the random number generator sequence in the particle filter, the likelihood function would not be continuous in $\boldsymbol{\theta}$ (see, e.g., Kantas et al., 2009). This also renders the use of gradient based optimization methods impossible.

By comparing to the Rao–Blackwellized particle filter in Algorithm 7.6, it is easy to see that the corresponding likelihood approximation can be

obtained by setting

$$v_k^{(i)} = \frac{p(\mathbf{y}_k \mid \mathbf{u}_{0:k}^{(i)}, \mathbf{y}_{1:k-1}, \boldsymbol{\theta})\; p(\mathbf{u}_k^{(i)} \mid \mathbf{u}_{k-1}^{(i)}, \boldsymbol{\theta})}{\pi(\mathbf{u}_k^{(i)} \mid \mathbf{u}_{0:k-1}^{(i)}, \mathbf{y}_{1:k}, \boldsymbol{\theta})}. \tag{12.65}$$

The likelihood approximation itself remains the same as in Equation (12.61):

$$\hat{p}(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}) = \sum_i w_{k-1}^{(i)}\, v_k^{(i)}.$$

We can also implement the EM algorithm using particle smoothing. Recall that to implement the EM algorithm we need to evaluate $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})$ via Equation (12.29) which in turn requires us to evaluate the expectations appearing in Equation (12.30). The actual form of the approximation depends on the particle smoother that we are using. In the case of the backward-simulation smoother we have the following simple algorithm (Wills et al., 2013).

Algorithm 12.9 (Evaluation of $\mathcal{Q}$ via the backward-simulation smoother) *Assume that we have simulated S trajectories $\{\tilde{\mathbf{x}}_{0:T}^{(i)} : i = 1, \ldots, S\}$ using the backward-simulation smoother in Algorithm 11.2 with parameter values fixed to $\boldsymbol{\theta}^{(n)}$. Then the integrals in Equation* (12.30) *can be approximated as*

$$\begin{aligned}
I_1(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) &\approx \frac{1}{S} \sum_{i=1}^{S} \log p(\tilde{\mathbf{x}}_0^{(i)} \mid \boldsymbol{\theta}), \\
I_2(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) &\approx \sum_{k=0}^{T-1} \frac{1}{S} \sum_{i=1}^{S} \log p(\tilde{\mathbf{x}}_{k+1}^{(i)} \mid \tilde{\mathbf{x}}_k^{(i)}, \boldsymbol{\theta}), \\
I_3(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) &\approx \sum_{k=1}^{T} \frac{1}{S} \sum_{i=1}^{S} \log p(\mathbf{y}_k \mid \tilde{\mathbf{x}}_k^{(i)}, \boldsymbol{\theta}).
\end{aligned} \tag{12.66}$$

If we are using the reweighting (or marginal) particle smoother in Algorithm 11.3, the corresponding expectations can be approximated as follows (Schön et al., 2011).

Algorithm 12.10 (Evaluation of $\mathcal{Q}$ via the reweighting smoother) *Assume that we have the set of particles $\{\mathbf{x}_k^{(i)} : k = 0, \ldots, T; i = 1, \ldots, N\}$ representing the filtering distribution and we have calculated the weights $\{w_{k|T}^{(i)} : k = 0, \ldots, T; i = 1, \ldots, N\}$ using Algorithm 11.3. Then we can*

approximate the integrals in Equation (12.30) *as follows:*

$$
\begin{aligned}
I_1(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) &\approx \sum_i w_{0|T}^{(i)} \log p(\mathbf{x}_0^{(i)} \mid \boldsymbol{\theta}), \\
I_2(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) &\approx \sum_{k=0}^{T-1} \sum_i \sum_j \frac{w_{k+1|T}^{(j)} \, w_k^{(i)} \, p(\mathbf{x}_{k+1}^{(j)} \mid \mathbf{x}_k^{(i)}, \boldsymbol{\theta}^{(n)})}{\left[\sum_l w_k^{(l)} \, p(\mathbf{x}_{k+1}^{(j)} \mid \mathbf{x}_k^{(l)}, \boldsymbol{\theta}^{(n)})\right]} \\
&\qquad \times \log p(\mathbf{x}_{k+1}^{(j)} \mid \mathbf{x}_k^{(i)}, \boldsymbol{\theta}), \\
I_3(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)}) &\approx \sum_{k=1}^{T} \sum_i w_{k|T}^{(i)} \log p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}, \boldsymbol{\theta}).
\end{aligned} \tag{12.67}
$$

By differentiating the expressions in the above algorithms with respect to $\boldsymbol{\theta}$, we can also get an approximation for $\partial \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})/\partial \boldsymbol{\theta}$. This approximation can be further used in Fisher's identity in Equation (12.32) to give approximations for the gradients of the energy function (Ninness et al., 2010). For more information on this kind of approach as well as other methods for particle filtering based parameter estimation, the reader is referred to Andrieu et al. (2004), Kantas et al. (2009), and Poyiadjis et al. (2011).

12.3.5 Rao–Blackwellization of parameters

In this section we show how Rao–Blackwellization can sometimes be used for marginalizing out the static parameters in state space models. Let's start by considering the following generalized version of the pendulum example used in Särkkä (2006) and Särkkä and Sottinen (2008):

$$
\begin{aligned}
\mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, \\
y_k &= h(\mathbf{x}_k) + r_k, \\
r_k &\sim \mathrm{N}(0, R), \\
R &\sim \text{Inv-}\chi^2(\nu_0, R_0),
\end{aligned} \tag{12.68}
$$

where $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$. This is thus the same kind of non-linear state space model that we have already considered in this book, except that here the measurement noise variance R is considered as unknown and given an inverse-chi-squared prior distribution Inv-$\chi^2(\nu_0, R_0)$.

It turns out that we can do sequential Monte Carlo sampling in this model such that we do not need to sample the values of R. Instead, it is enough to sample the state values and then carry the parameters of the distribution of R, conditional on the previous measurements and the histories of samples. The idea is the following.

1 Assume that we have generated a set of particle histories $\{w_k^{(i)}, \mathbf{x}_{0:k}^{(i)} : i = 1, \ldots, N\}$ which approximate the full distribution of the states as follows:

$$p(\mathbf{x}_{0:k-1} \mid y_{1:k-1}) \approx \sum_i w_{k-1}^{(i)} \, \delta(\mathbf{x}_{0:k-1} - \mathbf{x}_{0:k-1}^{(i)}), \tag{12.69}$$

which is thus the conventional SIR filter approximation when we store the full sample histories (see Section 11.1). Further assume that the conditional distribution of R given measurements $y_{1:k-1}$ and the sampled state history $\mathbf{x}_{0:k-1}^{(i)}$ is

$$p(R \mid \mathbf{x}_{0:k-1}^{(i)}, y_{1:k-1}) = \text{Inv-}\chi^2(R \mid \nu_{k-1}^{(i)}, R_{k-1}^{(i)}), \tag{12.70}$$

where $\nu_{k-1}^{(i)}, R_{k-1}^{(i)}$ have already been computed for each i. Then we have the following approximation for the full distribution of states and parameters:

$$\begin{aligned} &p(\mathbf{x}_{0:k-1}, R \mid y_{1:k-1}) \\ &= p(R \mid \mathbf{x}_{0:k-1}, y_{1:k-1}) \, p(\mathbf{x}_{0:k-1} \mid y_{1:k-1}) \\ &\approx \sum_i w_{k-1}^{(i)} \, \text{Inv-}\chi^2(R \mid \nu_{k-1}^{(i)}, R_{k-1}^{(i)}) \, \delta(\mathbf{x}_{0:k-1} - \mathbf{x}_{0:k-1}^{(i)}). \end{aligned} \tag{12.71}$$

2 Let's now draw samples from an importance distribution:

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, y_k). \tag{12.72}$$

3 We can now evaluate the likelihood of the measurement given $\mathbf{x}_k^{(i)}$ and the previous measurements as follows:

$$\begin{aligned} &p(y_k \mid \mathbf{x}_k^{(i)}, y_{1:k-1}) \\ &= \int \text{N}(y_k \mid h(\mathbf{x}_k^{(i)}), R) \, \text{Inv-}\chi^2(R \mid \nu_{k-1}^{(i)}, R_{k-1}^{(i)}) \, \mathrm{d}R \\ &= t_{\nu_k^{(i)}}(y_k \mid h(\mathbf{x}_k^{(i)}), R_k^{(i)}), \end{aligned} \tag{12.73}$$

where the parameters of the Student's t-distribution above are

$$\begin{aligned} \nu_k^{(i)} &= \nu_{k-1}^{(i)} + 1, \\ R_k^{(i)} &= \frac{\nu_{k-1}^{(i)} R_{k-1}^{(i)} + (y_k - h(\mathbf{x}_k^{(i)}))^2}{\nu_{k-1}^{(i)} + 1}. \end{aligned} \tag{12.74}$$

This allows us to compute the next step importance weights for the SIR algorithm as follows:

$$w_k^{(i)} \propto \frac{p(y_k \mid \mathbf{x}_k^{(i)}, y_{1:k-1})\ \mathrm{N}(\mathbf{x}_k^{(i)} \mid \mathbf{f}(\mathbf{x}_{k-1}^{(i)}), \mathbf{Q})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, y_k)}. \tag{12.75}$$

4 Given the measurement and the state we can further compute the conditional distribution of R given $y_{1:k}$ and $\mathbf{x}_{0:k}^{(i)}$:

$$p(R \mid \mathbf{x}_{0:k}^{(i)}, y_{1:k}) = \text{Inv-}\chi^2(R \mid \nu_k^{(i)}, R_k^{(i)}). \tag{12.76}$$

5 Now we again have a similar representation for the filtering distribution as in step 1 and we can set $k \leftarrow k+1$ and go back to step 1. But before that we can also do resampling jointly for the state histories $\mathbf{x}_{0:k}^{(i)}$ and the parameters $\nu_k^{(i)}$, $R_k^{(i)}$ as in the conventional SIR algorithm.

In the above algorithm, we would not actually need to carry the whole state histories in the filter, but theoretically the parameters of the inverse chi squared distributions are indeed conditioned on the full state histories.

The procedure above is a Rao–Blackwellized particle filter where the static parameter R has been marginalized out and is carried via its sufficient statistics $\nu_k^{(i)}$, $R_k^{(i)}$ conditioned on the particle histories $\mathbf{x}_{0:k}^{(i)}$ and measurements. In another example given in Särkkä (2006) and Särkkä and Sottinen (2008) this procedure is used for marginalizing out the unknown population size in a Poisson measurement model. The same idea can be used in various other types of model.

In an abstract sense the method can be applied to a class of models of the form

$$\begin{aligned} \mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}), \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}), \\ \boldsymbol{\theta} &\sim p(\boldsymbol{\theta}), \end{aligned} \tag{12.77}$$

where the vector $\boldsymbol{\theta}$ contains the unknown static parameters. Now if the posterior distribution of the parameters $\boldsymbol{\theta}$ depends only on some sufficient statistics

$$\mathcal{T}_k = \mathcal{T}_k(\mathbf{x}_{1:k}, \mathbf{y}_{1:k}), \tag{12.78}$$

and if the sufficient statistics are easy to update recursively, $\mathcal{T}_k \leftarrow \mathcal{T}_{k-1}$, then sampling of the state and parameters can be performed by recursively computing the sufficient statistics conditionally on the sampled states and the measurements analogously to the example above. The original idea of the method seems to have appeared quite independently in Storvik (2002),

Fearnhead (2002), Djuric and Miguez (2002), and more recently it has been applied to estimation of full noise covariances in state space models by Saha et al. (2010).

A particularly useful special case, which includes the example above, is obtained when the dynamic model is independent of the parameters $\boldsymbol{\theta}$. In this case, if conditionally to the state $\mathbf{x}_k$ the prior $p(\boldsymbol{\theta})$ belongs to the conjugate family of the likelihood $p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta})$, the static parameters $\boldsymbol{\theta}$ can be marginalized out and only the states need to be sampled. This idea can be extended to the time-varying case if the dynamic model has such a form which keeps the predicted distribution of the parameter within the conjugate family (see Särkkä and Nummenmaa, 2009).

When the static parameter appears linearly in the model we recover a noise free version of the conditionally Gaussian Rao–Blackwellization considered in Section 7.5 (see Schön and Gustafsson, 2003). The Rao–Blackwellized particle filter can then be seen as a time-varying extension of this method in the conditionally linear Gaussian case.

12.4 Exercises

12.1 Implement the EM algorithm for ML estimation of the measurement noise variance in the Gaussian random walk model considered in Examples 4.1, 4.2, and 8.1. Test the algorithm with simulated data.

12.2 Implement the algorithm for computing the energy function for the Gaussian random walk model as well as its derivative with respect to the noise variance (via the sensitivity equations given in Section A.3). Generate some simulated data and use a gradient based optimization method to find the ML estimate of the parameter.

12.3 With the Gaussian random walk model, find the expression for the Fisher's identity based derivative with respect to the noise parameter. Check numerically that it matches the expression obtained with the sensitivity equations.

12.4 Implement a random walk Metropolis based MCMC method for estimating the noise variance in the Gaussian random walk model. Use the Kalman filter for evaluating the energy function. For simplicity, you can assume that the prior for the parameter has the form $p(R) \propto 1$.

12.5 Derive the sensitivity equations for the first order extended Kalman filter.

12.6 Derive the equation for the derivative of the energy function resulting from differentiating the Gaussian smoothing based approximation in Algorithm 12.7 and using Fisher's identity (Equation (12.32)).

12.7 Compute and plot the approximate energy function obtained for the noise variance in the model given in Exercise 5.1 by using a non-linear Kalman filter based estimate of the energy function. You can select the filter freely.

12.8 Implement a random walk Metropolis based MCMC method for estimating the noise variance in the model given in Exercise 5.1. Use one of the non-linear Kalman filters to approximate the energy function.

12.9 Implement a random walk Metropolis based particle MCMC method for estimating the noise variance in the Gaussian random walk model. Use a simple bootstrap filter as the particle filter.

12.10 Implement a random walk Metropolis based particle MCMC method for estimating the noise variance in the model given in Exercise 5.1.

13

Epilogue

13.1 Which method should I choose?

An important question when preparing to solve a specific filtering, smoothing, or parameter estimation problem for state space models is: *which of the numerous methods should I choose for a particular application*? Obviously if the problem is linear, then the Kalman filter and RTS smoother are natural choices – also for evaluating the quantities needed for parameter estimation. But if the system is non-linear/non-Gaussian the question is harder.

When the noises in the system can be modeled as Gaussian and the model is of the form

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k, \end{aligned} \tag{13.1}$$

where $\mathbf{f}$ and $\mathbf{h}$ are somewhat well-behaved functions, then the first choice would be one of the Gaussian approximation based filters and smoothers – provided that we are working on an application and the theoretical exactness of the solution is not important per se, but we are interested in getting good estimates of the state and parameters. If theoretical exactness is needed, then the only option is to use particle filters and smoothers (or grid based solutions).

Among the Gaussian approximation based filters and smoothers it is always a good idea to start with an EKF and an ERTSS. These are the only algorithms that have been used for over half a century in practical applications and there are good reasons for that – they simply work. Statistical linearization can sometimes be used to enhance implementations of the EKF afterwards by replacing some of the function evaluations by their expectations. Otherwise the SLF is a theoretical tool rather than a practical filtering algorithm.

With some models the EKF and ERTSS do not work well or at all, and in that case we can move to the sigma-point methods. The spherical cubature and unscented methods have the advantage of being computationally quite light, but still they tend to produce very good results. However, these methods have the problem that their error estimates might not always be consistent with the actual errors, a problem which the EKF/ERTSS methods also tend to have. The unscented transform has more parameters to tune for a particular problem than the spherical cubature method, which can be an advantage or a disadvantage (recall that the cubature spherical method is an unscented transform with a certain selection of parameters). The Gauss–Hermite based methods tend to be more consistent in errors and are thus more robust approximations, but have the disadvantage of having high computational complexity. One should always remember that there is no guarantee that using more complicated filtering and smoothing algorithms would actually improve the results, therefore it is a good idea to always test the EKF and ERTSS first. The bootstrap filter has the advantage that it is very easy to implement and thus it can sometimes be used as a reference solution when testing the performance and debugging Gaussian approximation based filters and smoothers.

If the problem has a more complicated form which cannot be fitted into the non-linear Gaussian framework or when the Gaussian approximations do not work for other reasons, we need to go to particle based solutions. Because the bootstrap filter is very easy to implement, it (and probably one of the particle smoothers) should be the first option to test with a sufficiently large number of particles. However, the clear disadvantage of particle methods is the high computational load and thus it is a good idea to check at quite an early stage if any of the states or parameters can be marginalized out ("Rao–Blackwellized") exactly or approximately. If this is the case, then one should always prefer marginalization to sampling.[1] The other thing to check is if it is possible to use the optimal or almost optimal importance distribution in the particle filter. In principle, non-linear Gaussian approximation based filters can be used to form such importance distributions, but this may also lead to overly heavy computational methods as well as to convergence problems. If they are used, then it might be advisable to artificially increase the filter covariances a bit or to use Student's t distributions instead of using the Gaussian approximations as such.

[1] The rule of thumb is: use Monte Carlo sampling only as a last resort when all the other options have failed.

When it comes to parameter estimation, it is generally a good idea to use the same approximations in the parameter estimation method as will be used in the final application, assuming that the parameter estimation results will later be used in filters and smoothers to solve a state estimation problem. Furthermore, if a single parameter estimation result (point estimate) is needed anyway, ML and MAP estimates are not bad choices, but it might be useful to check the spread of the posterior distribution of parameters using an MCMC method. But if the true values of the parameters are of interest, then the combination of particle filtering and MCMC is probably the safest bet. However, one should remember that estimating the true parameters of the system is possible only in simulated scenarios and in real applications the models used will be more or less wrong anyway. On the other hand, we should be careful not to ruin already probably inaccurate models with bad approximations of filters and smoothers.

13.2 Further topics

This book is mainly concerned with non-linear Kalman filtering and smoothing as well as particle filtering and smoothing approaches to Bayesian filtering and smoothing, but numerous other methods exist as well. It is impossible to list all of them, but below we try to give pointers to some of the methods. Regarding filters and smoothers themselves, there are also various subareas that we did not discuss and we try to give some pointers to them as well.

First of all, one huge area that we have not mentioned at all is continuous time filters and smoothers. In these methods the dynamics of the state and sometimes the measurements as well are modeled using stochastic differential equations (SDEs) (Øksendal, 2003). The full theory of Bayesian filtering in such models can be found in the classic book of Jazwinski (1970) and the smoothing theory is due to Striebel (1965) and Leondes et al. (1970). The extended Kalman type of filter approximation can be found in the above-mentioned references as well. Extensions of unscented Kalman filters and smoothers to the continuous-time setting can be found in Särkkä (2006, 2007, 2010). Extensions of Gaussian filters and smoothers to continuous-time setting have been discussed in Singer (2008); Arasaratnam et al. (2010); Singer (2011); Särkkä and Solin (2012); Särkkä and Sarmavuori (2013). Extensions of particle filters and smoothers to continuous-time setting can be found, for example, in Crisan and Rozovskii (2011); Särkkä and Sottinen (2008); Murray and Storkey (2011), and references therein.

There also exist various other kinds of Gaussian integration methods that we have not presented here that could be used for constructing new kinds of Gaussian filters and smoothers (see, e.g., O'Hagan, 1991; Nørgaard et al., 2000; Lefebvre et al., 2002; Wu et al., 2006; Särkkä and Hartikainen, 2010b; Sandblom and Svensson, 2012). One particularly interesting approach is to approximate the non-linear functions with a Gaussian process based non-parametric model which is fitted using a finite number of sample points (Deisenroth et al., 2009, 2012).

One useful class of discrete-time methods is the multiple model approaches such as the generalized pseudo-Bayesian methods (GPB1 and GPB2) as well as the interacting multiple model (IMM) algorithm (Bar-Shalom et al., 2001). These methods can be used for approximating the Bayesian solutions to problems with a fixed number of models or modes of operation. The active mode of the system is described by a discrete latent variable which is modeled as a discrete-state Markov chain. Given the value of the latent variable, the system is (approximately) Gaussian. The GPB1, GPB2, and IMM algorithms are based on forming a mixture of Gaussians approximation (a bank of Kalman or extended Kalman filters) to the Bayesian filtering solutions by using moment matching.

The above-mentioned multiple model methods are also closely related to so-called expectation correction (EC, Barber, 2006) and expectation propagation (EP, Zoeter and Heskes, 2011) methods, which can also be used for Bayesian filtering and smoothing in switching linear dynamic systems (SLDS), which is another term used for multiple mode/model problems. These models can also be considered as special cases of the conditionally Gaussian models considered in the previous section and the history of similar approximations dates back to the works of Alspach and Sorenson (1972) and Akashi and Kumamoto (1977). The relationship between various methods for this type of model has recently been analyzed by Barber (2011).

When the measurement model is non-Gaussian (e.g., Student's t), it is sometimes possible to use variational Bayes approximations (Agamennoni et al., 2011; Piché et al., 2012) to yield to tractable inference. The expectation propagation (EP) algorithm (Ypma and Heskes, 2005) can also be used for approximate inference in non-linear and non-Gaussian dynamic systems. Both of these approaches are also closely related to the Gaussian filters and smoothers considered in this book. Variational Bayesian approximations can also be used for estimation of unknown time-varying parameters in state space models (Särkkä and Nummenmaa, 2009).

In the multiple target tracking context there exist a number of specific methods to cope with the problems arising there, namely, the data association problem and an unknown numbers of targets. The classical approaches to the multiple target tracking problem can be found in the books of Bar-Shalom and Li (1995) and Blackman and Popoli (1999). For more recent approaches the reader is referred to Ristic et al. (2004) and Challa et al. (2011) (see also Särkkä et al., 2007b).

There are a number of other important topics that have had to be omitted here. For example, the Cramér–Rao lower bounds (CRLB, see, e.g., Ristic et al., 2004) are theoretical lower bounds for the mean-squared error that can be achieved with any non-linear filter in a given filtering problem. We have also omitted the observability and controllability questions of linear and non-linear systems (see, e.g., Kailath et al., 2000; Bar-Shalom et al., 2001), which are related to the question of whether it is possible to determine the state of a given system from the available measurements at all. A somewhat related issue is the question that under what conditions does a particle filter converge to the true distribution. For more details on this topic the reader is referred to the works of Crisan and Doucet (2002) and Hu et al. (2008, 2011). The numerical stability of linear and non-linear Kalman filters and RTS smoothers can sometimes also be enhanced by using square root versions of them. This kind of method can be found, for example, in the works of Bierman (1977), Grewal and Andrews (2001), Van der Merwe and Wan (2001), and Arasaratnam and Haykin (2009, 2011).

Appendix

Additional material

A.1 Properties of Gaussian distribution

Definition A.1 (Gaussian distribution) *A random variable* $\mathbf{x} \in \mathbb{R}^n$ *has a Gaussian distribution with mean* $\mathbf{m} \in \mathbb{R}^n$ *and covariance* $\mathbf{P} \in \mathbb{R}^{n \times n}$ *if its probability density has the form*

$$\mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P}) = \frac{1}{(2\,\pi)^{n/2}\,|\mathbf{P}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathsf{T}}\,\mathbf{P}^{-1}\,(\mathbf{x}-\mathbf{m})\right), \tag{A.1}$$

where $|\mathbf{P}|$ *is the determinant of the matrix* $\mathbf{P}$.

Lemma A.1 (Joint distribution of Gaussian variables) *If random variables* $\mathbf{x} \in \mathbb{R}^n$ *and* $\mathbf{y} \in \mathbb{R}^m$ *have the Gaussian probability distributions*

$$\begin{aligned} \mathbf{x} &\sim \mathrm{N}(\mathbf{m}, \mathbf{P}), \\ \mathbf{y} \mid \mathbf{x} &\sim \mathrm{N}(\mathbf{H}\,\mathbf{x} + \mathbf{u}, \mathbf{R}), \end{aligned} \tag{A.2}$$

then the joint distribution of $\mathbf{x}, \mathbf{y}$ *and the marginal distribution of* $\mathbf{y}$ *are given as*

$$\begin{aligned} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} &\sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \mathbf{H}\,\mathbf{m} + \mathbf{u} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{P}\,\mathbf{H}^{\mathsf{T}} \\ \mathbf{H}\,\mathbf{P} & \mathbf{H}\,\mathbf{P}\,\mathbf{H}^{\mathsf{T}} + \mathbf{R} \end{pmatrix}\right), \\ \mathbf{y} &\sim \mathrm{N}(\mathbf{H}\,\mathbf{m} + \mathbf{u}, \mathbf{H}\,\mathbf{P}\,\mathbf{H}^{\mathsf{T}} + \mathbf{R}). \end{aligned} \tag{A.3}$$

Lemma A.2 (Conditional distribution of Gaussian variables) *If the random variables* $\mathbf{x}$ *and* $\mathbf{y}$ *have the joint Gaussian probability distribution*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix}, \begin{pmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^{\mathsf{T}} & \mathbf{B} \end{pmatrix}\right), \tag{A.4}$$

then the marginal and conditional distributions of $\mathbf{x}$ *and* $\mathbf{y}$ *are given as follows:*

$$\begin{aligned}
\mathbf{x} &\sim \mathrm{N}(\mathbf{a}, \mathbf{A}),\\
\mathbf{y} &\sim \mathrm{N}(\mathbf{b}, \mathbf{B}),\\
\mathbf{x} \mid \mathbf{y} &\sim \mathrm{N}(\mathbf{a} + \mathbf{C}\,\mathbf{B}^{-1}\,(\mathbf{y} - \mathbf{b}), \mathbf{A} - \mathbf{C}\,\mathbf{B}^{-1}\mathbf{C}^{\mathsf{T}}),\\
\mathbf{y} \mid \mathbf{x} &\sim \mathrm{N}(\mathbf{b} + \mathbf{C}^{\mathsf{T}}\,\mathbf{A}^{-1}\,(\mathbf{x} - \mathbf{a}), \mathbf{B} - \mathbf{C}^{\mathsf{T}}\,\mathbf{A}^{-1}\,\mathbf{C}).
\end{aligned} \tag{A.5}$$

A.2 Cholesky factorization and its derivative

The Cholesky factor of the symmetric positive definite matrix $\mathbf{P}$ is a lower triangular matrix $\mathbf{A}$ such that

$$\mathbf{P} = \mathbf{A}\,\mathbf{A}^{\mathsf{T}}. \tag{A.6}$$

The matrix $\mathbf{A}$ can be computed by the Cholesky factorization algorithm (see, e.g., Golub and van Loan, 1996) presented below.

Algorithm A.1 (Cholesky factorization) *The Cholesky factor* $\mathbf{A}$ *of the matrix* $\mathbf{P} \in \mathbb{R}^{n \times n}$ *can be computed as follows:*

procedure CHOL($\mathbf{P}$)
  for $i \leftarrow 1 \ldots n$ ***do***
    $A_{ii} = \sqrt{P_{ii} - \sum_{k<i} A_{ik}^2}$
    for $j \leftarrow i + 1 \ldots n$ ***do***
      $A_{ji} = \left(P_{ji} - \sum_{k<i} A_{jk}\,A_{ik}\right) / A_{ii}$
    end for
  end for
  return $\mathbf{A}$
end procedure

The partial derivative of the Cholesky factor $\partial\mathbf{A}/\partial\theta$ with respect to a scalar parameters θ can be computed using the following algorithm once $\mathbf{P}$ and $\partial\mathbf{P}/\partial\theta$ are known. The algorithm can be derived by formally differentiating the Cholesky factorization algorithm equations.

Algorithm A.2 (Partial derivative of Cholesky factorization) *The partial derivative* $\mathbf{D} = \partial\mathbf{A}/\partial\theta$ *of the Cholesky factor of* $\mathbf{P} \in \mathbb{R}^{n \times n}$ *with respect to a scalar parameter* θ *can be computed as follows:*

procedure DCHOL($\mathbf{P}, \partial\mathbf{P}/\partial\theta$)
  for $i \leftarrow 1 \ldots n$ ***do***
    $A_{ii} \leftarrow \sqrt{P_{ii} - \sum_{k<i} A_{ik}^2}$

```
        D_ii ← (∂P_ii/∂θ − Σ_{k<i} 2 D_ik A_ik)/A_ii/2
        for j ← i + 1 ... n do
            A_ji ← (P_ji − Σ_{k<i} A_jk A_ik)/A_ii
            temp ← ∂P_ji/∂θ − Σ_{k<i} (D_jk A_ik + A_jk D_ik)
            D_ji ← temp/A_ii − (D_ii/A_ii) A_ji
        end for
    end for
    return D
end procedure
```

Another way to compute the same derivative is via the following theorem.

Theorem A.1 (Partial derivative of Cholesky factorization) *The partial derivative* $\partial \mathbf{A}/\partial\theta$ *of the lower triangular Cholesky factor* $\mathbf{A}$ *such that* $\mathbf{P} = \mathbf{A}\,\mathbf{A}^\mathsf{T}$ *with respect to a scalar parameter* θ *can be computed as*

$$\frac{\partial \mathbf{A}}{\partial \theta} = \mathbf{A}\,\Phi\left(\mathbf{A}^{-1}\,\frac{\partial \mathbf{P}}{\partial \theta}\,\mathbf{A}^{-\mathsf{T}}\right), \tag{A.7}$$

where $\Phi(\cdot)$ *is a function returning the lower triangular part and half the diagonal of the argument as follows:*

$$\Phi_{ij}(\mathbf{M}) = \begin{cases} M_{ij}, & \text{if } i > j, \\ \frac{1}{2} M_{ij}, & \text{if } i = j, \\ 0, & \text{if } i < j. \end{cases} \tag{A.8}$$

Proof We use a similar trick that was used in the derivation of the time derivative of the Cholesky factor in Morf et al. (1978). We have

$$\mathbf{P} = \mathbf{A}\,\mathbf{A}^\mathsf{T}. \tag{A.9}$$

By taking the derivative with respect to θ we get

$$\frac{\partial \mathbf{P}}{\partial \theta} = \frac{\partial \mathbf{A}}{\partial \theta}\,\mathbf{A}^\mathsf{T} + \mathbf{A}\,\frac{\partial \mathbf{A}^\mathsf{T}}{\partial \theta}. \tag{A.10}$$

Multiplying from the left with $\mathbf{A}^{-1}$ and from the right with $\mathbf{A}^{-\mathsf{T}}$ gives

$$\mathbf{A}^{-1}\,\frac{\partial \mathbf{P}}{\partial \theta}\,\mathbf{A}^{-\mathsf{T}} = \mathbf{A}^{-1}\,\frac{\partial \mathbf{A}}{\partial \theta} + \frac{\partial \mathbf{A}^\mathsf{T}}{\partial \theta}\,\mathbf{A}^{-\mathsf{T}}. \tag{A.11}$$

Now the right-hand side is the sum of a lower triangular matrix and an upper triangular matrix with identical diagonals. Thus we can recover

$\mathbf{A}^{-1}\ \partial\mathbf{A}/\partial\theta$ via

$$\mathbf{A}^{-1}\,\frac{\partial \mathbf{A}}{\partial \theta} = \Phi\left(\mathbf{A}^{-1}\,\frac{\partial \mathbf{P}}{\partial \theta}\,\mathbf{A}^{-\mathsf{T}}\right), \tag{A.12}$$

where the function $\Phi(\cdot)$ returns the (strictly) lower triangular part of the argument and half of the diagonal. Multiplying from the left with $\mathbf{A}$ gives the result. □

A.3 Parameter derivatives for the Kalman filter

Theorem 12.3 gives the energy function (i.e., the negative logarithm of the unnormalized posterior distribution) of the parameters for the following linear Gaussian model:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}(\boldsymbol{\theta})\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{H}(\boldsymbol{\theta})\,\mathbf{x}_k + \mathbf{r}_k, \end{aligned} \tag{A.13}$$

where $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}(\boldsymbol{\theta}))$, $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}(\boldsymbol{\theta}))$, and $\mathbf{x}_0 \sim \mathrm{N}(\mathbf{m}_0(\boldsymbol{\theta}), \mathbf{P}_0(\boldsymbol{\theta}))$. The parameters derivatives which are needed, for example, for implementing a gradient based optimization method for finding ML or MAP estimates can be evaluated via the following *sensitivity equations* (Gupta and Mehra, 1974) which can be derived by termwise differentiation of the energy function and Kalman filter equations in Theorem 12.3.

Theorem A.2 (Energy function derivative for linear Gaussian model I) *The derivative of the energy function given in Theorem 12.3 can be computed via the following recursion along with Kalman filtering:*

$$\begin{aligned} \frac{\partial \varphi_k(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\partial \varphi_{k-1}(\boldsymbol{\theta})}{\partial \theta_i} + \frac{1}{2}\,\mathrm{tr}\left(\mathbf{S}_k^{-1}(\boldsymbol{\theta})\,\frac{\partial \mathbf{S}_k(\boldsymbol{\theta})}{\partial \theta_i}\right) + \mathbf{v}_k^{\mathsf{T}}(\boldsymbol{\theta})\,\mathbf{S}_k^{-1}(\boldsymbol{\theta})\,\frac{\partial \mathbf{v}_k(\boldsymbol{\theta})}{\partial \theta_i} \\ &\quad - \frac{1}{2}\mathbf{v}_k^{\mathsf{T}}(\boldsymbol{\theta})\,\mathbf{S}_k^{-1}(\boldsymbol{\theta})\,\frac{\partial \mathbf{S}_k(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{S}_k^{-1}(\boldsymbol{\theta})\,\mathbf{v}_k(\boldsymbol{\theta}), \end{aligned} \tag{A.14}$$

where on the Kalman filter prediction step we compute

$$\begin{aligned} \frac{\partial \mathbf{m}_k^-(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\partial \mathbf{A}(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{m}_{k-1}(\boldsymbol{\theta}) + \mathbf{A}(\boldsymbol{\theta})\,\frac{\partial \mathbf{m}_{k-1}(\boldsymbol{\theta})}{\partial \theta_i}, \\ \frac{\partial \mathbf{P}_k^-(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\partial \mathbf{A}(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{P}_{k-1}(\boldsymbol{\theta})\,\mathbf{A}^{\mathsf{T}}(\boldsymbol{\theta}) + \mathbf{A}(\boldsymbol{\theta})\,\frac{\partial \mathbf{P}_{k-1}(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{A}^{\mathsf{T}}(\boldsymbol{\theta}) \\ &\quad + \mathbf{A}(\boldsymbol{\theta})\,\mathbf{P}_{k-1}(\boldsymbol{\theta})\,\frac{\partial \mathbf{A}^{\mathsf{T}}(\boldsymbol{\theta})}{\partial \theta_i} + \frac{\partial \mathbf{Q}(\boldsymbol{\theta})}{\partial \theta_i}, \end{aligned} \tag{A.15}$$

and on the Kalman filter update step we compute

$$
\begin{aligned}
\frac{\partial \mathbf{v}_k(\boldsymbol{\theta})}{\partial \theta_i} &= -\frac{\partial \mathbf{H}(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{m}_k^-(\boldsymbol{\theta}) - \mathbf{H}(\boldsymbol{\theta})\,\frac{\partial \mathbf{m}_k^-(\boldsymbol{\theta})}{\partial \theta_i},\\
\frac{\partial \mathbf{S}_k(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\partial \mathbf{H}(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{P}_k^-(\boldsymbol{\theta})\,\mathbf{H}^\mathsf{T}(\boldsymbol{\theta}) + \mathbf{H}(\boldsymbol{\theta})\,\frac{\partial \mathbf{P}_k^-(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{H}^\mathsf{T}(\boldsymbol{\theta})\\
&\quad + \mathbf{H}(\boldsymbol{\theta})\,\mathbf{P}_k^-(\boldsymbol{\theta})\,\frac{\partial \mathbf{H}^\mathsf{T}(\boldsymbol{\theta})}{\partial \theta_i} + \frac{\partial \mathbf{R}(\boldsymbol{\theta})}{\partial \theta_i},\\
\frac{\partial \mathbf{K}_k(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\partial \mathbf{P}_k^-(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{H}^\mathsf{T}(\boldsymbol{\theta})\,\mathbf{S}_k^{-1}(\boldsymbol{\theta}) + \mathbf{P}_k^-(\boldsymbol{\theta})\,\frac{\partial \mathbf{H}^\mathsf{T}(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{S}_k^{-1}(\boldsymbol{\theta})\\
&\quad - \mathbf{P}_k^-(\boldsymbol{\theta})\,\mathbf{H}^\mathsf{T}(\boldsymbol{\theta})\,\mathbf{S}_k^{-1}(\boldsymbol{\theta})\,\frac{\partial \mathbf{S}_k(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{S}_k^{-1}(\boldsymbol{\theta}),\\
\frac{\partial \mathbf{m}_k(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\partial \mathbf{m}_k^-(\boldsymbol{\theta})}{\partial \theta_i} + \frac{\partial \mathbf{K}_k(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{v}_k(\boldsymbol{\theta}) + \mathbf{K}_k(\boldsymbol{\theta})\,\frac{\partial \mathbf{v}_k(\boldsymbol{\theta})}{\partial \theta_i},\\
\frac{\partial \mathbf{P}_k(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\partial \mathbf{P}_k^-(\boldsymbol{\theta})}{\partial \theta_i} - \frac{\partial \mathbf{K}_k(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{S}_k(\boldsymbol{\theta})\,\mathbf{K}_k^\mathsf{T}(\boldsymbol{\theta})\\
&\quad - \mathbf{K}_k(\boldsymbol{\theta})\,\frac{\partial \mathbf{S}_k(\boldsymbol{\theta})}{\partial \theta_i}\,\mathbf{K}_k^\mathsf{T}(\boldsymbol{\theta}) - \mathbf{K}_k(\boldsymbol{\theta})\,\mathbf{S}_k(\boldsymbol{\theta})\,\frac{\partial \mathbf{K}_k^\mathsf{T}(\boldsymbol{\theta})}{\partial \theta_i}.
\end{aligned}
\tag{A.16}
$$

The recursion should be started from the initial condition $\partial \varphi_0(\boldsymbol{\theta})/\partial \boldsymbol{\theta} = -\,\partial \log p(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$.

Another way to compute the same derivative is by using Fisher's identity (Equation 12.32) together with the expression for $\mathcal{Q}$ in Theorem 12.4. The result is the following.

Theorem A.3 (Energy function derivative for linear Gaussian model II) *The derivative of the energy function given in Theorem 12.3 can be computed as*

$$
\frac{\partial \varphi_T(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{\partial \log p(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \left.\frac{\partial \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}^{(n)}=\boldsymbol{\theta}},
\tag{A.17}
$$

where

$$
\begin{aligned}
&\left.\frac{\partial \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(n)})}{\partial \theta_i}\right|_{\boldsymbol{\theta}^{(n)}=\boldsymbol{\theta}} \\
&= -\frac{1}{2}\operatorname{tr}\left(\mathbf{P}_0^{-1}\frac{\partial \mathbf{P}_0}{\partial \theta_i}\right) - \frac{T}{2}\operatorname{tr}\left(\mathbf{Q}^{-1}\frac{\partial \mathbf{Q}}{\partial \theta_i}\right) - \frac{T}{2}\operatorname{tr}\left(\mathbf{R}^{-1}\frac{\partial \mathbf{R}}{\partial \theta_i}\right) \\
&\quad + \frac{1}{2}\operatorname{tr}\left\{\mathbf{P}_0^{-1}\frac{\partial \mathbf{P}_0}{\partial \theta_i}\mathbf{P}_0^{-1}\left[\mathbf{P}_0^{\mathrm{s}} + (\mathbf{m}_0^{\mathrm{s}} - \mathbf{m}_0)(\mathbf{m}_0^{\mathrm{s}} - \mathbf{m}_0)^{\mathsf{T}}\right]\right\} \\
&\quad + \frac{1}{2}\operatorname{tr}\left\{\mathbf{P}_0^{-1}\left[\frac{\partial \mathbf{m}_0}{\partial \theta_i}(\mathbf{m}_0^{\mathrm{s}} - \mathbf{m}_0)^{\mathsf{T}} + (\mathbf{m}_0^{\mathrm{s}} - \mathbf{m}_0)\frac{\partial \mathbf{m}_0^{\mathsf{T}}}{\partial \theta_i}\right]\right\} \\
&\quad + \frac{T}{2}\operatorname{tr}\left\{\mathbf{Q}^{-1}\frac{\partial \mathbf{Q}}{\partial \theta_i}\mathbf{Q}^{-1}\left[\boldsymbol{\Sigma} - \mathbf{C}\,\mathbf{A}^{\mathsf{T}} - \mathbf{A}\,\mathbf{C}^{\mathsf{T}} + \mathbf{A}\,\boldsymbol{\Phi}\,\mathbf{A}^{\mathsf{T}}\right]\right\} \\
&\quad - \frac{T}{2}\operatorname{tr}\left\{\mathbf{Q}^{-1}\left[-\mathbf{C}\frac{\partial \mathbf{A}^{\mathsf{T}}}{\partial \theta_i} - \frac{\partial \mathbf{A}}{\partial \theta_i}\mathbf{C}^{\mathsf{T}} + \frac{\partial \mathbf{A}}{\partial \theta_i}\boldsymbol{\Phi}\,\mathbf{A}^{\mathsf{T}} + \mathbf{A}\,\boldsymbol{\Phi}\frac{\partial \mathbf{A}^{\mathsf{T}}}{\partial \theta_i}\right]\right\} \\
&\quad + \frac{T}{2}\operatorname{tr}\left\{\mathbf{R}^{-1}\frac{\partial \mathbf{R}}{\partial \theta_i}\mathbf{R}^{-1}\left[\mathbf{D} - \mathbf{B}\,\mathbf{H}^{\mathsf{T}} - \mathbf{H}\,\mathbf{B}^{\mathsf{T}} + \mathbf{H}\,\boldsymbol{\Sigma}\,\mathbf{H}^{\mathsf{T}}\right]\right\} \\
&\quad - \frac{T}{2}\operatorname{tr}\left\{\mathbf{R}^{-1}\left[-\mathbf{B}\frac{\partial \mathbf{H}^{\mathsf{T}}}{\partial \theta_i} - \frac{\partial \mathbf{H}}{\partial \theta_i}\mathbf{B}^{\mathsf{T}} + \frac{\partial \mathbf{H}}{\partial \theta_i}\boldsymbol{\Sigma}\,\mathbf{H}^{\mathsf{T}} + \mathbf{H}\,\boldsymbol{\Sigma}\frac{\partial \mathbf{H}^{\mathsf{T}}}{\partial \theta_i}\right]\right\},
\end{aligned}
\tag{A.18}
$$

where all the terms are evaluated at $\boldsymbol{\theta}$.

A.4 Parameter derivatives for the Gaussian filter

In this section we consider the computation of the gradient of the Gaussian filtering based energy function in Algorithm 12.6 which was considered with models of the form

$$
\begin{aligned}
\mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta}) + \mathbf{q}_{k-1}, \\
\mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \boldsymbol{\theta}) + \mathbf{r}_k.
\end{aligned}
\tag{A.19}
$$

In order to compute the derivative, it is convenient to first rewrite the expectations as expectations over unit Gaussian distributions as follows:

$$
\begin{aligned}
\mathbf{m}_k^-(\boldsymbol{\theta}) &= \int \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta})\ \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}(\boldsymbol{\theta}), \mathbf{P}_{k-1}(\boldsymbol{\theta}))\ \mathrm{d}\mathbf{x}_{k-1} \\
&= \int \mathbf{f}(\mathbf{m}_{k-1}(\boldsymbol{\theta}) + \sqrt{\mathbf{P}_{k-1}(\boldsymbol{\theta})}\,\boldsymbol{\xi}, \boldsymbol{\theta})\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi}.
\end{aligned}
\tag{A.20}
$$

The derivative of this expression can now be computed as

$$\frac{\partial \mathbf{m}_k^-(\boldsymbol{\theta})}{\partial \theta_i} = \int \left[\mathbf{F}_x(\mathbf{m}_{k-1}(\boldsymbol{\theta}) + \sqrt{\mathbf{P}_{k-1}(\boldsymbol{\theta})}\,\boldsymbol{\xi}, \boldsymbol{\theta}) \times \left(\frac{\partial \mathbf{m}_{k-1}(\boldsymbol{\theta})}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_{k-1}(\boldsymbol{\theta})}}{\partial \theta_i}\,\boldsymbol{\xi} \right) + \frac{\partial \mathbf{f}}{\partial \theta_i}(\mathbf{m}_{k-1}(\boldsymbol{\theta}) + \sqrt{\mathbf{P}_{k-1}(\boldsymbol{\theta})}\,\boldsymbol{\xi}, \boldsymbol{\theta}) \right] \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\, \mathrm{d}\boldsymbol{\xi}.$$

Assuming that we are using the Cholesky factorization based matrix square root, the derivative $\partial \sqrt{\mathbf{P}_{k-1}(\boldsymbol{\theta})}/\partial \theta_i$ can be computed with Algorithm A.2 or Theorem A.1 given in Section A.2.

The above form is directly suitable for sigma-point methods, because they are based on the same change of variables. That is, the corresponding derivative of the sigma-point approximation will be

$$\frac{\partial \mathbf{m}_k^-(\boldsymbol{\theta})}{\partial \theta_i} \approx \sum_j W_j \left[\mathbf{F}_x(\mathbf{m}_{k-1}(\boldsymbol{\theta}) + \sqrt{\mathbf{P}_{k-1}(\boldsymbol{\theta})}\,\boldsymbol{\xi}^{(j)}, \boldsymbol{\theta}) \times \left(\frac{\partial \mathbf{m}_{k-1}(\boldsymbol{\theta})}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_{k-1}(\boldsymbol{\theta})}}{\partial \theta_i}\,\boldsymbol{\xi}^{(j)} \right) + \frac{\partial \mathbf{f}}{\partial \theta_i}(\mathbf{m}_{k-1}(\boldsymbol{\theta}) + \sqrt{\mathbf{P}_{k-1}(\boldsymbol{\theta})}\,\boldsymbol{\xi}^{(j)}, \boldsymbol{\theta}) \right], \tag{A.21}$$

where W_j and $\boldsymbol{\xi}^{(j)}$ are the weights and unit sigma points of the used sigma-point method. A nice thing in the above expression is that it is exactly the derivative of the sigma-point approximation to $\mathbf{m}_k^-(\boldsymbol{\theta})$.

The derivatives of the Gaussian filtering based energy function can now be expressed as follows.

Algorithm A.3 (Derivatives of Gaussian filtering based energy function) *The recursion for the derivative of the approximate energy function is*

$$\frac{\partial \varphi_k(\boldsymbol{\theta})}{\partial \theta_i} \simeq \frac{\partial \varphi_{k-1}(\boldsymbol{\theta})}{\partial \theta_i} + \frac{1}{2} \operatorname{tr}\left(\mathbf{S}_k^{-1}(\boldsymbol{\theta}) \frac{\partial \mathbf{S}_k(\boldsymbol{\theta})}{\partial \theta_i} \right) + \mathbf{v}_k^\top(\boldsymbol{\theta})\, \mathbf{S}_k^{-1}(\boldsymbol{\theta}) \frac{\partial \mathbf{v}_k(\boldsymbol{\theta})}{\partial \theta_i} - \frac{1}{2} \mathbf{v}_k^\top(\boldsymbol{\theta})\, \mathbf{S}_k^{-1}(\boldsymbol{\theta}) \frac{\partial \mathbf{S}_k(\boldsymbol{\theta})}{\partial \theta_i} \mathbf{S}_k^{-1}(\boldsymbol{\theta})\, \mathbf{v}_k(\boldsymbol{\theta}). \tag{A.22}$$

The derivatives of the prediction step are

$$\begin{aligned}
\frac{\partial \mathbf{m}_k^-}{\partial \theta_i} &= \int \left[\mathbf{F}_x \left(\mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) \left(\frac{\partial \mathbf{m}_{k-1}}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_{k-1}}}{\partial \theta_i}\, \boldsymbol{\xi} \right) \right. \\
&\quad \left. + \frac{\partial \mathbf{f}}{\partial \theta_i} \left(\mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) \right] \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\, \mathrm{d}\boldsymbol{\xi}, \\
\frac{\partial \mathbf{P}_k^-}{\partial \theta_i} &= \int \left[\mathbf{F}_x \left(\mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) \left(\frac{\partial \mathbf{m}_{k-1}}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_{k-1}}}{\partial \theta_i}\, \boldsymbol{\xi} \right) \right. \\
&\quad \left. + \frac{\partial \mathbf{f}}{\partial \theta_i} \left(\mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) - \frac{\partial \mathbf{m}_k^-}{\partial \theta_i} \right] \\
&\quad \times \left[\mathbf{f} \left(\mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) - \mathbf{m}_k^- \right]^{\mathsf{T}} \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\, \mathrm{d}\boldsymbol{\xi} \\
&+ \int \left[\mathbf{f} \left(\mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) - \mathbf{m}_k^- \right] \\
&\quad \times \left[\mathbf{F}_x \left(\mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) \left(\frac{\partial \mathbf{m}_{k-1}}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_{k-1}}}{\partial \theta_i}\, \boldsymbol{\xi} \right) \right. \\
&\quad \left. + \frac{\partial \mathbf{f}}{\partial \theta_i} \left(\mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) - \frac{\partial \mathbf{m}_k^-}{\partial \theta_i} \right]^{\mathsf{T}} \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\, \mathrm{d}\boldsymbol{\xi} + \frac{\partial \mathbf{Q}_{k-1}}{\partial \theta_i},
\end{aligned} \tag{A.23}$$

where $\mathbf{F}_x$ *is the Jacobian matrix of* $\mathbf{x} \mapsto \mathbf{f}(\mathbf{x}, \boldsymbol{\theta})$*. In the above expressions as well as in the following we have dropped the dependencies of various terms on the parameters* $\boldsymbol{\theta}$ *to simplify the notation. The derivatives of the update step are*

$$\begin{aligned}
\frac{\partial \boldsymbol{\mu}_k}{\partial \theta_i} &= \int \left[\mathbf{H}_x \left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) \left(\frac{\partial \mathbf{m}_k^-}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_k^-}}{\partial \theta_i}\, \boldsymbol{\xi} \right) \right. \\
&\quad \left. + \frac{\partial \mathbf{h}}{\partial \theta_i} \left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) \right] \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\, \mathrm{d}\boldsymbol{\xi}, \\
\frac{\partial \mathbf{v}_k}{\partial \theta_i} &= -\frac{\partial \boldsymbol{\mu}_k}{\partial \theta_i}, \\
\frac{\partial \mathbf{S}_k}{\partial \theta_i} &= \int \left[\mathbf{H}_x \left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\, \boldsymbol{\xi}, \boldsymbol{\theta} \right) \left(\frac{\partial \mathbf{m}_k^-}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_k^-}}{\partial \theta_i}\, \boldsymbol{\xi} \right) \right.
\end{aligned}$$

$$
\begin{aligned}
&+ \frac{\partial \mathbf{h}}{\partial \theta_i}\left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}, \boldsymbol{\theta}\right) - \frac{\partial \boldsymbol{\mu}_k}{\partial \theta_i}\Bigg] \\
&\quad \times \left[\mathbf{h}\left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}, \boldsymbol{\theta}\right) - \boldsymbol{\mu}_k\right]^\mathsf{T} \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\, \mathrm{d}\boldsymbol{\xi} \\
&+ \int \left[\mathbf{h}\left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}, \boldsymbol{\theta}\right) - \boldsymbol{\mu}_k\right] \\
&\quad \times \Bigg[\mathbf{H}_x\left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}, \boldsymbol{\theta}\right)\left(\frac{\partial \mathbf{m}_k^-}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_k^-}}{\partial \theta_i}\,\boldsymbol{\xi}\right) \\
&\qquad + \frac{\partial \mathbf{h}}{\partial \theta_i}\left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}, \boldsymbol{\theta}\right) - \frac{\partial \boldsymbol{\mu}_k}{\partial \theta_i}\Bigg]^\mathsf{T} \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\, \mathrm{d}\boldsymbol{\xi} + \frac{\partial \mathbf{R}_k}{\partial \theta_i}, \\
\frac{\partial \mathbf{C}_k}{\partial \theta_i} &= \int \left\{ \frac{\partial \sqrt{\mathbf{P}_k^-}}{\partial \theta_i}\,\boldsymbol{\xi}\left(\mathbf{h}\left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}, \boldsymbol{\theta}\right) - \boldsymbol{\mu}_k\right)^\mathsf{T} \right. \\
&\quad + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}\left[\mathbf{H}_x\left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}, \boldsymbol{\theta}\right)\left(\frac{\partial \mathbf{m}_k^-}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_k^-}}{\partial \theta_i}\,\boldsymbol{\xi}\right)\right. \\
&\qquad \left.\left. + \frac{\partial \mathbf{h}}{\partial \theta_i}\left(\mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}, \boldsymbol{\theta}\right) - \frac{\partial \boldsymbol{\mu}_k}{\partial \theta_i}\right]^\mathsf{T} \right\} \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\, \mathrm{d}\boldsymbol{\xi}, \\
\frac{\partial \mathbf{K}_k}{\partial \theta_i} &= \frac{\partial \mathbf{C}_k}{\partial \theta_i}\,\mathbf{S}_k^{-1} - \mathbf{C}_k\,\mathbf{S}_k^{-1}\,\frac{\partial \mathbf{S}_k}{\partial \theta_i}\,\mathbf{S}_k^{-1}, \\
\frac{\partial \mathbf{m}_k}{\partial \theta_i} &= \frac{\partial \mathbf{m}_k^-}{\partial \theta_i} + \frac{\partial \mathbf{K}_k}{\partial \theta_i}\,\mathbf{v}_k + \mathbf{K}_k\,\frac{\partial \mathbf{v}_k}{\partial \theta_i}, \\
\frac{\partial \mathbf{P}_k}{\partial \theta_i} &= \frac{\partial \mathbf{P}_k^-}{\partial \theta_i} - \frac{\partial \mathbf{K}_k}{\partial \theta_i}\,\mathbf{S}_k\,\mathbf{K}_k^\mathsf{T} - \mathbf{K}_k\,\frac{\partial \mathbf{S}_k}{\partial \theta_i}\,\mathbf{K}_k^\mathsf{T} - \mathbf{K}_k\,\mathbf{S}_k\,\frac{\partial \mathbf{K}_k^\mathsf{T}}{\partial \theta_i},
\end{aligned}
\tag{A.24}
$$

where $\mathbf{H}_x$ *is the Jacobian matrix of* $\mathbf{x} \mapsto \mathbf{h}(\mathbf{x}, \boldsymbol{\theta})$. *The derivatives of the Cholesky factors can be computed with Algorithm A.2 or Theorem A.1 given in Section A.2.*

The corresponding sigma-point approximations can be obtained by approximating the integrals analogously to Equation (A.21). The resulting derivative will be exact in the sense that it is the exact derivative of the corresponding sigma-point based approximation to the energy function.

We could also change back to the original variable, which gives, for example, the following representation for the derivative of the predicted

mean:

$$\frac{\partial \mathbf{m}_k^-}{\partial \theta_i} = \int \left[\mathbf{F}_x(\mathbf{x}_{k-1}, \boldsymbol{\theta})\, \mathbf{g}(\mathbf{x}_{k-1}, \boldsymbol{\theta}) + \frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \boldsymbol{\theta})}{\partial \theta_i} \right] \times \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1})\ \mathrm{d}\mathbf{x}_{k-1}, \tag{A.25}$$

where

$$\mathbf{g}(\mathbf{x}_{k-1}, \boldsymbol{\theta}) = \frac{\partial \mathbf{m}_{k-1}}{\partial \theta_i} + \frac{\partial \sqrt{\mathbf{P}_{k-1}}}{\partial \theta_i} \left(\sqrt{\mathbf{P}_{k-1}} \right)^{-1} (\mathbf{x}_{k-1} - \mathbf{m}_{k-1}). \tag{A.26}$$

The derivation of the full set of equations is left as an exercise to the reader.

References

Agamennoni, G., Nieto, J., and Nebot, E. 2011. An outlier-robust Kalman filter. Pages 1551–1558 of: *IEEE International Conference on Robotics and Automation (ICRA)*.

Akashi, H. and Kumamoto, H. 1977. Random sampling approach to state estimation in switching environments. *Automatica*, **13**(4), 429–434.

Alspach, D. L. and Sorenson, H. W. 1972. Nonlinear Bayesian estimation using Gaussian sum approximations. *IEEE Transactions on Automatic Control*, **17**(4).

Andrieu, C., De Freitas, N., and Doucet, A. 2002. Rao-Blackwellised particle filtering via data augmentation. In: Dietterich, T. G., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems 14*. MIT Press.

Andrieu, C., Doucet, A., Singh, S., and Tadic, V. 2004. Particle methods for change detection, system identification, and control. *Proceedings of the IEEE*, **92**(3), 423–438.

Andrieu, C. and Thoms, J. 2008. A tutorial on adaptive MCMC. *Statistics and Computing*, **18**(4), 343–373.

Andrieu, C., Doucet, A., and Holenstein, R. 2010. Particle Markov chain Monte Carlo methods. *The Royal Statistical Society: Series B (Statistical Methodology)*, **72**(3), 269–342.

Arasaratnam, I. and Haykin, S. 2009. Cubature Kalman filters. *IEEE Transactions on Automatic Control*, **54**(6), 1254–1269.

Arasaratnam, I. and Haykin, S. 2011. Cubature Kalman smoothers. *Automatica*, **47**(10), 2245–2250.

Arasaratnam, I., Haykin, S., and Elliott, R. J. 2007. Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature. *Proceedings of the IEEE*, **95**(5), 953–977.

Arasaratnam, I., Haykin, S., and Hurd, T. R. 2010. Cubature Kalman filtering for continuous-discrete systems: theory and simulations. *IEEE Transactions on Signal Processing*, **58**(10), 4977–4993.

Bar-Shalom, Y. and Li, X.-R. 1995. *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS Publishing.

Bar-Shalom, Y., Li, X.-R., and Kirubarajan, T. 2001. *Estimation with Applications to Tracking and Navigation*. Wiley.

Barber, D. 2006. Expectation correction for smoothed inference in switching linear dynamical systems. *The Journal of Machine Learning Research*, **7**, 2515–2540.

Barber, D. 2011. Approximate inference in switching linear dynamical systems using Gaussian mixtures. Chapter 8, pages 166–181 of: Barber, D., Cemgil, A. T., and Chiappa, S. (eds.), *Bayesian Time Series Models*. Cambridge University Press.

Berger, J. O. 1985. *Statistical Decision Theory and Bayesian Analysis*. Springer.

Bernardo, J. M. and Smith, A. F. M. 1994. *Bayesian Theory*. John Wiley & Sons.

Bierman, G. J. 1977. *Factorization Methods for Discrete Sequential Estimation*. Academic Press.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Blackman, S. and Popoli, R. 1999. *Design and Analysis of Modern Tracking Systems*. Artech House Radar Library.

Briers, M., Doucet, A., and Maskell, S. 2010. Smoothing algorithms for state-space models. *Annals of the Institute of Statistical Mathematics*, **62**(1), 61–89.

Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L. 2011. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC.

Cappé, O., Moulines, E., and Rydén, T. 2005. *Inference in Hidden Markov Models*. Springer.

Challa, S., Morelande, M. R., Mušicki, D., and Evans, R. J. 2011. *Fundamentals of Object Tracking*. Cambridge University Press.

Chen, R. and Liu, J. S. 2000. Mixture Kalman filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **62**(3), 493–508.

Cox, H. 1964. On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Transactions on Automatic Control*, **9**(1), 5–12.

Crassidis, J. L. and Junkins, J. L. 2004. *Optimal Estimation of Dynamic Systems*. Chapman & Hall/CRC.

Creal, D. 2012. A survey of sequential Monte Carlo methods for economics and finance. *Econometric Reviews*, **31**(3), 245–296.

Crisan, D. and Doucet, A. 2002. A survey of convergence results on particle filtering for practitioners. *IEEE Transactions on Signal Processing*, **50**(3), 736–746.

Crisan, D. and Rozovskii, B. (eds.) 2011. *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press.

Daum, F. and Huang, J. 2003. Curse of dimensionality and particle filters. Pages 1979–1993 of: *Proceedings of the IEEE Aerospace Conference*, vol. 4.

Deisenroth, M. P., Huber, M. F., and Hanebeck, U. D. 2009. Analytic moment-based Gaussian process filtering. In: *Proceedings of the 26th International Conference on Machine Learning*.

Deisenroth, M., Turner, R., Huber, M., Hanebeck, U., and Rasmussen, C. 2012. Robust filtering and smoothing with Gaussian processes. *IEEE Transactions on Automatic Control*, **57**(7), 1865–1871.

Dempster, A., Laird, N., and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, **39**(1), 1–38.

Djuric, P. and Miguez, J. 2002. Sequential particle filtering in the presence of additive Gaussian noise with unknown parameters. Pages 1621–1624 of: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2.

Douc, R., Garivier, A., Moulines, E., and Olsson, J. 2011. Sequential Monte Carlo smoothing for general state space hidden Markov models. *Annals of Applied Probability*, **21**(6), 2109–2145.

Doucet, A., Godsill, S. J., and Andrieu, C. 2000. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, **10**(3), 197–208.

Doucet, A., De Freitas, N., and Gordon, N. 2001. *Sequential Monte Carlo Methods in Practice*. Springer.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. 1987. Hybrid Monte Carlo. *Physics Letters B*, **195**(2), 216–222.

Fearnhead, P. 2002. Markov chain Monte Carlo, sufficient statistics, and particle filters. *Journal of Computational and Graphical Statistics*, **11**(4), 848–862.

Fearnhead, P. and Clifford, P. 2003. On-line inference for Hidden Markov models via particle filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **65**(4), 887–899.

Fong, W., Godsill, S. J., Doucet, A., and West, M. 2002. Monte Carlo smoothing with application to audio signal enhancement. *IEEE Transactions on Signal Processing*, **50**(2), 438–449.

Fraser, D. and Potter, J. 1969. The optimum linear smoother as a combination of two optimum linear filters. *IEEE Transactions on Automatic Control*, **14**(4), 387–390.

Gelb, A. 1974. *Applied Optimal Estimation*. MIT Press.

Gelb, A. and Vander Velde, W. 1968. *Multiple-Input Describing Functions and Nonlinear System Design*. McGraw-Hill.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. R. 2004. *Bayesian Data Analysis*. Second edn. Chapman & Hall.

Gilks, W., Richardson, S., and Spiegelhalter, D. (eds.) 1996. *Markov Chain Monte Carlo in Practice*. Chapman & Hall.

Godsill, S. J. and Rayner, P. J. 1998. *Digital Audio Restoration: a Statistical Model Based Approach*. Springer-Verlag.

Godsill, S. J., Doucet, A., and West, M. 2004. Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, **99**(465), 156–168.

Golub, G. H. and van Loan, C. F. 1996. *Matrix Computations*. Third edn. The Johns Hopkins University Press.

Golub, G. H. and Welsch, J. H. 1969. Calculation of Gauss quadrature rules. *Mathematics of Computation*, **23**(106), 221–230.

Gonzalez, R. C. and Woods, R. E. 2008. *Digital Image Processing*. Third edn. Prentice Hall.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. Pages 107–113 of: *IEEE Proceedings on Radar and Signal Processing*, vol. 140.

Grewal, M. S. and Andrews, A. P. 2001. *Kalman Filtering, Theory and Practice Using MATLAB*. Wiley.

Grewal, M. S., Miyasako, R. S., and Smith, J. M. 1988. Application of fixed point smoothing to the calibration, alignment and navigation data of inertial navigation systems. Pages 476–479 of: *Position Location and Navigation Symposium.*

Grewal, M. S., Weill, L. R., and Andrews, A. P. 2001. *Global Positioning Systems, Inertial Navigation and Integration*. Wiley.

Gupta, N. and Mehra, R. 1974. Computational aspects of maximum likelihood estimation and reduction in sensitivity function calculations. *IEEE Transactions on Automatic Control*, **19**(6), 774–783.

Gustafsson, F. and Hendeby, G. 2012. Some relations between extended and unscented Kalman filters. *IEEE Transactions on Signal Processing*, **60**(2), 545–555.

Haario, H., Saksman, E., and Tamminen, J. 1999. Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, **14**(3), 375–395.

Haario, H., Saksman, E., and Tamminen, J. 2001. An adaptive Metropolis algorithm. *Bernoulli*, **7**(2), 223–242.

Hartikainen, J. and Särkkä, S. 2010. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. Pages 379–384 of: *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*.

Hauk, O. 2004. Keep it simple: a case for using classical minimum norm estimation in the analysis of EEG and MEG data. *NeuroImage*, **21**(4), 1612–1621.

Hayes, M. H. 1996. *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, Inc.

Haykin, S. 2001. *Kalman Filtering and Neural Networks*. Wiley.

Hiltunen, P., Särkkä, S., Nissilä, I., Lajunen, A., and Lampinen, J. 2011. State space regularization in the nonstationary inverse problem for diffuse optical tomography. *Inverse Problems*, **27**, 025009.

Ho, Y. C. and Lee, R. C. K. 1964. A Bayesian approach to problems in stochastic estimation and control. *IEEE Transactions on Automatic Control*, **9**(4), 333–339.

Hu, X., Schön, T., and Ljung, L. 2008. A basic convergence result for particle filtering. *IEEE Transactions on Signal Processing*, **56**(4), 1337–1348.

Hu, X., Schön, T., and Ljung, L. 2011. A general convergence result for particle filtering. *IEEE Transactions on Signal Processing*, **59**(7), 3424–3429.

Hürzeler, M. and Kunsch, H. R. 1998. Monte Carlo approximations for general state-space models. *Journal of Computational and Graphical Statistics*, **7**(2), 175–193.

Ito, K. and Xiong, K. 2000. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, **45**(5), 910–927.

Jazwinski, A. H. 1966. Filtering for nonlinear dynamical systems. *IEEE Transactions on Automatic Control*, **11**(4), 765–766.

Jazwinski, A. H. 1970. *Stochastic Processes and Filtering Theory*. Academic Press.

Julier, S. J. and Uhlmann, J. K. 1995. *A General Method of Approximating Nonlinear Transformations of Probability Distributions*. Tech. rept. Robotics Research Group, Department of Engineering Science, University of Oxford.

Julier, S. J. and Uhlmann, J. K. 2004. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, **92**(3), 401–422.

Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. 1995. A new approach for filtering nonlinear systems. Pages 1628–1632 of: *Proceedings of the 1995 American Control, Conference, Seattle, Washington*.

Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. 2000. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, **45**(3), 477–482.

Kailath, T., Sayed, A. H., and Hassibi, B. 2000. *Linear Estimation*. Prentice Hall.

Kaipio, J. and Somersalo, E. 2005. *Statistical and Computational Inverse Problems*. Applied Mathematical Sciences, no. 160. Springer.

Kalman, R. E. 1960a. Contributions to the theory of optimal control. *Boletin de la Sociedad Matematica Mexicana*, **5**(1), 102–119.

Kalman, R. E. 1960b. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, **82**(1), 35–45.

Kalman, R. E. and Bucy, R. S. 1961. New results in linear filtering and prediction theory. *Transactions of the ASME, Journal of Basic Engineering*, **83**(3), 95–108.

Kantas, N., Doucet, A., Singh, S., and Maciejowski, J. 2009. An overview of sequential Monte Carlo methods for parameter estimation in general state-space models. In: *Proceedings IFAC Symposium on System Identification (SYSID)*.

Kaplan, E. D. 1996. *Understanding GPS, Principles and Applications*. Artech House.

Keeling, M. and Rohani, P. 2007. *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press.

Kelly, C. N. and Anderson, B. D. O. 1971. On the stability of fixed-lag smoothing algorithms. *Journal of Franklin Institute*, **291**(4), 271–281.

Kim, C.-J. 1994. Dynamic linear models with Markov-switching. *Journal of Econometrics*, **60**, 1–22.

Kitagawa, G. 1987. Non-Gaussian state-space modeling of nonstationary time series. *Journal of the American Statistical Association*, **82**(400), 1032–1041.

Kitagawa, G. 1994. The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, **46**(4), 605–623.

Kitagawa, G. 1996. Monte Carlo filter and smoother for Non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, **5**(1), 1–25.

Lee, R. C. K. 1964. *Optimal Estimation, Identification and Control*. MIT Press.

Lefebvre, T., Bruyninckx, H., and Schutter, J. D. 2002. Comment on "A new method for the nonlinear transformation of means and covariances in filters and estimators" [and authors' reply]. *IEEE Transactions on Automatic Control*, **47**(8), 1406–1409.

Leondes, C. T., Peller, J. B., and Stear, E. B. 1970. Nonlinear smoothing theory. *IEEE Transactions on Systems Science and Cybernetics*, **6**(1), 63–71.

Lin, F.-H., Wald, L. L., Ahlfors, S. P., Hämäläinen, M. S., Kwong, K. K., and Belliveau, J. W. 2006. Dynamic magnetic resonance inverse imaging of human brain function. *Magnetic Resonance in Medicine*, **56**(4), 787–802.

Lindsten, F. 2011. *Rao–Blackwellised Particle Methods for Inference and Identification*. Licentiate's thesis, Linköping University.

Liu, J. S. 2001. *Monte Carlo Strategies in Scientific Computing*. Springer.

Liu, J. S. and Chen, R. 1995. Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, **90**(430), 567–576.

Luenberger, D. G. and Ye, Y. 2008. *Linear and Nonlinear Programming*. Third edn. Springer.

Maybeck, P. 1982a. *Stochastic Models, Estimation and Control*. Vol. 3. Academic Press.

Maybeck, P. 1982b. *Stochastic Models, Estimation and Control*. Vol. 2. Academic Press.

Mbalawata, I. S., Särkkä, S., and Haario, H. 2013. Parameter estimation in stochastic differential equations with Markov chain Monte Carlo and non-linear Kalman filtering. *Computational Statistics*, **28**(3), 1195–1223.

Meditch, J. S. 1969. *Stochastic Optimal Linear Estimation and Control*. McGraw-Hill.

Milton, J. S. and Arnold, J. C. 1995. *Introduction to Probability and Statistics, Principles and Applications for Engineering and the Computing Sciences*. McGraw-Hill.

Moore, J. B. 1973. Discrete-time fixed-lag smoothing algorithms. *Automatica*, **9**(2), 163–174.

Moore, J. B. and Tam, P. 1973. Fixed-lag smoothing of nonlinear systems with discrete measurement. *Information Sciences*, **6**, 151–160.

Morf, M., Lévy, B., and Kailath, T. 1978. Square-root algorithms for the continuous-time linear least-square estimation problem. *IEEE Transactions on Automatic Control*, **23**(5), 907–911.

Murray, J. D. 1993. *Mathematical Biology*. Springer.

Murray, L. and Storkey, A. 2011. Particle smoothing in continuous time: a fast approach via density estimation. *IEEE Transactions on Signal Processing*, **59**(3), 1017–1026.

Neal, R. M. 2011. MCMC using Hamiltonian dynamics. Chapter 5 of: Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L. (eds.), *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC.

Neal, R. and Hinton, G. 1999. A view of the EM algorithm that justifies incremental, sparse, and other variants. Pages 355–370 of: Jordan, M. I. (ed.), *Learning in Graphical Models*. MIT Press.

Ninness, B. and Henriksen, S. 2010. Bayesian system identification via Markov chain Monte Carlo techniques. *Automatica*, **46**(1), 40–51.

Ninness, B., Wills, A., and Schön, T. B. 2010. Estimation of general nonlinear state-space systems. Pages 6371–6376 of: *Proceedings of the 49th IEEE Conference on Decision and Control (CDC), Atlanta, USA*.

Nørgaard, M., Poulsen, N. K., and Ravn, O. 2000. New developments in state estimation for nonlinear systems. *Automatica*, **36**(11), 1627–1638.

O'Hagan, A. 1991. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, **29**(3), 245–260.

Øksendal, B. 2003. *Stochastic Differential Equations: an Introduction with Applications*. Sixth edn. Springer-Verlag.

Olsson, R., Petersen, K., and Lehn-Schiøler, T. 2007. State-space models: from the EM algorithm to a gradient approach. *Neural Computation*, **19**(4), 1097–1111.

Piché, R., Särkkä, S., and Hartikainen, J. 2012. Recursive outlier-robust filtering and smoothing for nonlinear systems using the multivariate Student-t distribution. In: *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*.

Pitt, M. K. and Shephard, N. 1999. Filtering via simulation: auxiliary particle filters. *Journal of the American Statistical Association*, **94**(446), 590–599.

Poyiadjis, G., Doucet, A., and Singh, S. 2011. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, **98**(1), 65–80.

Proakis, J. G. 2001. *Digital Communications*. Fourth edn. McGraw-Hill.

Punskaya, E., Doucet, A., and Fitzgerald, W. J. 2002. On the use and misuse of particle filtering in digital communications. In: *Proceedings of EUSIPCO*.

Raiffa, H. and Schlaifer, R. 2000. *Applied Statistical Decision Theory*. John Wiley & Sons, Wiley Classics Library.

Rasmussen, C. E. and Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Rauch, H. E. 1963. Solutions to the linear smoothing problem. *IEEE Transactions on Automatic Control*, **8**(4), 371–372.

Rauch, H. E., Tung, F., and Striebel, C. T. 1965. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, **3**(8), 1445–1450.

Ristic, B., Arulampalam, S., and Gordon, N. 2004. *Beyond the Kalman Filter*. Artech House.

Roberts, G. O. and Rosenthal, J. S. 2001. Optimal scaling for various Metropolis–Hastings algorithms. *Statistical Science*, **16**(4), 351–367.

Roweis, S. and Ghahramani, Z. 2001. Learning nonlinear dynamical systems using the expectation–maximization algorithm. Chapter 6, pages 175–220 of: Haykin, S. (ed.), *Kalman Filtering and Neural Networks*. Wiley-Interscience.

Sage, A. P. and Melsa, J. L. 1971. *Estimation Theory with Applications to Communications and Control*. McGraw-Hill.

Saha, S., Ozkan, E., Gustafsson, F., and Smidl, V. 2010. Marginalized particle filters for Bayesian estimation of Gaussian noise parameters. Pages 1–8 of: *13th Conference on Information Fusion (FUSION)*.

Sandblom, F. and Svensson, L. 2012. Moment estimation using a marginalized transform. *IEEE Transactions on Signal Processing*, **60**(12), 6138–6150.

Särkkä, S. 2006. *Recursive Bayesian Inference on Stochastic Differential Equations*. Doctoral dissertation, Helsinki University of Technology.

Särkkä, S. 2007. On unscented Kalman filtering for state estimation of continuous-time nonlinear systems. *IEEE Transactions on Automatic Control*, **52**(9), 1631–1641.

Särkkä, S. 2008. Unscented Rauch-Tung-Striebel smoother. *IEEE Transactions on Automatic Control*, **53**(3), 845–849.

Särkkä, S. 2010. Continuous-time and continuous-discrete-time unscented Rauch-Tung-Striebel smoothers. *Signal Processing*, **90**(1), 225–235.

Särkkä, S. 2011. Linear operators and stochastic partial differential equations in Gaussian process regression. In: *Proceedings of ICANN*.

Särkkä, S. and Hartikainen, J. 2010a. On Gaussian optimal smoothing of non-linear state space models. *IEEE Transactions on Automatic Control*, **55**(8), 1938–1941.

Särkkä, S. and Hartikainen, J. 2010b. Sigma point methods in optimal smoothing of non-linear stochastic state space models. Pages 184–189 of: *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*.

Särkkä, S. and Hartikainen, J. 2012. Infinite-dimensional Kalman filtering approach to spatio-temporal Gaussian process regression. In: *Proceedings of AISTATS 2012*.

Särkkä, S. and Nummenmaa, A. 2009. Recursive noise adaptive Kalman filtering by variational Bayesian approximations. *IEEE Transactions on Automatic Control*, **54**(3), 596–600.

Särkkä, S. and Sarmavuori, J. 2013. Gaussian filtering and smoothing for continuous-discrete dynamic systems. *Signal Processing*, **93**(2), 500–510.

Särkkä, S. and Solin, A. 2012. On continuous-discrete cubature Kalman filtering. Pages 1210–1215 of: *Proceedings of SYSID 2012*.

Särkkä, S. and Sottinen, T. 2008. Application of Girsanov theorem to particle filtering of discretely observed continuous-time non-linear systems. *Bayesian Analysis*, **3**(3), 555–584.

Särkkä, S., Vehtari, A., and Lampinen, J. 2007a. CATS benchmark time series prediction by Kalman smoother with cross-validated noise density. *Neurocomputing*, **70**(13–15), 2331–2341.

Särkkä, S., Vehtari, A., and Lampinen, J. 2007b. Rao-Blackwellized particle filter for multiple target tracking. *Information Fusion Journal*, **8**(1), 2–15.

Särkkä, S., Bunch, P., and Godsill, S. J. 2012a. A backward-simulation based Rao-Blackwellized particle smoother for conditionally linear Gaussian models. Pages 506–511 of: *Proceedings of SYSID 2012*.

Särkkä, S., Solin, A., Nummenmaa, A., Vehtari, A., Auranen, T., Vanni, S., and Lin, F.-H. 2012b. Dynamic retrospective filtering of physiological noise in BOLD fMRI: DRIFTER. *NeuroImage*, **60**(2), 1517–1527.

Sarmavuori, J. and Särkkä, S. 2012a. Fourier-Hermite Kalman filter. *IEEE Transactions on Automatic Control*, **57**(6), 1511–1515.

Sarmavuori, J. and Särkkä, S. 2012b. Fourier-Hermite Rauch-Tung-Striebel Smoother. In: *Proceedings of EUSIPCO*.

Schön, T. and Gustafsson, F. 2003. Particle filters for system identification of state-space models linear in either parameters or states. Pages 1287–1292 of: *Proceedings of the 13th IFAC Symposium on System Identification, Rotterdam, The Netherlands*.

Schön, T., Gustafsson, F., and Nordlund, P.-J. 2005. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, **53**(7), 2279–2289.

Schön, T., Wills, A., and Ninness, B. 2011. System identification of nonlinear state-space models. *Automatica*, **47**(1), 39–49.

Segal, M. and Weinstein, E. 1989. A new method for evaluating the log-likelihood gradient, the Hessian, and the Fisher information matrix for linear dynamic systems. *IEEE Transactions on Information Theory*, **35**(3), 682–687.

Shiryaev, A. N. 1996. *Probability*. Springer.

Shumway, R. and Stoffer, D. 1982. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, **3**(4), 253–264.

Šimandl, M. and Duník, J. 2006. Design of derivative-free smoothers and predictors. Pages 991–996 of: *Preprints of the 14th IFAC Symposium on System Identification*.

Singer, H. 2008. Nonlinear continuous time modeling approaches in panel research. *Statistica Neerlandica*, **62**(1), 29–57.

Singer, H. 2011. Continuous-discrete state-space modeling of panel data with nonlinear filter algorithms. *AStA Advances in Statistical Analysis*, **95**(4), 375–413.

Snyder, C., Bengtsson, T., Bickel, P., and Anderson, J. 2008. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, **136**(12), 4629–4640.

Stengel, R. F. 1994. *Optimal Control and Estimation*. Dover.

Stone, L. D., Barlow, C. A., and Corwin, T. L. 1999. *Bayesian Multiple Target Tracking*. Artech House.

Storvik, G. 2002. Particle filters in state space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing*, **50**(2), 281–289.

Stratonovich, R. L. 1968. *Conditional Markov Processes and Their Application to the Theory of Optimal Control*. Elsevier.

Striebel, C. T. 1965. Partial differential equations for the conditional distribution of a Markov process given noisy observations. *Journal of Mathematical Analysis and Applications*, **11**, 151–159.

Tam, P., Tam, D., and Moore, J. 1973. Fixed-lag demodulation of discrete noisy measurements of FM signals. *Automatica*, **9**(6), 725–729.

Tarantola, A. 2004. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM.

Titterton, D. H. and Weston, J. L. 1997. *Strapdown Inertial Navigation Technology*. Peter Peregrinus Ltd.

Väänänen, V. 2012. *Gaussian Filtering and Smoothing Based Parameter Estimation in Nonlinear Models for Sequential Data*. Master's Thesis, Aalto University.

Van der Merwe, R. and Wan, E. 2003. Sigma-point Kalman filters for probabilistic inference in dynamic state-space models. In: *Proceedings of the Workshop on Advances in Machine Learning*.

Van der Merwe, R. and Wan, E. A. 2001. The square-root unscented Kalman filter for state and parameter estimation. Pages 3461–3464 of: *International Conference on Acoustics, Speech, and Signal Processing*.

Van der Merwe, R., De Freitas, N., Doucet, A., and Wan, E. 2001. The unscented particle filter. Pages 584–590 of: *Advances in Neural Information Processing Systems 13*.

Van Trees, H. L. 1968. *Detection, Estimation, and Modulation Theory Part I*. John Wiley & Sons.

Van Trees, H. L. 1971. *Detection, Estimation, and Modulation Theory Part II*. John Wiley & Sons.

Vihola, M. 2012. Robust adaptive Metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, **22**(5), 997–1008.

Viterbi, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, **13**(2).

Wan, E. A. and Van der Merwe, R. 2001. The unscented Kalman filter. Chapter 7 of: Haykin, S. (ed.), *Kalman Filtering and Neural Networks*. Wiley.

West, M. and Harrison, J. 1997. *Bayesian Forecasting and Dynamic Models*. Springer-Verlag.

Wiener, N. 1950. *Extrapolation, Interpolation and Smoothing of Stationary Time Series with Engineering Applications*. John Wiley & Sons.

Wills, A., Schön, T. B., Ljung, L., and Ninness, B. 2013. Identification of Hammerstein–Wiener models. *Automatica*, **49**(1), 70–81.

Wu, Y., Hu, D., Wu, M., and Hu, X. 2005. Unscented Kalman filtering for additive noise case: augmented versus nonaugmented. *IEEE Signal Processing Letters*, **12**(5), 357–360.

Wu, Y., Hu, D., Wu, M., and Hu, X. 2006. A numerical-integration perspective on Gaussian filters. *IEEE Transactions on Signal Processing*, **54**(8), 2910–2921.

Ypma, A. and Heskes, T. 2005. Novel approximations for inference in nonlinear dynamical systems using expectation propagation. *Neurocomputing*, **69**(1), 85–99.

Zoeter, O. and Heskes, T. 2011. Expectation propagation and generalized EP methods for inference in switching linear dynamical systems. Chapter 7, pages 141–165 of: *Bayesian Time Series Models*. Cambridge University Press.

Index